Schulinternen Lehrplan
zum Kernlehrplan für die gymnasiale Oberstufe

Informatik

(Stand: 01.06.2018)

Dieses Beispiel für einen schulinternen Lehrplan wird um Angaben für den Leistungskurs ergänzt.

Inhalt

Seite

1 Die Fachgruppe Informatik des Albert-Schweitzer-Gymnasium, Hürth 3

2 Entscheidungen zum Unterricht 6

2.1 Unterrichtsvorhaben 6
2.1.1 Übersichtsraster Unterrichtsvorhaben 8
2.1.2 Konkretisierte Unterrichtsvorhaben 22
2.2 Grundsätze der fachmethodischen und fachdidaktischen Arbeit 23
2.3 Grundsätze der Leistungsbewertung und Leistungsrückmeldung 115

3 Entscheidungen zu fach- und unterrichtsübergreifenden Fragen 119

4 Qualitätssicherung und Evaluation 121

Hinweis: Als Beispiel für einen schulinternen Lehrplan auf der Grundlage des Kernlehrplans Informatik steht hier der schulinterne Lehrplan einer fiktiven Schule zur Verfügung. Diese Schule verfügt zurzeit über keine Leistungskurse in Informatik. Daher liegen keine Planungen für entsprechende Unterrichtsvorhaben vor.

Als Orientierung für Schulen mit Leistungskurs ist eine Übersicht über mögliche Unterrichtsvorhaben für den Leistungskurs in Arbeit.

Um zu verdeutlichen, wie die jeweils spezifischen Rahmenbedingungen in den schulinternen Lehrplan einfließen, wird die Schule in Kapitel 1 zunächst näher vorgestellt. Den Fachkonferenzen wird empfohlen, eine nach den Aspekten im vorliegenden Beispiel strukturierte Beschreibung für ihre Schule zu erstellen.

1 Die Fachgruppe Informatik des Konrad-Zuse-Gymnasiums Paderborn

Hinweis: Um die Ausgangsbedingungen für die Erstellung des schulinternen Lehrplans festzuhalten, können beispielsweise folgende Aspekte berücksichtigt werden:

- Lage der Schule
- Funktionen und Aufgaben der Fachgruppe vor dem Hintergrund des Schulprogramms
- Beitrag der Fachgruppe zur Erreichung der Erziehungsziele ihrer Schule
- Beitrag zur Qualitätssicherung und –entwicklung innerhalb der Fachgruppe
- Zusammenarbeit mit anderen Fachgruppen (fächerübergreifende Unterrichtsvorhaben und Projekte)
- Ressourcen der Schule (personell, räumlich, sächlich), Größe der Lerngruppen, Unterrichtstaktung, Stundenverortung
- Name des/der Fachvorsitzenden und des Stellvertreters/der Stellvertreterin
- ggf. Arbeitsgruppen bzw. weitere Beauftragte

Beim Konrad-Zuse-Gymnasium handelt es sich um eine dreizügige Schule im Zentrum von Paderborn mit zurzeit ca. 700 Schülerinnen und Schülern, 43 Planstellen und 60 Lehrerinnen und Lehrern. Das Einzugsgebiet der Schule umfasst den größten Teil der Paderborner Innenstadt sowie umliegender Städte was zum Teil auf das Angebot der Schule im Fach Informatik zurückzuführen ist. Im Bereich der Sekundarstufe II kooperiert das Konrad-

Zuse-Gymnasium mit seinen Nachbarschulen und bietet mit ihnen zahlreiche gemeinsame Kurse an, unter anderem in Informatik.
Das Fach Informatik wird am Konrad-Zuse-Gymnasium ab der Jahrgangsstufe 8 im Wahlpflichtbereich II (WP II) dreistündig unterrichtet und von etwa der Hälfte der Schülerinnen und Schüler besucht. In der zweijährigen Laufzeit dieser Kurse wird in altersstufengerechter Weise unter anderem auf Grundlagen der Algorithmik am Beispiel einer didaktischen Lernumgebung, auf die technische Informatik am Beispiel von Schaltwerken und Schaltnetzen und auf Robotik eingegangen. Der Unterricht erfolgt dabei in enger Verzahnung mit Inhalten der Mathematik und Physik und wird zum Teil in Form von fächerverbindenden Projekten und in Kooperation mit außerschulischen Partnern gestaltet.

In den Jahrgangsstufen 5 und 6 wird ein für alle verpflichtender Kurs zum Umgang mit informatischen Systemen durchgeführt, ein sogenannter *Office-Kurs*, der jedoch nicht unmittelbar dem Fach Informatik zuzuordnen ist.

Organisatorisch ist das Fach Informatik in der Sekundarstufe I in den MINT-Zweig der Schule eingebunden, den Schülerinnen und Schüler als Alternative zu einem bilingualen Zweig anwählen können.

In der Sekundarstufe II bietet das Konrad-Zuse-Gymnasium für die eigenen Schülerinnen und Schüler in allen Jahrgangsstufen jeweils einen Grundkurs in Informatik an. Zukünftig sollen potentiell auch Leistungskurse in Informatik angeboten werden.

Um insbesondere Schülerinnen und Schülern gerecht zu werden, die in der Sekundarstufe I keinen Informatikunterricht besucht haben, wird in Kursen der Einführungsphase besonderer Wert darauf gelegt, dass keine Vorkenntnisse aus der Sekundarstufe I zum erfolgreichen Durchlaufen des Kurses erforderlich sind.

Der Unterricht der Sekundarstufe II wird mit Hilfe der Programmiersprache Java durchgeführt. In der Einführungsphase kommt dabei zusätzlich eine didaktische Bibliothek zum Einsatz, welche das Erstellen von grafischen Programmen erleichtert.

Durch projektartiges Vorgehen, offene Aufgaben und Möglichkeiten, Problemlösungen zu verfeinern oder zu optimieren, entspricht der Informatikunterricht der Oberstufe in besonderem Maße den Erziehungszielen, Leistungsbereitschaft zu fördern, ohne zu überfordern.

Die gemeinsame Entwicklung von Materialien und Unterrichtsvorhaben, die Evaluation von Lehr- und Lernprozessen sowie die stetige Überprüfung und eventuelle Modifikation des schulinternen Curriculums durch die

Fachkonferenz Informatik stellen einen wichtigen Beitrag zur Qualitätssicherung und -entwicklung des Unterrichts dar.

Zurzeit besteht die Fachschaft Informatik des Konrad-Zuse-Gymnasiums aus zwei Lehrkräften, denen zwei Computerräume mit 31 bzw. 16 Computerarbeitsplätzen und ein Selbstlernzentrum mit 12 Plätzen zur Verfügung stehen. Alle Arbeitsplätze sind an das schulinterne Rechnernetz angeschlossen, so dass Schülerinnen und Schüler über einen individuell gestaltbaren Zugang zum zentralen Server der Schule alle Arbeitsplätze der drei Räume zum Zugriff auf ihre eigenen Daten, zur Recherche im Internet oder zur Bearbeitung schulischer Aufgaben verwenden können.

Der Unterricht erfolgt im 45-Minuten-Takt. Die Kursblockung sieht grundsätzlich für Grundkurse eine Doppelstunde und eine Einzelstunde vor.

2 Entscheidungen zum Unterricht

Hinweis: Die nachfolgend dargestellte Umsetzung der verbindlichen Kompetenzerwartungen des Kernlehrplans findet auf zwei Ebenen statt. Das **Übersichtsraster** gibt den Lehrkräften einen raschen Überblick über die laut Fachkonferenz verbindlichen Unterrichtsvorhaben pro Schuljahr. In dem Raster sind außer dem Thema des jeweiligen Vorhabens das schwerpunktmäßig damit verknüpfte Inhaltsfeld bzw. die Inhaltsfelder, inhaltliche Schwerpunkte des Vorhabens sowie Schwerpunktkompetenzbereiche ausgewiesen. Die **Konkretisierung von Unterrichtsvorhaben** führt weitere Kompetenzerwartungen auf und verdeutlicht vorhabenbezogene Absprachen, z.B. zur Festlegung auf einen Aufgabentyp bei der Lernerfolgsüberprüfung durch eine Klausur.

2.1 Unterrichtsvorhaben

Die Darstellung der Unterrichtsvorhaben im schulinternen Lehrplan besitzt den Anspruch, sämtliche im Kernlehrplan angeführten Kompetenzen abzudecken. Dies entspricht der Verpflichtung jeder Lehrkraft, Schülerinnen und Schülern Lerngelegenheiten zu ermöglichen, so dass alle Kompetenzerwartungen des Kernlehrplans von ihnen erfüllt werden können.

Die entsprechende Umsetzung erfolgt auf zwei Ebenen: der Übersichts- und der Konkretisierungsebene.

Im „Übersichtsraster Unterrichtsvorhaben“ (Kapitel 2.1.1) wird die für alle Lehrerinnen und Lehrer gemäß Fachkonferenzbeschluss verbindliche Verteilung der Unterrichtsvorhaben dargestellt. Das Übersichtsraster dient dazu, den Kolleginnen und Kollegen einen schnellen Überblick über die Zuordnung der Unterrichtsvorhaben zu den einzelnen Jahrgangsstufen sowie den im Kernlehrplan genannten Kompetenzen, Inhaltsfeldern und inhaltlichen Schwerpunkten zu verschaffen. Der ausgewiesene Zeitbedarf versteht sich als grobe Orientierungsgröße, die nach Bedarf über- oder unterschritten werden kann. Um Freiraum für Vertiefungen, besondere Schülerinteressen, aktuelle Themen bzw. die Erfordernisse anderer besonderer Ereignisse (z.B. Praktika, Kursfahrten o.ä.) zu erhalten, wurden im Rahmen dieses schulinternen Lehrplans ca. 75 Prozent der Bruttounterrichtszeit verplant.

Während der Fachkonferenzbeschluss zum „Übersichtsraster Unterrichtsvorhaben“ zur Gewährleistung vergleichbarer Standards sowie zur Absicherung von Lerngruppenübertritten und Lehrkraftwechseln für alle Mitglieder der Fachkonferenz Bindekraft entfalten soll, beinhaltet die Ausweisung „konkretisierter Unterrichtsvorhaben“ (Kapitel 2.1.2) Beispiele und Materialien, die empfehlenden Charakter haben. Referendarinnen und

Referendaren sowie neuen Kolleginnen und Kollegen dienen diese vor allem zur standardbezogenen Orientierung in der neuen Schule, aber auch zur Verdeutlichung von unterrichtsbezogenen fachgruppeninternen Absprachen zu didaktisch-methodischen Zugängen, fächerübergreifenden Kooperationen, Lernmitteln und -orten sowie vorgesehenen Leistungsüberprüfungen, die im Einzelnen auch den Kapiteln 2.2 bis 2.3 zu entnehmen sind.

Da in den folgenden Unterrichtsvorhaben Inhalte in der Regel anhand von Problemstellungen in Anwendungskontexten bearbeitet werden, werden in einigen Unterrichtsvorhaben jeweils mehrere Inhaltsfelder angesprochen.

2.1.1 Übersichtsraster Unterrichtsvorhaben

I) Einführungsphase

Einführungsphase	
Unterrichtsvorhaben E-I **Thema:** *Einführung in die Nutzung von Informatiksystemen und in grundlegende Begrifflichkeiten* **Zentrale Kompetenzen:** • Argumentieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Informatiksysteme • Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** • Einzelrechner • Dateisystem • Internet • Einsatz von Informatiksystemen **Zeitbedarf**: 6 Stunden	*Unterrichtsvorhaben E-II* **Thema**: *Grundlagen der objektorientierten Analyse, Modellierung und Implementierung anhand von statischen Grafikszenen* **Zentrale Kompetenzen:** • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Formale Sprachen und Automaten **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Syntax und Semantik einer Programmiersprache **Zeitbedarf**: 8 Stunden

Einführungsphase	
Unterrichtsvorhaben E-III **Thema**: *Grundlagen der objektorientierten Programmierung und algorithmischer Grundstrukturen in Java anhand von einfachen Animationen* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Syntax und Semantik einer Programmiersprache • Analyse, Entwurf und Implementierung einfacher Algorithmen **Zeitbedarf**: 18 Stunden	*Unterrichtsvorhaben E-IV* **Thema**: *Modellierung und Implementierung von Klassen- und Objektbeziehungen anhand von grafischen Spielen und Simulationen* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Syntax und Semantik einer Programmiersprache • Analyse, Entwurf und Implementierung einfacher Algorithmen **Zeitbedarf**: 18 Stunden

<table>
<tr><th colspan="2">Einführungsphase</th></tr>
<tr>
<td><u>Unterrichtsvorhaben E-V</u>

Thema:
Such- und Sortieralgorithmen anhand kontextbezogener Beispiele

Zentrale Kompetenzen:
• Argumentieren
• Modellieren
• Darstellen und Interpretieren
• Kommunizieren und Kooperieren

Inhaltsfelder:
• Algorithmen

Inhaltliche Schwerpunkte:
• Algorithmen zum Suchen und Sortieren
• Analyse, Entwurf und Implementierung einfacher Algorithmen

Zeitbedarf: 9 Stunden</td>
<td><u>Unterrichtsvorhaben E-VI</u>

Thema:
Geschichte der digitalen Datenverarbeitung und die Grundlagen des Datenschutzes

Zentrale Kompetenzen:
• Argumentieren
• Darstellen und Interpretieren
• Kommunizieren und Kooperieren

Inhaltsfelder:
• Informatik, Mensch und Gesellschaft
• Informatiksysteme

Inhaltliche Schwerpunkte:
• Wirkungen der Automatisierung
• Geschichte der automatischen Datenverarbeitung
• Digitalisierung

Zeitbedarf: 15 Stunden</td>
</tr>
<tr><td colspan="2">Summe Einführungsphase: 74</td></tr>
</table>

II) Qualifikationsphase (Q1 und Q2) - GRUNDKURS

<table>
<tr><th colspan="2">Qualifikationsphase 1</th></tr>
<tr>
<td>

Unterrichtsvorhaben Q1-I

Thema:
Wiederholung der objektorientierten Modellierung und Programmierung anhand einer kontextbezogenen Problemstellung

Zentrale Kompetenzen:
- Argumentieren
- Modellieren
- Implementieren
- Darstellen und Interpretieren
- Kommunizieren und Kooperieren

Inhaltsfelder:
- Daten und ihre Strukturierung
- Algorithmen
- Formale Sprachen und Automaten
- Informatiksysteme

Inhaltliche Schwerpunkte:
- Objekte und Klassen
- Analyse, Entwurf und Implementierung von Algorithmen
- Syntax und Semantik einer Programmiersprache
- Nutzung von Informatiksystemen

Zeitbedarf: 8 Stunden

</td>
<td>

Unterrichtsvorhaben Q1-II

Thema:
Modellierung und Implementierung von Anwendungen mit dynamischen, linearen Datenstrukturen

Zentrale Kompetenzen:
- Argumentieren
- Modellieren
- Implementieren
- Darstellen und Interpretieren
- Kommunizieren und Kooperieren

Inhaltsfelder:
- Daten und ihre Strukturierung
- Algorithmen
- Formale Sprachen und Automaten

Inhaltliche Schwerpunkte:
- Objekte und Klassen
- Analyse, Entwurf und Implementierung von Algorithmen
- Algorithmen in ausgewählten informatischen Kontexten
- Syntax und Semantik einer Programmiersprache

Zeitbedarf: 20 Stunden

</td>
</tr>
</table>

Qualifikationsphase 1	
Unterrichtsvorhaben Q1-III **Thema:** *Suchen und Sortieren auf linearen Datenstrukturen* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Algorithmen • Formale Sprachen und Automaten **Inhaltliche Schwerpunkte:** • Analyse, Entwurf und Implementierung von Algorithmen • Algorithmen in ausgewählten informatischen Kontexten • Syntax und Semantik einer Programmiersprache **Zeitbedarf**: 16 Stunden	*Unterrichtsvorhaben Q1-IV* **Thema:** *Modellierung und Nutzung von relationalen Datenbanken in Anwendungskontexten* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten • Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** • Datenbanken • Algorithmen in ausgewählten informatischen Kontexten • Syntax und Semantik einer Programmiersprache • Sicherheit **Zeitbedarf**: 20 Stunden

Qualifikationsphase 1	
Unterrichtsvorhaben Q1-V **Thema:** *Sicherheit und Datenschutz in Netzstrukturen* **Zentrale Kompetenzen:** • Argumentieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Informatiksysteme • Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** • Einzelrechner und Rechnernetzwerke • Sicherheit • Nutzung von Informatiksystemen, Wirkungen der Automatisierung **Zeitbedarf**: 10 Stunden	
Summe Qualifikationsphase 1: 74 Stunden	

Qualifikationsphase 2	
Unterrichtsvorhaben Q2-I **Thema:** *Modellierung und Implementierung von Anwendungen mit dynamischen, nichtlinearen Datenstrukturen* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Analyse, Entwurf und Implementierung von Algorithmen • Algorithmen in ausgewählten informatischen Kontexten • Syntax und Semantik einer Programmiersprache **Zeitbedarf**: 24 Stunden	*Unterrichtsvorhaben Q2-II* **Thema:** *Endliche Automaten und formale Sprachen* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Endliche Automaten und formale Sprachen **Inhaltliche Schwerpunkte:** • Endliche Automaten • Grammatiken regulärer Sprachen • Möglichkeiten und Grenzen von Automaten und formalen Sprachen **Zeitbedarf**: 20 Stunden

Qualifikationsphase 2	
Unterrichtsvorhaben Q2-III **Thema:** *Prinzipielle Arbeitsweise eines Computers und Grenzen der Automatisierbarkeit* **Zentrale Kompetenzen:** • Argumentieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Informatiksysteme • Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** • Einzelrechner und Rechnernetzwerke • Grenzen der Automatisierung **Zeitbedarf:** 12 Stunden	
Summe Qualifikationsphase 2: 56 Stunden	

II) Qualifikationsphase (Q1 und Q2) - LEISTUNGSKURS

<table>
<tr><th colspan="2">Qualifikationsphase 1</th></tr>
<tr>
<td>

<u>*Unterrichtsvorhaben Q1-I*</u>

Thema:

Wiederholung der objektorientierten Modellierung und Programmierung anhand einer kontextbezogenen Problemstellung unter Berücksichtigung der Gestaltung einer Benutzungsoberfläche

Zentrale Kompetenzen:

- Argumentieren
- Modellieren
- Implementieren
- Darstellen und Interpretieren
- Kommunizieren und Kooperieren

Inhaltsfelder:

- Daten und ihre Strukturierung
- Algorithmen
- Formale Sprachen und Automaten
- Informatiksysteme

Inhaltliche Schwerpunkte:

- Objekte und Klassen
- Analyse, Entwurf und Implementierung von Algorithmen
- Syntax und Semantik einer Programmiersprache
- Nutzung von Informatiksystemen

Zeitbedarf: 15 Stunden

</td>
<td>

<u>*Unterrichtsvorhaben Q1-II*</u>

Thema:

Modellierung und Implementierung von dynamischen linearen Datenstrukturen und von Anwendungen mit dynamischen linearen Datenstrukturen in kontextbezogenen Problemstellungen

Zentrale Kompetenzen:

- Argumentieren
- Modellieren
- Implementieren
- Darstellen und Interpretieren
- Kommunizieren und Kooperieren

Inhaltsfelder:

- Daten und ihre Strukturierung
- Algorithmen
- Formale Sprachen und Automaten

Inhaltliche Schwerpunkte:

- Objekte und Klassen
- Analyse, Entwurf und Implementierung von Algorithmen
- Algorithmen in ausgewählten informatischen Kontexten
- Syntax und Semantik einer Programmiersprache

Zeitbedarf: 25 Stunden

</td>
</tr>
</table>

Qualifikationsphase 1

Unterrichtsvorhaben Q1-III

Thema:

Modellierung, Implementierung, Analyse und Beurteilung von Such- und Sortierverfahren unterschiedlicher Komplexitätsklassen in kontextbezogenen Problemstellungen

Zentrale Kompetenzen:

- Argumentieren
- Modellieren
- Implementieren
- Darstellen und Interpretieren
- Kommunizieren und Kooperieren

Inhaltsfelder:

- Algorithmen
- Formale Sprachen und Automaten

Inhaltliche Schwerpunkte:

- Analyse, Entwurf und Implementierung von Algorithmen
- Algorithmen in ausgewählten informatischen Kontexten
- Syntax und Semantik einer Programmiersprache

Zeitbedarf: 20 Stunden

Unterrichtsvorhaben Q1-IV

Thema:

Modellierung, Implementierung und Nutzung von relationalen Datenbanken in Anwendungskontexten

Zentrale Kompetenzen:

- Argumentieren
- Modellieren
- Implementieren
- Darstellen und Interpretieren
- Kommunizieren und Kooperieren

Inhaltsfelder:

- Daten und ihre Strukturierung
- Algorithmen
- Formale Sprachen und Automaten
- Informatik, Mensch und Gesellschaft

Inhaltliche Schwerpunkte:

- Datenbanken
- Algorithmen in ausgewählten informatischen Kontexten
- Syntax und Semantik einer Programmiersprache

Zeitbedarf: 20 Stunden

Qualifikationsphase 1

Unterrichtsvorhaben Q1-V

Thema:

Modellierung und Implementierung von dynamischen nicht-linearen Datenstrukturen und von Anwendungen mit dynamischen nicht-linearen Datenstrukturen in kontextbezogenen Problemstellungen.

Zentrale Kompetenzen:

- Argumentieren
- Modellieren
- Implementieren
- Darstellen und Interpretieren
- Kommunizieren und Kooperieren

Inhaltsfelder:

- Daten und ihre Strukturierung
- Algorithmen
- Formale Sprachen und Automaten

Inhaltliche Schwerpunkte:

- Objekte und Klassen
- Analyse, Entwurf und Implementierung von Algorithmen
- Algorithmen in ausgewählten informatischen Kontexten
- Syntax und Semantik einer Programmiersprache

Zeitbedarf: 40 Stunden

Unterrichtsvorhaben Q1-VI

Thema:

Projektorientierte Softwareentwicklung am Beispiel einer Anwendung mit Datenbankanbindung

Zentrale Kompetenzen:

- Argumentieren
- Modellieren
- Implementieren
- Darstellen und Interpretieren
- Kommunizieren und Kooperieren

Inhaltsfelder:

- Daten und ihre Strukturierung
- Algorithmen
- Formale Sprachen und Automaten
- Informatik, Mensch und Gesellschaft

Inhaltliche Schwerpunkte:

- Objekte und Klassen
- Analyse, Entwurf und Implementierung von Algorithmen
- Datenbanken
- Algorithmen in ausgewählten informatischen Kontexten
- Syntax und Semantik einer Programmiersprache
- Sicherheit

Zeitbedarf: 15 Stunden

Summe Qualifikationsphase 1: 135 Stunden

Qualifikationsphase 2

Unterrichtsvorhaben Q2-I

Thema:

Sicherheit und Datenschutz in Informatiksystemen sowie Grenzen und Auswirkungen der Automatisierung

Zentrale Kompetenzen:

- Argumentieren
- Darstellen und Interpretieren
- Kommunizieren und Kooperieren

Inhaltsfelder:

- Informatiksysteme
- Informatik, Mensch und Gesellschaft

Inhaltliche Schwerpunkte:

- Nutzung von Informatiksystemen
- Sicherheit
- Wirkungen der Automatisierung
- Grenzen der Automatisierung

Zeitbedarf: 20 Stunden

Unterrichtsvorhaben Q2-II

Thema:

Grundlagen der Netzwerkkommunikation sowie Modellierung und Implementierung von Client-Server-Anwendungen in kontextbezogenen Problemstellungen

Zentrale Kompetenzen:

- Argumentieren
- Modellieren
- Implementieren
- Darstellen und Interpretieren
- Kommunizieren und Kooperieren

Inhaltsfelder:

- Algorithmen
- Informatiksysteme

Inhaltliche Schwerpunkte:

- Algorithmen in ausgewählten Kontexten
- Einzelrechner und Rechnernetzwerke
- Nutzung von Informatiksystemen

Zeitbedarf: 20 Stunden

Qualifikationsphase 2

Unterrichtsvorhaben Q2-III

Thema:

Grundlagen von Automaten und formalen Sprachen sowie die Modellierung und Implementierung eines Parsers zu einer formalen Sprache

Zentrale Kompetenzen:

- Argumentieren
- Modellieren
- Implementieren
- Darstellen und Interpretieren
- Kommunizieren und Kooperieren

Inhaltsfelder:

- Formale Sprachen und Automaten

Inhaltliche Schwerpunkte:

- Endliche Automaten und Kellerautomaten
- Grammatiken regulärer und kontextfreier Sprachen
- Scanner, Parser und Interpreter für eine reguläre Sprache
- Möglichkeiten und Grenzen von Automaten und formalen Sprachen

Zeitbedarf: 30 Stunden

Unterrichtsvorhaben Q2-IV

Thema:

Prinzipielle Arbeitsweise eines Computers sowie Modellierung und Implementierung eines Scanners, Parsers und Interpreters für eine einfache maschinennahe Programmiersprache

Zentrale Kompetenzen:

- Argumentieren
- Modellieren
- Implementieren
- Kommunizieren und Kooperieren

Inhaltsfelder:

- Algorithmen
- Formale Sprachen und Automaten
- Informatiksysteme
- Informatik, Mensch und Gesellschaft

Inhaltliche Schwerpunkte:

- Analyse, Entwurf und Implementierung von Algorithmen
- Scanner, Parser und Interpreter für eine reguläre Sprache
- Einzelrechner und Rechnernetzwerke

Zeitbedarf: 12 Stunden

Qualifikationsphase 2

Unterrichtsvorhaben Q2-V

Thema:

Entwicklung eines Netzwerkspiels mit Durchführung eines vollständigen Softwareentwicklungszyklus

Zentrale Kompetenzen:

- Argumentieren
- Modellieren
- Implementieren
- Darstellen und Interpretieren
- Kommunizieren und Kooperieren

Inhaltsfelder:

- Daten und ihre Strukturierung
- Algorithmen
- Formale Sprachen und Automaten
- Informatiksysteme

Inhaltliche Schwerpunkte:

- Objekte und Klassen
- Analyse, Entwurf und Implementierung von Algorithmen
- Syntax und Semantik einer Programmiersprache
- Einzelrechner und Rechnernetzwerke
- Nutzung von Informatiksystemen

Zeitbedarf: 15 Stunden

Summe Qualifikationsphase 2: 97 Stunden

2.1.2 Konkretisierte Unterrichtsvorhaben

Im Folgenden sollen die im *Unterkapitel 2.1.1* aufgeführten Unterrichtsvorhaben konkretisiert werden.

Hinweis:

Verbindliche Festlegungen der Fachkonferenz:
Die Fachkonferenz der Beispielschule hat Themen, Leitfragen und die Ausführungen unter der Überschrift *Vorhabenbezogene Konkretisierung* verbindlich vereinbart, ebenso die Sequenzierung der Unterrichtsvorhaben (erste Tabellenspalte) und die ausgewiesenen Kompetenzen (zweite Tabellenspalte). Alle Mitglieder der Fachkonferenz haben sich darauf verständigt, in ihrem Unterricht Lerngelegenheiten anzubieten, so dass Schülerinnen und Schüler diese Kompetenzen im Rahmen der festgelegten Unterrichtssequenzen erwerben oder vertiefen können.

Unterrichtliche Anregungen:
Die angeführten Beispiele, Medien und Materialien sind dagegen Vorschläge bzw. Hilfen für die Lehrkräfte der Beispielschule. In diesen Bereichen sind Abweichungen von den vorgeschlagenen Vorgehensweisen möglich.

In der Einführungsphase wird die didaktische Bibliothek GLOOP verwendet. Die folgenden Installationspakete und Dokumentationen stehen zur Verfügung:

- Installationspaket (Windows) (Download)
- Installationspaket (MacOS) (Download)
- Installationspaket (Linux) (Download)
- Dokumentationen (Download)

In der Qualifikationsphase werden die Unterrichtsvorhaben unter Berücksichtigung der Vorgaben für das Zentralabitur Informatik in NRW konkretisiert. Diese sind zu beziehen unter der Adresse

https://www.standardsicherung.schulministerium.nrw.de/cms/zentralabitur-gost/faecher/fach.php?fach=15
(abgerufen: 01. 06. 2018)

I) Einführungsphase

Die folgenden Kompetenzen aus dem Bereich *Kommunizieren und Kooperieren* werden in allen Unterrichtsvorhaben der Einführungsphase vertieft und sollen aus Gründen der Lesbarkeit nicht in jedem Unterrichtsvorhaben separat aufgeführt werden:

Die Schülerinnen und Schüler

- verwenden Fachausdrücke bei der Kommunikation über informatische Sachverhalte (K),
- präsentieren Arbeitsabläufe und -ergebnisse (K),
- kommunizieren und kooperieren in Gruppen und in Partnerarbeit (K),
- nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung und gemeinsamen Verwendung von Daten unter Berücksichtigung der Rechteverwaltung (K).

Unterrichtsvorhaben EF-I

Thema: Einführung in die Nutzung von Informatiksystemen und in grundlegende Begrifflichkeiten

Leitfragen: *Womit beschäftigt sich die Wissenschaft der Informatik? Wie kann die in der Schule vorhandene informatische Ausstattung genutzt werden?*

Vorhabenbezogene Konkretisierung:
Das erste Unterrichtsvorhaben stellt eine allgemeine Einführung in das Fach Informatik dar. Dabei ist zu berücksichtigen, dass für manche Schülerinnen und Schüler in der Einführungsphase der erste Kontakt mit dem Unterrichtsfach Informatik stattfindet, so dass zu Beginn Grundlagen des Fachs behandelt werden müssen.

Zunächst wird auf den Begriff der Information eingegangen und die Möglichkeit der Kodierung in Form von Daten thematisiert. Anschließend wird auf die Übertragung von Daten im Sinne des Sender-Empfänger-Modells eingegangen. Dabei wird eine überblickartige Vorstellung der Kommunikation von Rechnern in Netzwerken erarbeitet.

Des Weiteren soll der grundlegende Aufbau eines Rechnersystems im Sinne der Von-Neumann-Architektur erarbeitet werden und mit dem grundlegenden Prinzip der Datenverarbeitung (Eingabe-Verarbeitung-Ausgabe) in Beziehung gesetzt werden.

Bei der Beschäftigung mit Datenkodierung, Datenübermittlung und Datenverarbeitung ist jeweils ein Bezug zur konkreten Nutzung der informatischen Ausstattung der Schule herzustellen. So wird in die verantwortungsvolle Nutzung dieser Systeme eingeführt.

Zeitbedarf: 6 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Information, deren Kodierung und Speicherung (a) Informatik als Wissenschaft der Verarbeitung von Informationen (b) Darstellung von Informationen in Schrift, Bild und Ton (c) Speichern von Daten mit informatischen Systemen am Beispiel der Schulrechner (d) Vereinbarung von Richtlinien zur Datenspeicherung auf den Schulrechnern (z.B. Ordnerstruktur, Dateibezeichner usw.)	Die Schülerinnen und Schüler • beschreiben und erläutern den Aufbau und die Arbeitsweise singulärer Rechner am Beispiel der „Von-Neumann-Architektur“ (A), • nutzen die im Unterricht eingesetzten Informatiksysteme selbstständig, sicher, zielführend und verantwortungsbewusst (D), • nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation (K).	*Beispiel*: Textkodierung Kodierung und Dekodierung von Texten mit unbekannten Zeichensätzen (z.B. Wingdings) *Beispiel*: Bildkodierung Kodierung von Bildinformationen in Raster- und Vektorgrafiken
2. Informations- und Datenübermittlung in Netzen (a) „Sender-Empfänger-Modell“ und seine Bedeutung für die Eindeutigkeit von Kommunikation (b) Informatische Kommunikation in Rechnernetzen am Beispiel des Schulnetzwerks (z.B. Benut-zeranmeldung, Netzwerkordner, Zugriffsrechte, Client-Server)		*Beispiel*: Rollenspiel zur Paketvermittlung im Internet Schülerinnen und Schüler übernehmen die Rollen von Clients und Routern. Sie schicken spielerisch Informationen auf Karten von einem Schüler-Client zum anderen. Jede Schülerin und jeder Schüler hat eine Adresse, jeder Router darüber hinaus eine Routingtabelle. Mit Hilfe der Tabelle und einem Würfel wird entschieden, wie ein Paket weiter vermittelt wird.

(c) Grundlagen der technischen Umsetzung von Rechnerkommunikation am Beispiel des Internets (z.B. Netzwerkadresse, Paketvermittlung, Protokoll) (d) Richtlinien zum verantwortungsvollen Umgang mit dem Internet		
3. Aufbau informatischer Systeme (a) Identifikation typischer Komponenten informatischer Systeme und anschließende Beschränkung auf das Wesentliche, Herleitung der „Von-Neumann-Architektur“ (b) Identifikation des EVA-Prinzips (Eingabe-Verarbeitung-Ausgabe) als Prinzip der Verarbeitung von Daten und Grundlage der „Von-Neumann-Architektur“		*Material*: Demonstrationshardware Durch Demontage eines Demonstrationsrechners entdecken Schülerinnen und Schüler die verschiedenen Hardwarekomponenten eines Informatiksystems. Als Demonstrationsrechner bietet sich ein ausrangierter Schulrechner an.

Unterrichtsvorhaben EF-II

Thema: Grundlagen der objektorientierten Analyse, Modellierung und Implementierung anhand von statischen Grafikszenen

Leitfrage: *Wie lassen sich Gegenstandsbereiche informatisch modellieren und im Sinne einer Simulation informatisch realisieren?*

Vorhabenbezogene Konkretisierung:
Ein zentraler Bestandteil des Informatikunterrichts der Einführungsphase ist die Objektorientierte Programmierung. Dieses Unterrichtsvorhaben führt in die Grundlagen der Analyse, Modellierung und Implementierung in diesem Kontext ein.

Dazu werden zunächst konkrete Gegenstandsbereiche aus der Lebenswelt der Schülerinnen und Schüler analysiert und im Sinne des Objektorientierten Paradigmas strukturiert. Dabei werden die grundlegenden Begriffe der Objektorientierung und Modellierungswerkzeuge wie Objektkarten, Klassenkarten oder Beziehungsdiagramme eingeführt.

Im Anschluss wird mit der Realisierung erster Projekte mit Hilfe der didaktischen Programmierumgebung GLOOP begonnen. Die von der Bibliothek vorgegebenen Klassen werden von Schülerinnen und Schülern in Teilen analysiert und entsprechende Objekte anhand einfacher Problemstellungen erprobt. Dazu muss der grundlegende Aufbau einer Java-Klasse thematisiert und zwischen Deklaration, Initialisierung und Methodenaufrufen unterschieden werden.

Da bei der Umsetzung dieser ersten Projekte konsequent auf die Verwendung von Kontrollstrukturen verzichtet wird und der Quellcode aus einer rein linearen Sequenz besteht, ist auf diese Weise eine Fokussierung auf die Grundlagen der Objektorientierung möglich, ohne dass algorithmische Probleme ablenken. Natürlich kann die Arbeit an diesen Projekten unmittelbar zum nächsten Unterrichtsvorhaben führen. Dort stehen unter anderem Kontrollstrukturen im Mittelpunkt.

Zeitbedarf: 8 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Identifikation von Objekten (a) Am Beispiel eines lebensweltnahen Beispiels werden Objekte im Sinne der Objektorientierten Modellierung eingeführt. (b) Objekte werden mit Objektkarten visualisiert und mit sinnvollen Attributen und „Fähigkeiten“, d.h. Methoden versehen. (c) Manche Objekte sind prinzipiell typgleich und werden so zu einer Objektsorte bzw. Objektklasse zusammengefasst.	Die Schülerinnen und Schüler • ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modellieren Klassen mit ihren Attributen, ihren Methoden und Assoziationsbeziehungen (M), • stellen die Kommunikation zwischen Objekten grafisch dar (M),	*Beispiel:* Vogelschwarm Schülerinnen und Schüler betrachten einen Vogelschwarm als Menge gleichartiger Objekte, die in einer Klasse mit Attributen und Methoden zusammengefasst werden können. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator - Allgemeine Objektorientierung (Download EF-II.1)

(d) Vertiefung: Modellierung weiterer Beispiele ähnlichen Musters	• implementieren einfache Algorithmen unter Beachtung der Syntax und Semantik einer Programmiersprache (I), • stellen den Zustand eines Objekts dar (D).	
2. Analyse von Klassen didaktischer Lernumgebungen (a) Objektorientierte Programmierung als modularisiertes Vorgehen (Entwicklung von Problemlösungen auf Grundlage vorhandener Klassen) (b) Teilanalyse der Klassen der didaktischen Lernumgebungen GLOOP		*Materialien*: Dokumentation der didaktischen Bibliothek GLOOP (Download EF-II.2)
3. Implementierung dreidimensionaler, statischer Szenen (a) Grundaufbau einer Java-Klasse (b) Konzeption einer Szene mit Kamera, Licht und sichtbaren Objekten (c) Deklaration und Initialisierung von Objekten (d) Methodenaufrufe mit Parameterübergabe zur Manipulation von Objekteigenschaften (z.B. Farbe, Position, Drehung)		*Beispiel:* Skulpturengarten Schülerinnen und Schüler erstellen ein Programm, das mit Hilfe von geometrischen Objekten der GLOOP-Umgebung einen Skulpturengaten auf den Bildschirm bringt. *Beispiel*: Olympische Ringe Die Schülerinnen und Schüler bilden das Emblem der olympischen Spiele mit Hilfe von GLOOP-Objekten nach. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator - Sequenzielle Programmierung (Download EF-II.3)

Unterrichtsvorhaben EF-III

Thema: Grundlagen der objektorientierten Programmierung und algorithmischer Grundstrukturen in Java anhand von einfachen Animationen

Leitfragen: *Wie lassen sich Animationen und Simulationen optischer Gegenstandsbereiche unter Berücksichtigung von Tastatureingaben realisieren?*

Vorhabenbezogene Konkretisierung:
Der Schwerpunkt dieses Unterrichtsvorhabens liegt auf der Entwicklung mehrerer Projekte, die durch Eingaben des Benutzers gesteuerte Animationen aufweisen. Zunächst wird ein Projekt bearbeitet, das in Anlehnung an das vorangegangene Unterrichtsvorhaben eine Szene darstellt, die lediglich aus Objekten besteht, zu denen das didaktische System Klassen vorgibt. Einzelne Objekte der Szene werden animiert, um ein einfaches Spiel zu realisieren oder die Szene optisch aufzuwerten. Für die Umsetzung dieses Projekts werden Kontrollstrukturen in Form von Schleifen und Verzweigungen benötigt und eingeführt.

Sind an einem solchen Beispiel im Schwerpunkt Schleifen und Verzweigungen eingeführt worden, sollen diese Konzepte an weiteren Beispielprojekten eingeübt werden. Dabei muss es sich nicht zwangsläufig um solche handeln, bei denen Kontrollstrukturen lediglich zur Animation verwendet werden. Auch die Erzeugung größerer Mengen grafischer Objekte und deren Verwaltung in einem Feld kann ein Anlass zur Verwendung von Kontrollstrukturen sein.

Das Unterrichtsvorhaben schließt mit einem Projekt, das komplexere grafische Elemente beinhaltet, so dass die Schülerinnen und Schüler mehr als nur die Klasse erstellen müssen, welche die Szene als Ganzes darstellt. Elemente der Szene müssen zu sinnhaften eigenen Klassen zusammengefasst werden, die dann ihre eigenen Attribute und Dienste besitzen. Auch dieses Projekt soll eine Animation, ggf. im Sinne einer Simulation, sein, bei der Attributwerte von Objekten eigener Klassen verändert werden und diese Veränderungen optisch sichtbar gemacht werden.

Komplexere Assoziationsbeziehungen zwischen Klassen werden in diesem Unterrichtsvorhaben zunächst nicht behandelt. Sie stellen den Schwerpunkt des folgenden Vorhabens dar.

Zeitbedarf: 18 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Bewegungsanimationen am Beispiel einfacher grafischer Objekte (GLObjekte) (a) Kontinuierliche Verschiebung eines GLObjekts mit Hilfe einer Schleife (While-Schleife) (b) Tastaturabfrage zur Realisierung einer Schleifenbedingung für eine Animationsschleife (c) Mehrstufige Animationen mit mehreren sequenziellen Schleifen (d) Berechnung von Abständen zwischen GLObjekten mit Hilfsvariablen (e) Meldungen zur Kollision zweier GLObjekte mit Hilfe von Abstandsberechnungen und Verzweigungen (IF-Anweisungen)	Die Schülerinnen und Schüler • analysieren und erläutern einfache Algorithmen und Programme (A), • entwerfen einfache Algorithmen und stellen sie umgangssprachlich und grafisch dar (M), • ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modellieren Klassen mit ihren Attributen, ihren Methoden und Assoziationsbeziehungen (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare Datensammlungen zu (M), • ordnen Klassen, Attributen und Methoden ihren Sichtbarkeitsbereich zu (M), • modifizieren einfache Algorithmen und Programme (I), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), • implementieren Algorithmen unter Verwendung von Variablen und Wertzuweisungen, Kontrollstrukturen sowie Methodenaufrufen (I),	*Beispiel*: Wurfspiel Die Schülerinnen und Schüler realisieren mit Objekten der GLOOP-Umgebung ein Spiel, bei dem ein Ball über den Bildschirm bewegt und auf eine runde Zielscheibe geworfen werden soll. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator – Kontrollstrukturen (Download EF-III.1)
2. Erstellen und Verwalten größerer Mengen einfacher grafischer Objekte (GLObjekte) (a) Erzeugung von Objekten mit Hilfe von Zählschleifen (FOR-Schleife) (b) Verwaltung von Objekten in eindimensionalen Feldern (Arrays) (c) Animation von Objekten, die in eindimensionalen Feldern (Arrays) verwaltet werden		*Beispiel:* Hubschrauberlandeplatz Die Schülerinnen und Schüler realisieren einen runden Hubschrauberlandeplatz und eine Reihe von Landemarkierungen, die in einem Feld verwaltet werden. Mit Hilfe der Landemarkierungen werden verschiedene Lauflichter realisiert. *Beispiel*: Schachbrett Die Schülerinnen und Schüler realisieren mit Hilfe mehrerer Quader ein Schachbrett. *Beispiel*: Magischer Würfel

(d) Vertiefung: Verschiedene Feldbeispiele	• implementieren einfache Algorithmen unter Beachtung der Syntax und Semantik einer Programmiersprache (I), • testen Programme schrittweise anhand von Beispielen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I).	Die Schülerinnen und Schüler erstellen einen großen Würfel, der aus mehreren kleineren, farbigen Würfeln besteht. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator - Kontrollstrukturen (Download EF-III.2)
3. Modellierung und Animation komplexerer grafisch repräsentierbarer Objekte (a) Modellierung eines Simulationsprogramms mit eigenen Klassen, die sich selbst mit Hilfe von einfachen GLObjekten zeigen mit Hilfe eines Implementationsdiagramms (b) Implementierung eigener Methoden mit und ohne Parameterübergabe (c) Realisierung von Zustandsvariablen (d) Thematisierung des Geheimnisprinzips und des Autonomitätsprinzips von Objekten (e) Animation mit Hilfe des Aufrufs von selbstimplementierten Methoden (f) Vertiefung: Weitere Projekte		*Beispiel: Kerzensimulation* Die Schülerinnen und Schüler modellieren und erstellen eine Klasse, mit deren Hilfe Kerzen simuliert werden können. Eine Kerze kann angezündet und gelöscht werden. Abgesehen davon brennen Kerzen abhängig von ihrer Dicke unterschiedlich schnell ab. *Beispiel*: Uhren Die Schülerinnen und Schüler erstellen eine Simulation mehrerer Uhren für verschiedene Zeitzonen. *Beispiel:* Ampeln Die Schülerinnen und Schüler erstellen eine Ampelkreuzung mit mehreren Ampelanlagen an einem Bahnübergang. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator – Eigene Klassen (Download EF-III.3)

Unterrichtsvorhaben EF-IV

Thema: Modellierung und Implementierung von Klassen- und Objektbeziehungen anhand von grafischen Spielen und Simulationen

Leitfrage: *Wie lassen sich komplexere Datenflüsse und Beziehungen zwischen Objekten und Klassen realisieren?*

Vorhabenbezogene Konkretisierung:
Dieses Unterrichtsvorhaben beschäftigt sich im Schwerpunkt mit dem Aufbau komplexerer Objektbeziehungen. Während in vorangegangenen Unterrichtsvorhaben Objekte nur jeweils solchen Objekten Nachrichten schicken konnten, die sie selbst erstellt haben, soll in diesem Unterrichtsvorhaben diese hierarchische Struktur aufgebrochen werden.

Dazu bedarf es zunächst einer präzisen Unterscheidung zwischen Objektreferenzen und Objekten, so dass klar wird, dass Dienste eines Objektes von unterschiedlichen Objekten über unterschiedliche Referenzen in Anspruch genommen werden können. Auch der Aufbau solcher Objektbeziehungen muss thematisiert werden. Des Weiteren wird das Prinzip der Vererbung im objektorientierten Sinne angesprochen. Dazu werden die wichtigsten Varianten der Vererbung anhand von verschiedenen Projekten vorgestellt. Zunächst wird die Vererbung als Spezialisierung im Sinne einer einfachen Erweiterung einer Oberklasse vorgestellt. Darauf folgt ein Projekt, welches das Verständnis von Vererbung um den Aspekt der späten Bindung erweitert, indem Dienste einer Oberklasse überschrieben werden. Modellierungen sollen in Form von Implementationsdiagrammen erstellt werden.

Zum Abschluss kann kurz auf das Prinzip der abstrakten Klasse eingegangen werden. Dieser Inhalt ist aber nicht obligatorisch für die Einführungsphase.

Zeitbedarf: 18 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Vertiefung des Referenzbegriffs und Einführung des Prinzips der dynamischen Referenzierung (a) Einführung der GLOOP-Objektselektion mit der Maus	Die Schülerinnen und Schüler • analysieren und erläutern eine objektorientierte Modellierung (A),	*Beispiel*: Seifenblasen Die Schülerinnen und Schüler entwickeln ein Spiel, bei dem Seifenblasen über den Bildschirm schweben und durch anklicken mit der Maus zum Zerplatzen gebracht werden können.

(b) Einführung der Klasse `GLObjekt` als Oberklasse aller sichtbaren Objekte in GLOOP (c) Steuerung einfacher grafischer Objekte über eine Referenz `aktuell`, die jeweils durch eine Klickselektion mit der Maus auf ein neues Objekt gesetzt werden kann.	• stellen die Kommunikation zwischen Objekten grafisch dar (M), • ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modellieren Klassen mit ihren Attributen, ihren Methoden und Assoziationsbeziehungen (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare Datensammlungen zu (M), • ordnen Klassen, Attributen und Methoden ihren Sichtbarkeitsbereich zu (M), • modellieren Klassen unter Verwendung von Vererbung (M), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), • testen Programme schrittweise anhand von Beispielen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • modifizieren einfache Algorithmen und Programme (I), • stellen Klassen, Assoziations- und Vererbungsbeziehungen in Diagrammen grafisch dar (D),	*Beispiel*: Sonnensystem Die Schülerinnen und Schüler entwickeln eine Simulation des Sonnensystems bei der Daten zum angeklickten Planeten ausgegeben werden. *Materialien:* -
2. Entwicklung eines Spiels mit der Notwendigkeit von Kollisionskontrollen zwischen zwei oder mehr grafischen Objekten (a) Modellierung des Spiels ohne Berücksichtigung der Kollision mit Hilfe eines Implementationsdiagramms (b) Dokumentation der Klassen des Projekts (c) Implementierung eines Prototypen ohne Kollision (d) Ergänzung einer Kollisionsabfrage durch zusätzliche Assoziationsbeziehungen in Diagramm, Dokumentation und Quellcode (e) Verallgemeinerung der neuen Verwendung von Objektreferenzen (f) Vertiefung: Entwicklung weiterer Spiele und Simulationen mit vergleichbarer Grundmodellierung		*Beispiel:* Ufospiel Die Schülerinnen und Schüler entwickeln die Simulation eines Ufos, das Asteroiden ausweichen soll mit denen eine Kollision möglich ist. *Beispiel*: Billardkugeln Die Schülerinnen und Schüler entwickeln ein Spiel, bei dem tickende Billardkugeln mit einer beweglichen Box eingefangen werden sollen. *Beispiel*: Autospiel Die Schülerinnen und Schüler entwickeln ein Autospiel, bei dem ein Auto durch einen Wald fahren und mit Bäumen kollidieren kann. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator – Assoziationen (Download EF-IV.1) Informationsblatt: Implementationsdiagramme (Download EF-IV.2)
3. Erarbeitung einer Simulation mit grafischen Objekten, die sich durch		*Beispiel*: Schneemann

unterschiedliche Ergänzungen voneinander unterscheiden (Vererbung durch Spezialisierung ohne Überschreiben von Methoden) (a) Analyse und Erläuterung einer Basisversion der grafischen Klasse (b) Realisierung von grafischen Erweiterungen zur Basisklasse mit und ohne Vererbung (Implementationsdiagramm und Quellcode) (c) Verallgemeinerung und Reflexion des Prinzips der Vererbung am Beispiel der Spezialisierung	• dokumentieren Klassen durch Beschreibung der Funktionalität der Methoden (D).	Die Schülerinnen und Schüler erstellen eine Simulation von Schneemännern, die unterschiedliche Kopfbedeckungen tragen. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator – Vererbung (Download EF-IV.3)
4. Entwicklung einer komplexeren Simulation mit grafischen Elementen, die unterschiedliche Animationen durchführen (Vererbung mit Überschreiben von Methoden) (a) Analyse und Erläuterung einer einfachen grafischen Animationsklasse (b) Spezialisierung der Klasse zu Unterklassen mit verschiedenen Animationen durch Überschreiben der entsprechenden Animationsmethode (c) Reflexion des Prinzips der späten Bindung (d) Vertiefung: Entwicklung eines vergleichbaren Projekts mit einer (abstrakten) Oberklasse		*Beispiel*: Flummibälle Die Schülerinnen und Schüler entwickeln eine Simulation von Flummibällen, bei der unterschiedliche Bälle unterschiedliche Bewegungen durchführen. *Beispiel*: Weihnachtsbaum Die Schülerinnen und Schüler entwickeln eine Simulation eines Weihnachtsbaums mit Hilfe einer abstrakten Klasse `Schmuck`. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator – Vererbung (Download EF-IV.4)

Unterrichtsvorhaben EF-V

Thema: Such- und Sortieralgorithmen anhand kontextbezogener Beispiele

Leitfragen: *Wie können Objekte bzw. Daten effizient sortiert werden, so dass eine schnelle Suche möglich wird?*

Vorhabenbezogene Konkretisierung:
Dieses Unterrichtsvorhaben beschäftigt sich mit der Erarbeitung von Such- und Sortieralgorithmen. Der Schwerpunkt des Vorhabens liegt dabei auf den Algorithmen selbst und nicht auf deren Implementierung in einer Programmiersprache, auf die in diesem Vorhaben vollständig verzichtet werden soll.

Zunächst erarbeiten die Schülerinnen und Schüler mögliche Einsatzszenarien für Such- und Sortieralgorithmen, um sich der Bedeutung einer effizienten Lösung dieser Probleme bewusst zu werden. Anschließend werden Strategien zur Sortierung mit Hilfe eines explorativen Spiels von den Schülerinnen und Schülern selbst erarbeitet und hinsichtlich der Anzahl notwendiger Vergleiche auf ihre Effizienz untersucht.

Daran anschließend werden die erarbeiteten Strategien systematisiert und im Pseudocode notiert. Die Schülerinnen und Schüler sollen auf diese Weise das *Sortieren durch Vertauschen*, das *Sortieren durch Auswählen* und mindestens einen weiteren Sortieralgorithmus, kennen lernen.

Des Weiteren soll das Prinzip der *binären Suche* behandelt und nach Effizienzgesichtspunkten untersucht werden.

Zeitbedarf: 9 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Explorative Erarbeitung eines Sortierverfahrens (a) Sortierprobleme im Kontext informatischer Systeme und im Alltag (z.B. Dateisortierung, Tabellenkalkulation, Telefonbuch, Bundesligatabelle, usw.)	Die Schülerinnen und Schüler • beurteilen die Effizienz von Algorithmen am Beispiel von Sortierverfahren hinsichtlich Zeit und Speicherplatzbedarf (A),	*Beispiel:* Sortieren mit Waage Die Schülerinnen und Schüler bekommen die Aufgabe, kleine, optisch identische Kunststoffbehälter aufsteigend nach ihrem Gewicht zu sortieren. Dazu steht ihnen eine Balkenwaage zur Verfügung, mit deren Hilfe sie das Gewicht zweier Behälter vergleichen können.

(b) Vergleich zweier Elemente als Grundlage eines Sortieralgorithmus (c) Erarbeitung eines Sortieralgorithmus durch die Schülerinnen und Schüler	• entwerfen einen weiteren Algorithmus zum Sortieren (M), • analysieren Such- und Sortieralgorithmen und wenden sie auf Beispiele an (D).	*Materialien:* Computer science unplugged – Sorting Algorithms, URL: www.csunplugged.org/sorting-algorithms abgerufen: 30. 03. 2014
2. Systematisierung von Algorithmen und Effizienzbetrachtungen (a) Formulierung (falls selbst gefunden) oder Erläuterung von mehreren Algorithmen im Pseudocode (auf jeden Fall: Sortieren durch Vertauschen, Sortieren durch Auswählen) (b) Anwendung von Sortieralgorithmen auf verschiedene Beispiele (c) Bewertung von Algorithmen anhand der Anzahl der nötigen Vergleiche (d) Variante des Sortierens durch Auswählen (Nutzung eines einzigen oder zweier Felder bzw. lediglich eines einzigen zusätzlichen Ablageplatzes oder mehrerer neuer Ablageplätze) (e) Effizienzbetrachtungen an einem konkreten Beispiel bezüglich der Rechenzeit und des Speicherplatzbedarfs (f) Analyse des weiteren Sortieralgorithmus (sofern nicht in Sequenz 1 und 2 bereits geschehen)		*Beispiele*: Sortieren durch Auswählen, Sortieren durch Vertauschen, Quicksort Quicksort ist als Beispiel für einen Algorithmus nach dem Prinzip *Teile und Herrsche* gut zu behandeln. Kenntnisse in rekursiver Programmierung sind nicht erforderlich, da eine Implementierung nicht angestrebt wird. *Materialien:* Computer science unplugged – Sorting Algorithms, URL: www.csunplugged.org/sorting-algorithms abgerufen: 30. 03. 2014
3. Binäre Suche auf sortierten Daten		

(a) Suchaufgaben im Alltag und im Kontext informatischer Systeme (b) Evtl. Simulationsspiel zum effizienten Suchen mit binärer Suche (c) Effizienzbetrachtungen zur binären Suche		*Beispiel:* Simulationsspiel zur binären Suche nach Tischtennisbällen Mehrere Tischtennisbälle sind nummeriert, sortiert und unter Bechern verdeckt. Mit Hilfe der binären Suche kann sehr schnell ein bestimmter Tischtennisball gefunden werden. *Materialien:* Computer science unplugged – Searching Algorithms, URL: www.csunplugged.org/searching-algorithms, abgerufen: 30. 03. 2014

Unterrichtsvorhaben EF-VI

Thema: Geschichte der digitalen Datenverarbeitung und die Grundlagen des Datenschutzes

Leitfrage: *Welche Entwicklung durchlief die moderne Datenverarbeitung und welche Auswirkungen ergeben sich insbesondere hinsichtlich neuer Anforderungen an den Datenschutz daraus?*

Vorhabenbezogene Konkretisierung:
Das folgende Unterrichtsvorhaben stellt den Abschluss der Einführungsphase dar. Schülerinnen und Schüler sollen selbstständig informatische Themenbereiche aus dem Kontext der Geschichte der Datenverarbeitung und insbesondere den daraus sich ergebenen Fragen des Datenschutzes bearbeiten. Diese Themenbereiche werden in Kleingruppen bearbeitet und in Form von Plakatpräsentationen vorgestellt. Schülerinnen und Schüler sollen dabei mit Unterstützung des Lehrenden selbstständige Recherchen zu ihren Themen anstellen und auch eine sinnvolle Eingrenzung ihres Themas vornehmen.

Anschließend wird verstärkt auf den Aspekt des Datenschutzes eingegangen. Dazu wird das Bundesdatenschutzgesetz in Auszügen behandelt und auf schülernahe Beispielsituationen zur Anwendung gebracht. Dabei steht keine formale juristische Bewertung der Beispielsituationen im Vordergrund, die im Rahmen eines Informatikunterrichts auch nicht geleistet werden kann, sondern vielmehr eine persönliche Einschätzung von Fällen im Geiste des Datenschutzgesetzes.

Zeitbedarf: 15 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Selbstständige Erarbeitung von Themen durch die Schülerinnen und Schüler (a) Mögliche Themen zur Erarbeitung in Kleingruppen: • „Eine kleine Geschichte der Digitalisierung: vom Morsen zum modernen Digitalcomputer“ • „Eine kleine Geschichte der Kryptographie: von Caesar zur Enigma“ • „Von Nullen, Einsen und mehr: Stellenwertsysteme und wie man mit ihnen rechnet“ • „Kodieren von Texten und Bildern: ASCII, RGB und mehr“ • „Auswirkungen der Digitalisierung: Veränderungen der Arbeitswelt und Datenschutz“ (b) Vorstellung und Diskussion durch Schülerinnen und Schüler	Die Schülerinnen und Schüler • bewerten anhand von Fallbeispielen die Auswirkungen des Einsatzes von Informatiksystemen (A), • erläutern wesentliche Grundlagen der Geschichte der digitalen Datenverarbeitung (A), • stellen ganze Zahlen und Zeichen in Binärcodes dar (D), • interpretieren Binärcodes als Zahlen und Zeichen (D), • nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation. (K).	*Beispiel:* Ausstellung zu informatischen Themen Die Schülerinnen und Schüler bereiten eine Ausstellung zu informatischen Themen vor. Dazu werden Stellwände und Plakate vorbereitet, die ggf. auch außerhalb des Informatikunterrichts in der Schule ausgestellt werden können. *Materialien:* Schülerinnen und Schüler recherchieren selbstständig im Internet, in der Schulbibliothek, in öffentlichen Bibliotheken, usw.
2. Vertiefung des Themas Datenschutz (a) Erarbeitung grundlegender Begriffe des Datenschutzen (b) Problematisierung und Anknüpfung an die Lebenswelt der Schülerinnen und Schüler		*Beispiel:* Fallbeispiele aus dem aktuellen Tagesgeschehen Die Schülerinnen und Schüler bearbeiten Fallbeispiele aus ihrer eigenen Erfahrungswelt oder der aktuellen Medienberichterstattung. *Materialien:*

(c) Diskussion und Bewertung von Fallbeispielen aus dem Themenbereich „Datenschutz“		Materialblatt zum Bundesdatenschutzgesetz (Download EF-VI.1)

II) Qualifikationsphase - GRUNDKURS

Die folgenden Kompetenzen aus dem Bereich *Kommunizieren und Kooperieren* werden in allen Unterrichtsvorhaben der Qualifikationsphase vertieft und sollen aus Gründen der Lesbarkeit nicht in jedem Unterrichtsvorhaben separat aufgeführt werden:

Die Schülerinnen und Schüler

- verwenden die Fachsprache bei der Kommunikation über informatische Sachverhalte (K),
- nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Dateien unter Berücksichtigung der Rechteverwaltung (K),
- organisieren und koordinieren kooperatives und eigenverantwortliches Arbeiten (K),
- strukturieren den Arbeitsprozess, vereinbaren Schnittstellen und führen Ergebnisse zusammen (K),
- beurteilen Arbeitsorganisation, Arbeitsabläufe und Ergebnisse (K),
- präsentieren Arbeitsabläufe und -ergebnisse adressatengerecht (K).

Unterrichtsvorhaben Q1-I:

Thema: Wiederholung der objektorientierten Modellierung und Programmierung

Leitfragen: *Wie modelliert und implementiert man zu einer Problemstellung in einem geeigneten Anwendungskontext Java-Klassen inklusive ihrer Attribute, Methoden und Beziehungen? Wie kann man die Modellierung und die Funktionsweise der Anwendung grafisch darstellen?*

Vorhabenbezogenen Konkretisierung:
Zu einer Problemstellung in einem Anwendungskontext soll eine Java-Anwendung entwickelt werden. Die Problemstellung soll so gewählt sein, dass für diese Anwendung die Verwendung einer abstrakten Oberklasse als Generalisierung verschiedener Unterklassen sinnvoll erscheint und eine Klasse durch eine Unterklasse spezialisiert werden kann. Um die Aufgabe einzugrenzen, können (nach der ersten Problemanalyse) einige Teile (Modellierungen oder Teile von Java-Klassen) vorgegeben werden.

Die Schülerinnen und Schülern erläutern und modifizieren den ersten Entwurf und modellieren sowie implementieren weitere Klassen und Methoden für eine entsprechende Anwendung. Klassen und ihre Beziehungen werden in einem Implementationsdiagramm dargestellt. Dabei werden Sichtbarkeitsbereiche zugeordnet. Exemplarisch wird eine Klasse dokumentiert. Der Nachrichtenaustausch zwischen verschiedenen Objekten wird verdeutlicht, indem die Kommunikation zwischen zwei ausgewählten Objekten grafisch dargestellt wird. In diesem Zusammenhang wird das Nachrichtenkonzept der objektorientierten Programmierung wiederholt.

Zeitbedarf: 8 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Wiederholung und Erweiterung der objektorientierten Modellierung und Programmierung durch Analyse und Erweiterung eines kontextbezogenen Beispiels (a) Analyse der Problemstellung (b) Analyse der Modellierung (Implementationsdiagramm) (c) Erweiterung der Modellierung im Implementationsdiagramm (Vererbung, abstrakte Klasse) (d) Kommunikation zwischen mindestens zwei Objekten (grafische Darstellung) (e) Dokumentation von Klassen (f) Implementierung der Anwendung oder von Teilen der Anwendung	Die Schülerinnen und Schüler • analysieren und erläutern objektorientierte Modellierungen (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M), • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I),	*Beispiel:* Wetthuepfen Für ein Wetthüpfen zwischen einem Hasen, einem Hund und einem Vogel werden die Tiere gezeichnet. Alle Tiere springen wiederholt nach links. Die Höhe und Weite jedes Hüpfers ist zufällig. Evtl. marschieren sie anschließend hintereinander her. oder *Beispiel*: Tannenbaum Ein Tannenbaum soll mit verschiedenen Arten von Schmuckstücken versehen werden, die durch unterschiedliche geometrische Objekte dargestellt werden. Es gibt Kugeln, Päckchen in der Form von Würfeln und Zuckerringe in Form von Toren. Ein Prototyp, der bereits mit Kugeln geschmückt werden kann, kann zur Verfügung gestellt werden. Da alle Schmuckstücke über die Funktion des Auf- und Abschmückens verfügen sollen, liegt es nahe, dass entsprechende Methoden in einer gemeinsamen Oberklasse realisiert werden. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.1-Wiederholung (Download Q1-I.1)

	• wenden eine didaktisch orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen (D), • stellen die Kommunikation zwischen Objekten grafisch dar (D).	

Unterrichtsvorhaben Q1-II:

Thema: Modellierung und Implementierung von Anwendungen mit dynamischen, linearen Datenstrukturen

Leitfrage: *Wie können beliebig viele linear angeordnete Daten im Anwendungskontext verwaltet werden?*

Vorhabenbezogene Konkretisierung:
Nach Analyse einer Problemstellung in einem geeigneten Anwendungskontext, in dem Daten nach dem First-In-First-Out-Prinzip verwaltet werden, werden der Aufbau von Schlangen am Beispiel dargestellt und die Operationen der Klasse `Queue` erläutert. Anschließend werden für die Anwendung notwendige Klassen modelliert und implementiert. Eine Klasse für eine den Anforderungen der Anwendung entsprechende Oberfläche sowie die Klasse `Queue` wird dabei von der Lehrkraft vorgegeben. Anschließend wird die Anwendung modifiziert, um den Umgang mit der Datenstruktur zu üben. Anhand einer Anwendung, in der Daten nach dem Last-In-First-Out-Prinzip verwaltet werden, werden Unterschiede zwischen den Datenstrukturen Schlange und Stapel erarbeitet. Um einfacher an Objekte zu gelangen, die zwischen anderen gespeichert sind, wird die Klasse `List` eingeführt und in einem Anwendungskontext verwendet. In mindestens einem weiteren Anwendungskontext wird die Verwaltung von Daten in Schlangen, Stapeln oder Listen vertieft. Modellierungen werden dabei in Entwurfs- und Implementationsdiagrammen dargestellt.

Zeitbedarf: 20 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Die Datenstruktur Schlange im Anwendungskontext unter Nutzung der Klasse `Queue` (a) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen (b) Erarbeitung der Funktionalität der Klasse `Queue` (c) Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse `Queue`	Die Schülerinnen und Schüler • erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), • analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M), • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modifizieren Algorithmen und Programme (I), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I),	*Beispiel:* Patientenwarteschlange (jeder kennt seinen Nachfolger bzw. alternativ: seinen Vorgänger) Sobald ein Patient in einer Arztpraxis eintrifft, werden sein Name und seine Krankenkasse erfasst. Die Verwaltung der Patientenwarteschlange geschieht über eine Klasse, die hier als Wartezimmer bezeichnet wird. Wesentliche Operationen sind das „Hinzufügen“ eines Patienten und das „Entfernen“ eines Patienten, wenn er zur Behandlung gerufen wird. Die Simulationsanwendung stellt eine GUI zur Verfügung, legt ein Wartezimmer an und steuert die Abläufe. Wesentlicher Aspekt des Projektes ist die Modellierung des Wartezimmers mit Hilfe der Klasse Queue. Anschließend wird der Funktionsumfang der Anwendung erweitert: Patienten können sich zusätzlich in die Warteschlange zum Blutdruckmessen einreihen. Objekte werden von zwei Schlangen verwaltet. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.2 – Warteschlange (Download Q1-II.1)

2. Die Datenstruktur Stapel im Anwendungskontext unter Nutzung der Klasse `Stack` (a) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen (b) Erarbeitung der Funktionalität der Klasse `Stack` (c) Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse `Stack`	• interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme systematisch anhand von Beispielen (I), • stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D).	*Beispiel:* Heftstapel In einem Heftstapel soll das Heft einer Schülerin gefunden werden. oder *Beispiel*: Kisten stapeln In einem Stapel nummerierter Kisten soll eine bestimmte Kiste gefunden und an einen Kunden geliefert werden. Dazu müssen Kisten auf verschiedene Stapel gestapelt und wieder zurückgestellt werden.
3. Die Datenstruktur lineare Liste im Anwendungskontext unter Nutzung der Klasse `List` (a) Erarbeitung der Vorteile der Klasse `List` im Gegensatz zu den bereits bekannten linearen Strukturen (b) Modellierung und Implementierung einer kontextbezogenen Anwendung unter Verwendung der Klasse `List`.		*Beispiel*: Abfahrtslauf Bei einem Abfahrtslauf kommen die Skifahrer nacheinander an und werden nach ihrer Zeit in eine Rangliste eingeordnet. Diese Rangliste wird in einer Anzeige ausgegeben. Ankommende Abfahrer müssen an jeder Stelle der Struktur, nicht nur am Ende oder Anfang eingefügt werden können. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.2 - Listen (Download Q1-II.2)
4. Vertiefung - Anwendungen von Listen, Stapeln oder Schlangen in mindestens einem weiteren Kontext		*Beispiel:* Skispringen Ein Skispringen hat folgenden Ablauf: Nach dem Sprung erhält der Springer eine Punktzahl und wird nach dieser Punktzahl in eine Rangliste eingeordnet. Die besten 30 Springer qualifizieren sich für den zweiten Durchgang. Sie starten in umgekehrter Reihenfolge gegenüber der Platzierung

		auf der Rangliste. Nach dem Sprung erhält der Springer wiederum eine Punktzahl und wird nach der Gesamtpunktzahl aus beiden Durchgängen in die endgültige Rangliste eingeordnet. *Beispiel*: Terme in Postfix-Notation Die sog. UPN (*Umgekehrt-Polnische-Notation*) bzw. *Postfix-Notation* eines Terms setzt den Operator hinter die Operanden. Um einen Term aus der gewohnten Infixschreibweise in einen Term in UPN umzuwandeln oder um den Wert des Terms zu berechnen, kann ein Stack verwendet werden. *Beispiel*: Rangierbahnhof Auf einem Güterbahnhof gibt es drei Gleise, die nur zu einer Seite offen sind. Wagons können also von einer Seite auf das Gleis fahren und nur rückwärts wieder hinausfahren. Die Wagons tragen Nummern, wobei die Nummer jedoch erst eingesehen werden kann, wenn der Wagon der vorderste an der offenen Gleisseite ist. (Zwischen den Wagons herumzuturnen, um die anderen Wagonnummern zu lesen, wäre zu gefährlich.) Zunächst stehen alle Wagons unsortiert auf einem Gleis. Ziel ist es, alle Wagons in ein anderes Gleis zu fahren, so dass dort die Nummern der Wagons vom Gleisende aus aufsteigend in richtiger Reihenfolge sind. Zusätzlich zu diesen beiden Gleisen gibt es ein Abstellgleis, das zum Rangieren benutzt werden kann. *Beispiel*: Autos an einer Ampel zur Zufahrtsregelung Es soll eine Ampel zur Zufahrtsregelung in Java simuliert werden. An einem geradlinigen, senkrecht von unten nach oben verlaufenden Straßenstück, das von Autos nur einspurig in eine

		Richtung befahren werden kann, ist ein Haltepunkt markiert, an dem die Ampel steht. Bei einem Klick auf eine Schaltfläche mit der Aufschrift „Heranfahren“ soll ein neues Auto an den Haltepunkt heranfahren bzw. bis an das letzte Auto, das vor dem Haltepunkt wartet. Grünphasen der Ampel werden durch einen Klick auf eine Schaltfläche mit der Aufschrift „Weiterfahren“ simuliert. In jeder Grünphase darf jeweils nur ein Auto weiterfahren. Die anderen Autos rücken nach. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1-II.3 – Anwendungen für lineare Datenstrukturen (Download Q1-II.3)

Unterrichtsvorhaben Q1-III:

Thema: Suchen und Sortieren auf linearen Datenstrukturen

Leitfrage: *Wie kann man gespeicherte Informationen günstig (wieder-)finden?*

Vorhabenbezogene Konkretisierung:
In einem Anwendungskontext werden zunächst Informationen in einer linearen Liste bzw. einem Feld gesucht. Hierzu werden Verfahren entwickelt und implementiert bzw. analysiert und erläutert, wobei neben einem iterativen auch ein rekursives Verfahren thematisiert wird und mindestens ein Verfahren selbst entwickelt und implementiert wird. Die verschiedenen Verfahren werden hinsichtlich Speicherbedarf und Zahl der Vergleichsoperationen miteinander verglichen.

Anschließend werden Sortierverfahren entwickelt und implementiert (ebenfalls für lineare Listen und Felder). Hierbei soll auch ein rekursives Sortierverfahren entwickelt werden. Die Implementationen von Quicksort sowie dem Sortieren durch Einfügen werden analysiert und erläutert. Falls diese Verfahren vorher schon entdeckt wurden, sollen sie hier wiedererkannt werden. Die rekursive Abarbeitung eines Methodenaufrufs von Quicksort wird grafisch dargestellt.

Abschließend werden verschiedene Sortierverfahren hinsichtlich der Anzahl der benötigten Vergleichsoperationen und des Speicherbedarfs beurteilt.

Zeitbedarf: 16 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Suchen von Daten in Listen und Arrays (a) Lineare Suche in Listen und in Arrays (b) Binäre Suche in Arrays als Beispiel für rekursives Problemlösen (c) Untersuchung der beiden Suchverfahren hinsichtlich ihrer Effizienz (Laufzeitverhalten, Speicherbedarf)	Die Schülerinnen und Schüler • analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A), • entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“ und „Teilen und Herrschen“ (M), • modifizieren Algorithmen und Programme (I), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • implementieren und erläutern iterative und rekursive Such- und Sortierverfahren (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der	*Beispiel:* Karteiverwaltung Für ein Adressverwaltungsprogramm soll eine Methode zum Suchen einer Adresse geschrieben werden. oder *Beispiel*: Bundesjugendspiele Die Teilnehmer an Bundesjugendspielen nehmen an drei Disziplinen teil und erreichen dort Punktzahlen. Diese werden in einer Wettkampfkarte eingetragen und an das Wettkampfbüro gegeben. Zur Vereinfachung sollte sich das Modell auf die drei Disziplinen „Lauf”, „Sprung“ und „Wurf“ beschränken. Im Wettkampfbüro wird das Ergebnis erstellt. Das Programm soll dafür zunächst den Besten einer Disziplin heraussuchen können und später das gesamte Ergebnis nach gewissen Kriterien sortieren können. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.3 - Suchen und Sortieren (Download Q1-III.1)

2. Sortieren in Listen und Arrays - Entwicklung und Implementierung von iterativen und rekursiven Sortierverfahren (a) Entwicklung und Implementierung eines einfachen Sortierverfahrens für eine Liste (b) Implementierung eines einfachen Sortierverfahrens für ein Feld (c) Entwicklung eines rekursiven Sortierverfahren für ein Feld (z.B. Sortieren durch Mischen)	Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme systematisch anhand von Beispielen (I), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D).	*Beispiel:* Karteiverwaltung (s.o.) oder *Beispiel*: Bundesjugendspiele (s.o.) *Materialien:* (s.o.)
3. Untersuchung der Effizienz der Sortierverfahren „Sortieren durch direktes Einfügen“ und „Quicksort“ auf linearen Listen (a) Grafische Veranschaulichung der Sortierverfahren (b) Untersuchung der Anzahl der Vergleichsoperationen und des Speicherbedarf bei beiden Sortierverfahren (c) Beurteilung der Effizienz der beiden Sortierverfahren		*Beispiel:* Karteiverwaltung (s.o.) oder *Beispiel*: Bundesjugendspiele (s.o.) *Materialien:* (s.o.)

Unterrichtsvorhaben Q1-IV:

Thema: Modellierung und Nutzung von relationalen Datenbanken in Anwendungskontexten

Leitfragen*: Wie können Fragestellungen mit Hilfe einer Datenbank beantwortet werden? Wie entwickelt man selbst eine Datenbank für einen Anwendungskontext?*

Vorhabenbezogene Konkretisierung:
Ausgehend von einer vorhandenen Datenbank entwickeln Schülerinnen und Schüler für sie relevante Fragestellungen, die mit dem vorhandenen Datenbestand beantwortet werden sollen. Zur Beantwortung dieser Fragestellungen wird die vorgegebene Datenbank von den Schülerinnen und Schülern analysiert und die notwendigen Grundbegriffe für Datenbanksysteme sowie die erforderlichen SQL-Abfragen werden erarbeitet.

In anderen Anwendungskontexten müssen Datenbanken erst noch entwickelt werden, um Daten zu speichern und Informationen für die Beantwortung von möglicherweise auftretenden Fragen zur Verfügung zu stellen. Dafür ermitteln Schülerinnen und Schüler in den Anwendungssituationen Entitäten, zugehörige Attribute, Relationen und Kardinalitäten und stellen diese in Entity-Relationship-Modellen dar. Entity-Relationship-Modelle werden interpretiert und erläutert, modifiziert und in Datenbankschemata überführt. Mit Hilfe von SQL-Anweisungen können anschließend im Kontext relevante Informationen aus der Datenbank extrahiert werden.

Ein Entity-Relationship-Diagramm kann auch verwendet werden, um die Entitäten inklusive ihrer Attribute und Relationen in einem vorgegebenen Datenbankschema darzustellen.

An einem Beispiel wird verdeutlicht, dass in Datenbanken Redundanzen unerwünscht sind und Konsistenz gewährleistet sein sollte. Die 1. bis 3. Normalform wird als Gütekriterium für Datenbankentwürfe eingeführt. Datenbankschemata werden hinsichtlich der 1. bis 3. Normalform untersucht und (soweit nötig) normalisiert.

Zeitbedarf: 20 Stunden

Sequenzierung des Unterrichtsvorhabens

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Nutzung von relationalen Datenbanken (a) Aufbau von Datenbanken und Grundbegriffe • Entwicklung von Fragestellungen zur vorhandenen Datenbank • Analyse der Struktur der vorgegebenen Datenbank und Erarbeitung der Begriffe Tabelle, Attribut, Datensatz, Datentyp, Primärschlüssel, Fremdschlüssel, Datenbankschema (b) SQL-Abfragen • Analyse vorgegebener SQL-Abfragen und Erarbeitung der Sprachelemente von SQL (`SELECT (DISTINCT) …FROM, WHERE, AND, OR, NOT`) auf einer Tabelle • Analyse und Erarbeitung von SQL-Abfragen auf einer und mehrerer Tabelle zur Beantwortung der Fragestellungen (`JOIN, UNION, AS, GROUP BY,ORDER BY, ASC, DESC, COUNT, MAX, MIN, SUM`, Arithmetische Operatoren: `+, -, *, /, (…)`, Vergleichsoperatoren: `=, <>, >, <, >=, <=, LIKE, BETWEEN, IN, IS NULL`) (c) Vertiefung an einem weiteren Datenbankbeispiel	Die Schülerinnen und Schüler • erläutern die Eigenschaften und den Aufbau von Datenbanksystemen unter dem Aspekt der sicheren Nutzung (A), • analysieren und erläutern die Syntax und Semantik einer Datenbankabfrage (A), • analysieren und erläutern eine Datenbankmodellierung (A), • erläutern die Eigenschaften normalisierter Datenbankschemata (A), • bestimmen Primär- und Sekundärschlüssel (M), • ermitteln für anwendungsbezogene Problemstellungen Entitäten, zugehörige Attribute, Relationen und Kardinalitäten (M), • modifizieren eine Datenbankmodellierung (M), • modellieren zu einem Entity-Relationship-Diagramm ein relationales Datenbankschema (M), • bestimmen Primär- und Sekundärschlüssel (M), • überführen Datenbankschemata in vorgegebene Normalformen (M), • verwenden die Syntax und Semantik einer Datenbankabfragesprache, um Informationen aus einen Datenbanksystem zu extrahieren (I),	*Beispiel:* VideoCenter VideoCenter ist die Simulation einer Online-Videothek für den Informatik-Unterricht mit Webfrontends zur Verwaltung der Kunden, der Videos und der Ausleihe. Außerdem ist es möglich direkt SQL-Abfragen einzugeben. Es ist auch möglich, die Datenbank herunter zu laden und lokal zu installieren. Unter http://dokumentation.videocenter.schule.de/old/video/index.html (abgerufen: 30. 03. 2014) findet man den Link zu dem VideoCenter-System sowie nähere Informationen. Lesenswert ist auch die dort verlinkte „Dokumentation der Fallstudie" mit didaktischem Material, welches alternativ bzw. ergänzend zu der im Folgenden beschriebenen Durchführung verwendet werden kann. *Beispiel*: Schulbuchausleihe Unter Www.brd.nrw.de/lerntreffs/informatik/structure/material/sek2/datenbanken.php (abgerufen: 30. 03. 2014) wird eine Datenbank zur Verfügung gestellt, die Daten einer Schulbuch-Ausleihe enthält (über 1000 Entleiher, 200 Bücher mit mehreren tausend Exemplaren und viele Ausleihvorgänge). Die Datenbank kann in OpenOffice eingebunden werden.
2. Modellierung von relationalen Datenbanken		*Beispiel:* Fahrradverleih Der Fahrradverleih *BTR (BikesToRent)* verleiht

(a) Entity-Relationship-Diagramm • Ermittlung von Entitäten, zugehörigen Attributen, Relationen und Kardinalitäten in Anwendungssituationen und Modellierung eines Datenbankentwurfs in Form eines Entity-Relationship-Diagramms • Erläuterung und Modifizierung einer Datenbankmodellierung (b) Entwicklung einer Datenbank aus einem Datenbankentwurf • Modellierung eines relationalen Datenbankschematas zu einem Entity-Relationship-Diagramm inklusive der Bestimmung von Primär- und Sekundärschlüsseln (c) Redundanz, Konsistenz und Normalformen • Untersuchung einer Datenbank hinsichtlich Konsistenz und Redundanz in einer Anwendungssituation • Überprüfung von Datenbankschemata hinsichtlich der 1. bis 3. Normalform und Normalisierung (um Redundanzen zu vermeiden und Konsistenz zu gewährleisten)	• ermitteln Ergebnisse von Datenbankabfragen über mehrere verknüpfte Tabellen (D), • stellen Entitäten mit ihren Attributen und die Beziehungen zwischen Entitäten in einem Entity-Relationship-Diagramm grafisch dar (D), • überprüfen Datenbankschemata auf vorgegebene Normalisierungseigenschaften (D).	unterschiedliche Typen von Fahrrädern diverser Firmen an seine Kunden. Die Kunden sind bei *BTR* registriert (Name, Adresse, Telefon). *BTR* kennt von den Fahrradfirmen den Namen und die Telefonnummer. Kunden von *BTR* können CityBikes, Treckingräder und Mountainbikes ausleihen. *Beispiel*: Reederei Die Datenverwaltung einer Reederei soll in einem Datenbanksystem umgesetzt werden. Ausgehend von der Modellierung soll mit Hilfe eines ER-Modells und eines Datenbankschemas dieser erste Entwurf normalisiert und in einem Datenbanksystem umgesetzt werden. Es schließen sich diverse SQL-Abfragen an, wobei auf die Relationenalgebra eingegangen wird. *Beispiel:* Buchungssystem In dem Online-Buchungssystem einer Schule können die Lehrer Medienräume, Beamer, Laptops, Kameras, usw. für einen bestimmten Zeitpunkt buchen, der durch Datum und die Schulstunde festgelegt ist. Dazu ist die Datenbank zu modellieren, ggf. zu normalisieren und im Datenbanksystem umzusetzen. Weiter sollen sinnvolle Abfragen entwickelt werden. Unter http://mrbs.sourceforge.net (abgerufen: 30.03. 2014) findet man ein freies Online-Buchungssystem inklusive Demo, an Hand derer man erläutern kann, worum es in dem Projekt geht. *Beispiel: Schulverwaltung* In einer Software werden die Schulhalbjahre, Jahrgangsstufen, Kurse, Klassen, Schüler, Lehrer

		und Noten einer Schule verwaltet. Man kann dann ablesen, dass z.B. Schüler X von Lehrer Y im 2. Halbjahr des Schuljahrs 2011/2012 in der Jahrgangsstufe 9 im Differenzierungsbereich im Fach Informatik die Note „sehr gut“ erhalten hat. Dazu ist die Datenbank zu modellieren, ggf. zu normalisieren und im Datenbanksystem umzusetzen. Weiter sollen sinnvolle Abfragen entwickelt werden und das Thema Datenschutz besprochen werden.

Unterrichtsvorhaben Q1-V:

Thema: Sicherheit und Datenschutz in Netzstrukturen

Leitfragen: *Wie werden Daten in Netzwerken übermittelt? Was sollte man in Bezug auf die Sicherheit beachten?*

Vorhabenbezogene Konkretisierung:
Anschließend an das vorhergehende Unterrichtsvorhaben zum Thema Datenbanken werden der Datenbankzugriff aus dem Netz, Topologien von Netzwerken, eine Client-Server-Struktur, das TCP/IP-Schichtenmodell sowie Sicherheitsaspekte beim Zugriff auf Datenbanken und verschiedene symmetrische und asymmetrische kryptografische Verfahren analysiert und erläutert. Fallbeispiele zur Datenschutzproblematik und zum Urheberrecht runden das Unterrichtsvorhaben ab.

Zeitbedarf: 10 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Daten in Netzwerken und Sicherheitsaspekte in Netzen sowie beim Zugriff auf Datenbanken	Die Schülerinnen und Schüler	*Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben, Verschlüsselung Q1.5 - Zugriff auf Daten in Netzwerken

(a) Beschreibung eines Datenbankzugriffs im Netz anhand eines Anwendungskontextes und einer Client-Server-Struktur zur Klärung der Funktionsweise eines Datenbankzugriffs (b) Netztopologien als Grundlage von Client-Server-Strukturen und TCP/IP-Schichtenmodell als Beispiel für eine Paketübermittlung in einem Netz (c) Vertraulichkeit, Integrität, Authentizität in Netzwerken sowie symmetrische und asymmetrische kryptografische Verfahren (Cäsar-, Vigenère-, RSA-Verfahren) als Methoden Daten im Netz verschlüsselt zu übertragen	• beschreiben und erläutern Topologien, die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken (A), • analysieren und erläutern Eigenschaften und Einsatzbereiche symmetrischer und asymmetrischer Verschlüsselungsverfahren (A), • untersuchen und bewerten anhand von Fallbeispielen die Auswirkungen des Einsatzes von Informatiksystemen, die Sicherheit von Informatiksystemen sowie die Einhaltung der Datenschutzbestimmungen und des Urheberrechts (A), • untersuchen und bewerten Problemlagen, die sich aus dem Einsatz von Informatiksystemen ergeben, hinsichtlich rechtlicher Vorgaben, ethischer Aspekte und gesellschaftlicher Werte unter Berücksichtigung unterschiedlicher Interessenlagen (A), • nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zum Erschließen, zur Aufbereitung und Präsentation fachlicher Inhalte (D).	(Download Q1-V.1)
2. Fallbeispiele zur Datenschutzproblematik und zum Urheberrecht		*Materialien:* *Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1 5* - Datenschutz beim Videocenter, Materialblatt-Datenschutzgesetz (Download Q1-V.2)

Unterrichtsvorhaben Q2-I:

Thema: Modellierung und Implementierung von Anwendungen mit dynamischen, nichtlinearen Datenstrukturen

Leitfragen: *Wie können Daten im Anwendungskontext mit Hilfe binärer Baumstrukturen verwaltet werden? Wie kann dabei der rekursive Aufbau der Baumstruktur genutzt werden? Welche Vor- und Nachteile haben Suchbäume für die geordnete Verwaltung von Daten?*

Vorhabenbezogene Konkretisierung:
Anhand von Beispielen für Baumstrukturen werden grundlegende Begriffe eingeführt und der rekursive Aufbau binärer Bäume dargestellt. Anschließend werden für eine Problemstellung in einem der Anwendungskontexte Klassen modelliert und implementiert. Dabei werden die Operationen der Datenstruktur Binärbaum thematisiert und die entsprechende Klasse BinaryTree (der Materialien für das Zentralabitur in NRW) der Vorgaben für das Zentralabitur NRW verwendet. Klassen und ihre Beziehungen werden in Entwurfs- und Implementationsdiagrammen dargestellt. Die Funktionsweise von Methoden wird anhand grafischer Darstellungen von Binärbäumen erläutert.

Unter anderem sollen die verschiedenen Baumtraversierungen (Pre-, Post- und Inorder) implementiert werden. Unterschiede bezüglich der Möglichkeit, den Baum anhand der Ausgabe der Bauminhalte via Pre-, In- oder Postorder-Traversierung zu rekonstruieren, werden dabei ebenfalls angesprochen, indem die fehlende Umkehrbarbeit der Zuordnung Binärbaum → Inorder-Ausgabe an einem Beispiel verdeutlicht wird.

Eine Tiefensuche wird verwendet, um einen in der Baumstruktur gespeicherten Inhalt zu suchen.

Zu einer Problemstellung in einem entsprechenden Anwendungskontext werden die Operationen der Datenstruktur Suchbaum thematisiert und unter der Verwendung der Klasse BinarySearchTree (der Materialien für das Zentralabitur in NRW) weitere Klassen oder Methoden in diesem Anwendungskontext modelliert und implementiert. Auch in diesem Kontext werden grafische Darstellungen der Bäume verwendet.

Die Verwendung von binären Bäumen und Suchbäumen wird anhand weiterer Problemstellungen oder anderen Kontexten weiter geübt.

Zeitbedarf: 24 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Analyse von Baumstrukturen in verschiedenen Kontexten (a) Grundlegende Begriffe (Grad, Tiefe, Höhe, Blatt, Inhalt, Teilbaum, Ebene, Vollständigkeit)	Die Schülerinnen und Schüler • erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A),	*Beispiel:* Termbaum Der Aufbau von Termen wird mit Hilfe von binären Baumstrukturen verdeutlicht. oder *Beispiel*: Ahnenbaum

(b) Aufbau und Darstellung von binären Bäumen anhand von Baumstrukturen in verschiedenen Kontexten	• analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M), • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M), • verwenden bei der Modellierung geeigneter Problemstellungen die Möglichkeiten der Polymorphie (M), • entwickeln iterative und rekursive Algorithmen unter Nutzung der Konstruktionsstrategien „Modularisierung“ und „Teilen und Herrschen“ (M), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • modifizieren Algorithmen und Programme (I),	Die binäre Baumstruktur ergibt sich daraus, dass jede Person genau einen Vater und eine Mutter hat. *Weitere Beispiele für Anwendungskontexte für binäre Bäume:* *Beispiel*: Suchbäume (zur sortierten Speicherung von Daten) Alle Inhalte, die nach einer Ordnung vor dem Inhalt im aktuellen Teilbaum stehen, sind in dessen linkem Teilbaum, alle die nach dem Inhalt im aktuellen Teilbaum stehen, sind in dessen rechtem Teilbaum. (Dies gilt für alle Teilbäume.) *oder* *Beispiel*: Entscheidungsbäume Um eine Entscheidung zu treffen, werden mehrere Fragen mit ja oder nein beantwortet. Die Fragen, die möglich sind, wenn die Antwort auf eine Frage mit „ja“ beantwortet wird, befinden sich im linken Teilbaum, die Fragen, die möglich sind, wenn die Antwort „nein“ lautet, stehen im rechten Teilbaum. *oder* *Beispiel*: Codierungsbäume für Codierungen, deren Alphabet aus genau zwei Zeichen besteht Morse hat Buchstaben als Folge von Punkten und Strichen codiert. Diese Codierungen können in einem Binärbaum dargestellt werden, so dass ein Übergang zum linken Teilbaum einem Punkt und ein Übergang zum rechten Teilbaum einem

	• nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme systematisch anhand von Beispielen (I), • stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D).	Strich entspricht. Wenn man im Gesamtbaum startet und durch Übergänge zu linken oder rechten Teilbäumen einen Pfad zum gewünschten Buchstaben sucht, erhält man die Morsecodierung des Buchstabens. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.1 – Binärbaum (Download Q2-I.1)
2. Die Datenstruktur Binärbaum im Anwendungskontext unter Nutzung der Klasse `BinaryTree` (a) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen im Anwendungskontext (b) Modellierung eines Entwurfsdiagramms und Entwicklung eines Implementationsdiagramms (c) Erarbeitung der Klasse `BinaryTree` und beispielhafte Anwendung der Operationen (d) Implementierung der Anwendung oder von Teilen der Anwendung (e) Traversierung eines Binärbaums im Pre-, In- und Postorderdurchlauf		*Beispiel*: Informatikerbaum *als binärer Baum* In einem *binären Baum* werden die Namen und die Geburtsdaten von Informatikern lexikographisch geordnet abgespeichert. Alle Namen, die nach dieser Ordnung vor dem Namen im aktuellen Teilbaum stehen, sind in dessen linkem Teilbaum, alle die nach dem Namen im aktuellen Teilbaum stehen, sind in dessen rechtem Teilbaum. (Dies gilt für alle Teilbäume.) Folgende Funktionalitäten werden benötigt: • Einfügen der Informatiker-Daten in den Baum • Suchen nach einem Informatiker über den Schlüssel Name • Ausgabe des kompletten Datenbestands in nach Namen sortierter Reihenfolge *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.1 – Binärbaum (Download Q2-I.2)

3. Die Datenstruktur binärer Suchbaum im Anwendungskontext unter Verwendung der Klasse BinarySearchTree (a) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen (b) Modellierung eines Entwurfsdiagramms und Entwicklung eines Implementationsdiagramm, grafische Darstellung eines binären Suchbaums und Erarbeitung der Struktureigenschaften (c) Erarbeitung der Klasse BinarySearchTree und Einführung des Interface Item zur Realisierung einer geeigneten Ordnungsrelation (d) Implementierung der Anwendung oder von Teilen der Anwendung inklusive einer sortierten Ausgabe des Baums		*Beispiel*: Informatikerbaum als *Suchbaum* In einem binären *Suchbaum* werden die Namen und die Geburtsdaten von Informatikern lexikographisch geordnet abgespeichert. Alle Namen, die nach dieser Ordnung vor dem Namen im aktuellen Teilbaum stehen, sind in dessen linkem Teilbaum, alle die nach dem Namen im aktuellen Teilbaum stehen, sind in dessen rechtem Teilbaum. (Dies gilt für alle Teilbäume.) Folgende Funktionalitäten werden benötigt: • Einfügen der Informatiker-Daten in den Baum • Suchen nach einem Informatiker über den Schlüssel Name • Ausgabe des kompletten Datenbestands in nach Namen sortierter Reihenfolge *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.1 – Binärer Suchbaum (Download Q2-I.3)
4. Übung und Vertiefungen der Verwendung von Binärbäumen oder binären Suchbäumen anhand weiterer Problemstellungen		*Beispiel:* Codierungsbäume (s.o.) oder Huffman-Codierung *oder* *Beispiel*: Buchindex Es soll eine Anwendung entwickelt werden, die anhand von Stichworten und zugehörigen Seitenzahlen ein Stichwortregister erstellt. Da die Stichwörter bei der Analyse des Buches häufig gesucht werden müssen, werden sie in

		der Klasse `Buchindex` als Suchbaum (Objekt der Klasse `BinarySearchTree`) verwaltet. Alle Inhalte, die nach einer Ordnung vor dem Inhalt im aktuellen Teilbaum stehen, sind in dessen linkem Teilbaum, alle die nach dem Inhalt im aktuellen Teilbaum stehen, sind in dessen rechtem Teilbaum. (Dies gilt für alle Teilbäume.) *oder* *Beispiel*: Entscheidungsbäume (s.o.) *oder* *Beispiel*: Termbaum (s.o.) *oder* *Beispiel:* Ahnenbaum (s.o.) *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.1 – Anwendung Binärbaum (Download Q2-I.4)

Unterrichtsvorhaben Q2-II:

Thema: Endliche Automaten und formale Sprachen

Leitfragen: *Wie kann man (endliche) Automaten genau beschreiben? Wie können endliche Automaten (in alltäglichen Kontexten oder zu informatischen Problemstellungen) modelliert werden? Wie können Sprachen durch Grammatiken beschrieben werden? Welche Zusammenhänge gibt es zwischen formalen Sprachen, endlichen Automaten und regulären Grammatiken?*

Vorhabenbezogene Konkretisierung:
Anhand kontextbezogener Beispiele werden endliche Automaten entwickelt, untersucht und modifiziert. Dabei werden verschiedene Darstellungsformen für endliche Automaten ineinander überführt und die akzeptierten Sprachen endlicher Automaten ermittelt. An einem Beispiel wird ein nichtdeterministischer Akzeptor eingeführt als Alternative gegenüber einem entsprechenden deterministischen Akzeptor.

Anhand kontextbezogener Beispiele werden Grammatiken regulärer Sprachen entwickelt, untersucht und modifiziert. Der Zusammenhang zwischen regulären Grammatiken und endlichen Automaten wird verdeutlicht durch die Entwicklung von allgemeinen Verfahren zur Erstellung einer regulären Grammatik für die Sprache eines gegebenen endlichen Automaten bzw. zur Entwicklung eines endlichen Automaten, der genau die Sprache einer gegebenen regulären Grammatik akzeptiert.

Auch andere Grammatiken werden untersucht, entwickelt oder modifiziert. An einem Beispiel werden die Grenzen endlicher Automaten ausgelotet.

Zeitbedarf: 20 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien oder Materialien
1. Endliche Automaten (a) Vom Automaten in den Schülerinnen und Schülern bekannten Kontexten zur formalen Beschreibung eines endlichen Automaten (b) Untersuchung, Darstellung und Entwicklung endlicher Automaten	Die Schülerinnen und Schüler • analysieren und erläutern die Eigenschaften endlicher Automaten einschließlich ihres Verhaltens auf bestimmte Eingaben (A), • analysieren und erläutern Grammatiken regulärer Sprachen (A), • zeigen die Grenzen endlicher Automaten und regulärer Grammatiken im Anwendungszusammenhang auf (A), • ermitteln die formale Sprache, die durch eine Grammatik erzeugt wird (A), • entwickeln und modifizieren zu einer Problemstellung endliche Automaten (M), • entwickeln und modifizieren zu einer Problemstellung endliche Automaten (M), • entwickeln zur akzeptierten Sprache eines Automaten die zugehörige Grammatik (M), • entwickeln zur Grammatik einer regulären Sprache einen zugehörigen endlichen Automaten (M), • modifizieren Grammatiken regulärer Sprachen (M),	*Beispiele:* Cola-Automat, Geldspielautomat, Roboter, Zustandsänderung eines Objekts „Auto“, Akzeptor für bestimmte Zahlen, Akzeptor für Teilwörter in längeren Zeichenketten, Akzeptor für Terme *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.2 – Endliche Automaten, Formale Sprachen (Download Q2-II.1)
2. Untersuchung und Entwicklung von Grammatiken regulärer Sprachen (a) Erarbeitung der formalen Darstellung regulärer Grammatiken (b) Untersuchung, Modifikation und Entwicklung von Grammatiken (c) Entwicklung von endlichen Automaten zum Erkennen regulärer Sprachen die durch Grammatiken gegeben werden (d) Entwicklung regulärer Grammatiken zu endlichen Automaten		*Beispiele:* reguläre Grammatik für Wörter mit ungerader Parität, Grammatik für Wörter, die bestimmte Zahlen repräsentieren, Satzgliederungsgrammatik *Materialien: (s.o.)*
3. Grenzen endlicher Automaten		*Beispiele:* Klammerausdrücke, a^nb^n im Vergleich zu $(ab)^n$

	• entwickeln zu einer regulären Sprache eine Grammatik, die die Sprache erzeugt (M), • stellen endliche Automaten in Tabellen oder Graphen dar und überführen sie in die jeweils andere Darstellungsform (D), • ermitteln die Sprache, die ein endlicher Automat akzeptiert (D). • beschreiben an Beispielen den Zusammenhang zwischen Automaten und Grammatiken (D).	

Unterrichtsvorhaben Q2-III:

Thema: Prinzipielle Arbeitsweise eines Computers und Grenzen der Automatisierbarkeit

Leitfragen: *Was sind die strukturellen Hauptbestandteile eines Computers und wie kann man sich die Ausführung eines maschinenahen Programms mit diesen Komponenten vorstellen? Welche Möglichkeiten bieten Informatiksysteme und wo liegen ihre Grenzen?*

Vorhabenbezogene Konkretisierung:
Anhand einer von-Neumann-Architektur und einem maschinennahen Programm wird die prinzipielle Arbeitsweise von Computern verdeutlicht.

Ausgehend von den prinzipiellen Grenzen endlicher Automaten liegt die Frage nach den Grenzen von Computern bzw. nach Grenzen der Automatisierbarkeit nahe. Mit Hilfe einer entsprechenden Java-Methode wird plausibel, dass es unmöglich ist, ein Informatiksystem zu entwickeln, dass für jedes beliebige Computerprogramm und jede beliebige Eingabe entscheidet ob das Programm mit der Eingabe terminiert oder nicht (Halteproblem). Anschließend werden Vor- und Nachteile der Grenzen der Automatisierbarkeit angesprochen und der Einsatz von Informatiksystemen hinsichtlich prinzipieller Möglichkeiten und prinzipieller Grenzen beurteilt.

Zeitbedarf: 12 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien oder Materialien
1. Von-Neumann-Architektur und die Ausführung maschinennaher Programme a) prinzipieller Aufbau einer von Neumann-Architektur mit CPU, Rechenwerk, Steuerwerk, Register und Hauptspeicher b) einige maschinennahe Befehlen und ihre Repräsentation in einem Binär-Code, der in einem Register gespeichert werden kann c) Analyse und Erläuterung der Funktionsweise eines einfachen maschinennahen Programms	Die Schülerinnen und Schüler • erläutern die Ausführung eines einfachen maschinennahen Programms sowie die Datenspeicherung auf einer „Von-Neumann-Architektur“ (A), • untersuchen und beurteilen Grenzen des Problemlösens mit Informatiksystemen (A).	*Beispiel:* Addition von 4 zu einer eingegeben Zahl mit einem Rechnermodell *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.3 –Von-Neumann-Architektur und maschinennahe Programmierung (Download Q2-III.1)
2. Grenzen der Automatisierbarkeit a) Vorstellung des Halteproblems b) Unlösbarkeit des Halteproblems c) Beurteilung des Einsatzes von Informatiksystemen hinsichtlich prinzipieller Möglichkeiten und prinzipieller Grenzen		*Beispiel:* Halteproblem *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.3 - Halteproblem (Download Q2-III.2)

Unterrichtsvorhaben Q2-IV:

Wiederholung und Vertiefung ausgewählter Kompetenzen und Inhalte des ersten Jahrs der Qualifikationsphase

III) Qualifikationsphase - LEISTUNGSKURS

Die folgenden Kompetenzen aus dem Bereich *Kommunizieren und Kooperieren* werden in allen Unterrichtsvorhaben der Qualifikationsphase vertieft und sollen aus Gründen der Lesbarkeit nicht in jedem Unterrichtsvorhaben separat aufgeführt werden:

Die Schülerinnen und Schüler

- verwenden die Fachsprache bei der Kommunikation über informatische Sachverhalte (K),
- nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Dateien unter Berücksichtigung der Rechteverwaltung (K),
- organisieren und koordinieren kooperatives und eigenverantwortliches Arbeiten (K),
- strukturieren den Arbeitsprozess, vereinbaren Schnittstellen und führen Ergebnisse zusammen (K),
- beurteilen Arbeitsorganisation, Arbeitsabläufe und Ergebnisse (K),
- präsentieren Arbeitsabläufe und -ergebnisse adressatengerecht (K).

Unterrichtsvorhaben Q1-I:

Thema: Wiederholung der objektorientierten Modellierung und Programmierung anhand einer kontextbezogenen Problemstellung unter Berücksichtigung der Gestaltung einer Benutzungsoberfläche

Leitfragen: *Wie modelliert und implementiert man zu einer Problemstellung in einem geeigneten Anwendungskontext Java-Klassen inklusive ihrer Attribute, Methoden und Beziehungen? Wie kann man die Modellierung und die Funktionsweise der Anwendung grafisch darstellen? Wie kann ein Softwareprojekt mit angemessener Benutzungsoberfläche arbeitsteilig entwickelt und getestet werden?*

Vorhabenbezogenen Konkretisierung:
Zu einer Problemstellung in einem Anwendungskontext soll eine Java-Anwendung entwickelt werden. Diese Anwendung soll aus einer Benutzungsoberfläche und einer davon getrennten Verarbeitung der mit der Oberfläche eingegebenen Daten bestehen. Auf diese Weise werden nicht nur Grundideen der Objektorientierung wiederholt, sondern auch erste Schritte zur Thematisierung des MVC-Prinzips realisiert.

Die Schülerinnen und Schüler analysieren zu Beginn die gegebene Problemstellung und formulieren Anforderungen an die zu entwickelnde Java-Anwendung. Anschließend wird eine Benutzungsoberfläche nach softwareergonomischen Kriterien konzipiert und mit einem GUI-Builder umgesetzt.

Mit Blick auf die fertige Benutzungsoberfläche wird eine Objektorientierte Modellierung entwickelt und in Form eines Implementationsdiagramms fixiert. Wesentliche Klassen werden von Schülerinnen und Schülern dokumentiert. Der Nachrichtenaustausch zwischen verschiedenen Objekten wird verdeutlicht, indem die Kommunikation zwischen zwei ausgewählten Objekten grafisch dargestellt wird. In diesem Zusammenhang wird das Nachrichtenkonzept der Objektorientierten Programmierung wiederholt. Eine Implementierung der Java-Anwendung kann arbeitsteilig erfolgen. Zum Schluss wird das fertige Projekt mit Hilfe einer Testanwendung getestet.

Zeitbedarf: 15 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
2. Entwicklung eines Anforderungskatalogs (g) Analyse der Problemstellung (h) Entwicklung von Anwendungsfällen (use cases) (i) Formulierung eines Anforderungskatalogs	Die Schülerinnen und Schüler • analysieren und erläutern objektorientierte Modellierungen (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M), • entwickeln mit didaktisch orientierten Entwicklungsumgebungen einfache Benutzungsoberflächen zur Kommunikation mit einem Informatiksystem (M), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • wenden eine didaktisch orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung	*Beispiel:* Taschenrechner Schülerinnen und Schüler entwickeln eine Taschenrechneranwendung. Dabei kann es sich um einen Rechner mit einfachen Grundrechenarten bis hin zu einem wissenschaftlichen Taschenrechner mit umfangreicheren Funktionen handeln. Die Anforderungen sind von den Schülerinnen und Schülern selbst zu definieren.
3. Entwurf einer grafischen Benutzungsoberfläche (a) Erarbeitung softwareergonomischer Prinzipien für Benutzungsoberflächen (b) Entwurf von Prototypen für eine Benutzungsoberfläche unter Berücksichtigung des Anforderungskatalogs (c) Einarbeitung in die Funktionsweise eines GUI-Builders (d) Realisierung einer Benutzungsoberfläche mit einem GUI-Builder		*Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.I (Download LK-Q1-I.1)
4. Klassenmodellierung und Visualisierung von Objektkommunikation (a) Entwurf eines Implementationsdiagramms unter Berücksichtigung des Anforderungskatalogs und der Benutzungsoberfläche		

(b) Grafische Darstellung der Kommunikation zwischen mindestens zwei Objekten (c) Dokumentation von Klassen	und zum Test von Informatiksystemen an (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme systematisch anhand von Beispielen und mithilfe von Testanwendungen(I), • stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen (D), • stellen die Kommunikation zwischen Objekten grafisch dar (D), • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Daten, zur Organisation von Arbeitsabläufen sowie zur Verteilung und Zusammenführung von Arbeitsanteilen (K).	
5. Projektentwicklung und Test (ggf. arbeitsteilig) (a) Implementierung der Anwendung mit grafischer Benutzungsoberfläche (b) Test einzelner Klassen der Gesamtanwendung mit Hilfe von Testanwendungen		*Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.I (Download LK-Q1-I.2)

Unterrichtsvorhaben Q1-II:

Thema: Modellierung und Implementierung von dynamischen linearen Datenstrukturen und von Anwendungen mit dynamischen linearen Datenstrukturen in kontextbezogenen Problemstellungen

Leitfragen: *Wie können beliebig viele linear angeordnete Daten in unterschiedlichen Anwendungskontexten problemgerecht verwaltet werden?*

Vorhabenbezogenen Konkretisierung:
Nach Analyse einer Problemstellung in einem geeigneten Anwendungskontext, in dem Daten nach dem First-In-First-Out-Prinzip verwaltet werden, wird der Aufbau von einer geeigneten Datenstruktur entwickelt und implementiert. Es werden unterschiedliche Implementationsmöglichkeiten verglichen. Anschließend werden die für die Anwendung notwendigen Klassen modelliert und implementiert. Die Klasse `Queue` wird als Alternative zur selbstentwickelten Datenstruktur vorgestellt. Anschließend wird die Anwendung modifiziert, um den Umgang mit der Datenstruktur zu üben.

Anhand einer Anwendung, in der Daten nach dem Last-In-First-Out-Prinzip verwaltet werden, werden Unterschiede zwischen den Datenstrukturen Schlange und Stapel erarbeitet. Eine der beiden Klassen zu den linearen Strukturen sollte vollständig modelliert, implementiert und dokumentiert werden.

Um einfacher an Objekte zu gelangen, die zwischen anderen gespeichert sind, wird ein Anwendungskontext vorgestellt, bei dem die Anwendung der Klassen `Queue` und `Stack` nicht problemgerecht ist. Die Klasse `List`, bei der das Einfügen an beliebiger Stelle einer sequentiell angeordneten Struktur möglich ist, wird eingeführt und in einem Anwendungskontext verwendet.

In mindestens einem weiteren Anwendungskontext wird die Verwaltung von Daten in Schlangen, Stapeln oder Listen vertieft. Modellierungen werden dabei in Entwurfs- und Implementationsdiagrammen dargestellt.

Zeitbedarf: 25 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
5. Modellierung und Implementation einer Datenstruktur Schlange, die die Daten nach dem FIFO-Prinzip im Anwendungskontext verwaltet.	Die Schülerinnen und Schüler • modellieren Klassen mit ihren Attributen, Methoden und Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), • erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), • analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare und nichtlineare Datensammlungen zu (M), • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften,	*Beispiel:* (Schlange) Patientenwarteschlange Sobald eine Patientin oder ein Patient in einer Arztpraxis eintrifft, werden Name und Krankenkasse erfasst. Die Verwaltung der Patientenwarteschlange geschieht über eine Klasse, die hier als Wartezimmer bezeichnet wird. Wesentliche Operationen sind das „Hinzufügen“ von Patientinnen und Patienten sowie das „Entfernen“ von Patientinnen und Patienten, nachdem sie zur Behandlung gerufen wurden. Die Simulationsanwendung stellt eine GUI zur Verfügung, legt ein Wartezimmer an und steuert die Abläufe. Erster Lösungsansatz: Die FIFO-Datenstruktur wird problembezogen modelliert und implementiert. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.II (Download LK-Q1-II.1)

6. Die Datenstruktur Schlange im Anwendungskontext unter Nutzung der Klasse `Queue` (d) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen (e) Erarbeitung der Funktionalität der Klasse `Queue` und des Konzepts der parametrisierten Klassen Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse `Queue`	ihre Operationen und ihre Beziehungen (M), • modifizieren Algorithmen und Programme (I), • implementieren Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (I), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I), • stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D).	Zweiter Lösungsansatz: Bei der Modellierung und Implementation wird der generische Datentyp `Queue` verwendet. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.II (Download LK-Q1-II.2)
7. Die Datenstruktur Stapel im Anwendungskontext unter Nutzung der Klasse `Stack` (d) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen (e) Erarbeitung der Funktionalität der Klasse `Stack` (f) Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse `Stack`		*Beispiel:* (Stapel) Handdechiffrierer mit Undo-Operation. In einem Text sind Buchstaben vertauscht worden, er wird so zu einem Geheimtext. Dieser Text wird in ein Softwareprodukt kopiert und schrittweise dechiffriert. Es können für jedes Zeichen Ersetzungsvorschläge gemacht werden und die Software setzt diese Ersetzung im gesamten Text um. Am Kontrolltext bemerkt man, ob die Ersetzungen sinnvoll waren. Falls man einen Irrweg gegangen ist, kann man die Ersetzungen bis zu dem gewünschten Punkt schrittweise rückgängig machen. Die Modellierung und die Implementation erfolgt mit dem generischen Datentyp `Stack`. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.II (Download LK-Q1-II.3)

8. Die Datenstruktur Lineare Liste im Anwendungskontext unter Nutzung der Klasse `List` (c) Erarbeitung der Vorteile der Klasse `List` im Gegensatz zu den bereits bekannten linearen Strukturen (d) Modellierung und Implementierung einer kontextbezogenen Anwendung unter Verwendung der Klasse `List`		*Beispiel:* (Liste) Todo-Liste Es werden Aufgaben verwaltet, die in eine vom Benutzer gewünschte Reihenfolge angeordnet werden. Der Benutzer kann in der Todo-Liste navigieren, bestehende Einträge ändern oder löschen, sowie vor oder hinter einem Eintrag neue Aufgaben einfügen. Ein Test, in dem alle Aufgaben aufgelistet sind, kann ebenfalls abgerufen werden. Die Modellierung und die Implementation erfolgt mit dem generischen Datentyp `List`. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.II (Download LK-Q1-II.4)

9. Vertiefung – Anwendungen von Listen, Stapeln oder Schlangen in mindestens einem weiteren Kontext (a) Modellierung und Implementierung einer weiteren Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse `Queue` (b) Modellierung und Implementierung einer weiteren Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse `Stack` (c) Modellierung und Implementierung einer weiteren kontextbezogenen Anwendung unter Verwendung der Klasse `List`		***Beispiele für Vertiefungen*** *Beispiel:* (Schlange) Auslastungssimulation einer Arztpraxis. Bei einer Arztpraxis wird die Auslastung des Arztes während der Sprechstunde simuliert. Eingegeben werden die Öffnungszeiten und die voraussichtliche Anzahl der zu erwartenden Patienten. Die Behandlungsdauern der Patientinnen und Patienten werden zufällig gesetzt. Ergebnis der Simulation sind die Zeiten, die die Arztpraxis nicht ausgelastet ist und die Zeit, die am Ende der Sprechstunde notwendig ist, um alle noch wartenden Patienten zu behandeln. *Beispiel:* (Stapel) Terme in Postfix-Notation Die sog. UPN (*umgekehrte polnische Notation*) bzw. *Postfix-Notation* eines Terms setzt den Operator hinter die Operanden. Um den Wert des Terms zu berechnen, kann ein Stack verwendet werden. *Beispiel:* (Liste) Verwaltung einer Todo-Liste mit Prioritäten (Todo-Listenmanager) Es werden Aufgaben verwaltet, die mit einer gewissen Priorität erledigt werden sollen. Es werden drei Prioritätsstufen unterschieden und die Aufgaben so angeordnet, dass sie in passender Reihenfolge stehen.

Unterrichtsvorhaben Q1-III:

Thema: Modellierung, Implementierung, Analyse und Beurteilung von Such- und Sortierverfahren unterschiedlicher Komplexitätsklassen in kontextbezogenen Problemstellungen

Leitfrage: *Wie kann man gespeicherte Informationen günstig (wieder-)finden?*

Vorhabenbezogene Konkretisierung:
In einem Anwendungskontext werden zunächst Informationen in einer linearen Liste bzw. einem Feld gesucht. Hierzu werden Verfahren entwickelt und implementiert bzw. analysiert und erläutert, wobei neben einem iterativen auch ein rekursives Verfahren thematisiert wird und mindestens ein Verfahren selbst entwickelt und implementiert wird. Die verschiedenen Verfahren werden hinsichtlich Speicherbedarf und Zahl der Vergleichsoperationen miteinander verglichen.

Anschließend werden Sortierverfahren entwickelt und implementiert (ebenfalls für lineare Listen und Felder). Hierbei soll auch ein rekursives Sortierverfahren entwickelt werden. Die Implementationen von Quicksort sowie dem Sortieren durch Einfügen werden analysiert und erläutert. Falls diese Verfahren vorher schon entdeckt wurden, sollen sie hier wiedererkannt werden. Die rekursive Abarbeitung eines Methodenaufrufs von Quicksort wird grafisch dargestellt.

Abschließend werden verschiedene Sortierverfahren hinsichtlich der Anzahl der benötigten Vergleichsoperationen und des Speicherbedarfs beurteilt.

Zeitbedarf: 20 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
4. Suchen von Daten in Listen und Arrays (d) Lineare Suche in Listen und in Arrays (e) Binäre Suche in Arrays als Beispiel für rekursives Problemlösen (f) Untersuchung der beiden Suchverfahren hinsichtlich ihrer Effizienz (Speicherbedarf, Anzahl der Vergleiche)	Die Schülerinnen und Schüler • analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A), • entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“ , „Teilen und Herrschen“ und „Backtracking“(M), • modifizieren Algorithmen und Programme (I), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • implementieren und erläutern iterative und rekursive Such- und Sortierverfahren unterschiedlicher Komplexitätsklassen (Speicherbedarf und Laufzeitverhalten)(I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I),	*Beispiel:* Benutzerverwaltung für einen Fotokopierer Benutzerinnen und Benutzer des Kopierers geben eine vierstellige Ziffernkombination an, um sich zu legitimieren. Über diesen PIN-Code soll auch die Benutzerinnen und Benutzer des Kopierers identifizierbar sein, um auf sein Konto den Kopierverbrauch zu buchen. Die Benutzungsdaten werden in linearen Strukturen verwaltet, die nach den PIN-Nummern sortiert vorliegen. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.III - Suchen und Sortieren (Download Q1-III.1)
5. Sortieren in Listen und Arrays - Entwicklung und Implementierung von iterativen und rekursiven Sortierverfahren (d) Entwicklung und Implementierung eines einfachen Sortierverfahrens für eine Liste (e) Implementierung eines einfachen Sortierverfahrens für ein Feld (f) Entwicklung eines rekursiven Sortierverfahren für ein Feld (z.B. Sortieren durch Mischen)		*Beispiel:* Benutzerverwaltung für einen Fotokopierer (s.o.) Die Benutzungsdaten sollen nach den Kriterien Kopierverbrauch, Namen und PIN-Nummern geordnet ausgegeben werden können. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.III - Suchen und Sortieren (Download Q1-III.2)

6. Untersuchung der Effizienz der Sortierverfahren „Sortieren durch direktes Einfügen“ und „Quicksort“ auf linearen Listen (d) Grafische Veranschaulichung der Sortierverfahren (e) Untersuchung der Anzahl der Vergleichsoperationen bei beiden Sortierverfahren (f) Beurteilung der Effizienz der beiden Sortierverfahren, untere Schranke für die Laufzeit von Sortieralgorithmen	• testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D).	*Beispiel:* Test- und Analyseumgebung für Sortieralgorithmen Listen mit unterschiedlich vielen Listenelementen und unterschiedlicher Vorsortierung werden bezüglich der Anzahl der Vergleiche von Listenelementen analysiert. Die untere Schranke für die Laufzeit wird bestimmt. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.III - Suchen und Sortieren (Download Q1-III.3)

Unterrichtsvorhaben Q1-IV:

Thema: Modellierung, Implementierung und Nutzung von relationalen Datenbanken in Anwendungskontexten

Leitfragen*: Wie können Fragestellungen mit Hilfe einer Datenbank beantwortet werden? Wie entwickelt man selbst eine Datenbank für einen Anwendungskontext?*

Vorhabenbezogene Konkretisierung:
Ausgehend von einer vorhandenen Datenbank entwickeln Schülerinnen und Schüler für sie relevante Fragestellungen, die mit dem vorhandenen Datenbestand beantwortet werden sollen. Zur Beantwortung dieser Fragestellungen wird die vorgegebene Datenbank von den Schülerinnen und Schülern analysiert, und die notwendigen Grundbegriffe für Datenbanksysteme sowie die erforderlichen SQL-Abfragen werden erarbeitet.

In anderen Anwendungskontexten müssen Datenbanken erst noch entwickelt werden, um Daten zu speichern und Informationen für die Beantwortung von möglicherweise auftretenden Fragen zur Verfügung zu stellen. Dafür ermitteln Schülerinnen und Schüler in den Anwendungssituationen Entitätstypen, Attribute der zugehörigen Entitäten, Beziehungstypen und Kardinalitäten und stellen diese in Entity-Relationship-Diagramm dar. Entity-Relationship-Modelle werden interpretiert und erläutert, modifiziert und in Datenbankschemata überführt. Mit Hilfe von SQL-Anweisungen können anschließend im Kontext relevante Informationen aus der Datenbank extrahiert werden.

Ein Entity-Relationship-Diagramm kann auch verwendet werden, um die Entitätstypen inklusive der Attribute ihrer Entitäten und die Beziehungstypen in einem vorgegebenen Datenbankschema darzustellen.

An einem Beispiel wird verdeutlicht, dass in Datenbanken Redundanzen unerwünscht sind und Konsistenz gewährleistet sein sollte. Die 1. bis 3. Normalform wird als Gütekriterium für Datenbankentwürfe eingeführt. Datenbankschemata werden hinsichtlich der 1. bis 3. Normalform untersucht und (soweit nötig) normalisiert.

Zeitbedarf: 20 Stunden

Sequenzierung des Unterrichtsvorhabens

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
3. Nutzung von relationalen Datenbanken (d) Aufbau von Datenbanken und Grundbegriffe • Entwicklung von Fragestellungen zur vorhandenen Datenbank • Analyse der Struktur der vorgegebenen Datenbank und Erarbeitung der Begriffe Tabelle, Attribut, Datensatz, Datentyp, Primärschlüssel, Fremdschlüssel, Relationenschema, Datenbankschema (e) SQL-Abfragen • Analyse vorgegebener SQL-Abfragen und Erarbeitung der Sprachelemente von SQL (`SELECT (DISTINCT) …FROM, WHERE, AND, OR, NOT`) auf einer Tabelle • Analyse und Erarbeitung von SQL-Abfragen auf einer und mehreren Tabelle zur Beantwortung der Fragestellungen (`JOIN` und Varianten, `UNION, AS, GROUP BY,ORDER BY, ASC, DESC, COUNT, MAX, MIN, SUM, AVG`, Arithmetische Operatoren: `+, -, *, /, (…)`, Vergleichsoperatoren: `=, <>, >, <, >=, <=, LIKE, BETWEEN, IN, IS NULL`)	Die Schülerinnen und Schüler • ermitteln für anwendungsbezogene Problemstellungen Entitäten, zugehörige Attribute, Relationen und Kardinalitäten (M), • stellen Entitäten mit ihren Attributen und die Beziehungen zwischen Entitäten mit Kardinalitäten in einem Entity-Relationship-Diagramm grafisch dar (D), • modifizieren eine Datenbankmodellierung (M), • modellieren zu einem Entity-Relationship-Diagramm ein relationales Datenbankschema (M), • bestimmen Primär- und Sekundärschlüssel (M), • implementieren ein relationales Datenbankschema als Datenbank (I), • analysieren und erläutern eine Datenbankmodellierung (A), • erläutern die Eigenschaften normalisierter Datenbankschemata (A), • überprüfen Datenbankschemata auf vorgegebene Normalisierungseigenschaften (D),	*Beispiel:* VideoCenter VideoCenter ist die Simulation einer Online-Videothek für den Informatik-Unterricht mit Webfrontend zur Verwaltung der Kunden, der Videos und der Ausleihe. Außerdem ist es möglich direkt SQL-Abfragen einzugeben. Es ist auch möglich, die Datenbank herunterzuladen und lokal zu installieren. Unter `http://dokumentation.videocenter.schule.de/old/video/index.html` (abgerufen: 01. 02. 2016) findet man den Link zu dem VideoCenter-System sowie nähere Informationen. Lesenswert ist auch die dort verlinkte „Dokumentation der Fallstudie" mit didaktischem Material, welches alternativ bzw. ergänzend zu der im Folgenden beschriebenen Durchführung verwendet werden kann. *Beispiel*: Schulbuchausleihe Unter `Www.brd.nrw.de/lerntreffs/informatik/structure/material/sek2/datenbanken.php` (abgerufen: 01. 02. 2016) wird eine Datenbank zur Verfügung gestellt, die Daten einer Schulbuch-Ausleihe enthält (über 1000 Entleiher, 200 Bücher mit mehreren tausend Exemplaren und viele Ausleihvorgänge). Die Datenbank kann in OpenOffice eingebunden werden.

(f) Vertiefung an einem weiteren Datenbankbeispiel	• überführen Datenbankschemata in die 1. bis 3. Normalform (M), • ermitteln Ergebnisse von Datenbankabfragen über mehrere verknüpfte Tabellen (D), • analysieren und erläutern die Syntax und Semantik einer Datenbankabfrage (A), • verwenden die Syntax und Semantik einer Datenbankabfragesprache, um Informationen aus einem Datenbanksystem zu extrahieren (I), • erläutern Eigenschaften und Aufbau von Datenbanksystemen unter dem Aspekt der sicheren Nutzung (A).	
4. Modellierung von relationalen Datenbanken (d) Entity-Relationship-Diagramm • Ermittlung von Entitätstypen, Attributen zugehöriger Entitäten, Beziehungstypen und Kardinalitäten in Anwendungssituationen und Modellierung eines Datenbankentwurfs in Form eines Entity-Relationship-Diagramms • Erläuterung und Modifizierung einer Datenbankmodellierung (e) Entwicklung einer Datenbank aus einem Datenbankentwurf • Modellierung eines relationalen Datenbankschemas zu einem Entity-Relationship-Diagramm inklusive der Bestimmung von Primär- und Sekundärschlüsseln (f) Redundanz, Konsistenz und Normalformen • Untersuchung einer Datenbank hinsichtlich Konsistenz und Redundanz in einer Anwendungssituation		*Beispiel:* Fahrradverleih Der Fahrradverleih *BTR (BikesToRent)* verleiht unterschiedliche Typen von Fahrrädern diverser Firmen an seine Kundinnen und Kunden. Die Kundinnen und Kunden sind bei *BTR* registriert (Name, Adresse, Telefon). *BTR* kennt von den Fahrradfirmen den Namen und die Telefonnummer. Kundinnen und Kunden von *BTR* können CityBikes, Trekkingräder und Mountainbikes ausleihen. *Beispiel*: Reederei Die Datenverwaltung einer Reederei soll in einem Datenbanksystem umgesetzt werden. Ausgehend von der Modellierung soll mit Hilfe eines ER-Modells und eines Datenbankschemas dieser erste Entwurf normalisiert und in einem Datenbanksystem umgesetzt werden. Es schließen sich diverse SQL-Abfragen an, wobei auf die Relationenalgebra eingegangen wird. *Beispiel:* Buchungssystem In dem Online-Buchungssystem einer Schule können die Lehrenden Medienräume, Beamer, Laptops, Kameras, usw. für einen bestimmten

• Überprüfung von Datenbankschemata hinsichtlich der 1. bis 3. Normalform und Normalisierung (um Redundanzen zu vermeiden und Konsistenz zu gewährleisten)		Zeitpunkt buchen, der durch Datum und die Schulstunde festgelegt ist. Dazu ist die Datenbank zu modellieren, ggf. zu normalisieren und im Datenbanksystem umzusetzen. Weiter sollen sinnvolle Abfragen entwickelt werden. Unter http://mrbs.sourceforge.net (abgerufen: 01. 02. 2016) findet man ein freies Online-Buchungssystem inklusive Demo, an Hand derer man erläutern kann, worum es in dem Projekt geht. *Beispiel: Schulverwaltung* In einer Software werden die Schulhalbjahre, Jahrgangsstufen, Kurse, Klassen, Schülerinnen und Schüler, Lehrende und Noten einer Schule verwaltet. Man kann dann ablesen, dass z.B. Schüler X von Lehrerin Y im 2. Halbjahr des Schuljahrs 2011/2012 in der Jahrgangsstufe 9 im Differenzierungsbereich im Fach Informatik die Note „sehr gut“ erhalten hat. Dazu ist die Datenbank zu modellieren, ggf. zu normalisieren und im Datenbanksystem umzusetzen. Weiter sollen sinnvolle Abfragen entwickelt werden und auf das Thema Datenschutz eingegangen werden.

Unterrichtsvorhaben Q1 – V:

Thema: Modellierung und Implementierung von dynamischen nicht-linearen Datenstrukturen und von Anwendungen mit dynamischen nicht-linearen Datenstrukturen in kontextbezogenen Problemstellungen.

Leitfragen: *Wie können Daten im Anwendungskontext mit Hilfe von Graphen und binärer Baumstrukturen verwaltet werden? Wie kann in einem Graphen ein beliebiger, alle oder der kürzeste Weg zwischen zwei Knoten effizient gefunden werden? Welche Kanten müssen in einem zusammenhängenden Graphen mindestens verbleiben, sodass dieser bei minimaler Summe der Kantengewichte weiterhin zusammenhängend ist. Wie kann ich mit einem Binärbaum Daten sortiert verwalten?*

Vorhabenbezogene Konkretisierung:
Anhand von Beispielen für Graphen werden grundlegende Begriffe eingeführt und die beiden Darstellungsformen Adjazenzmatrix und Adjazenzliste erarbeitet. Die Funktionalität der Methoden der Klassen `Graph`, `Vertex` und `Edge` aus den Vorgaben für das Zentralabitur NRW wird erarbeitet. Die Fragestellung nach der Suche eines oder aller Wege zwischen zwei Knoten in einem Graphen motiviert die Erarbeitung von Algorithmen zur Tiefen- und Breitensuche, mit denen Graphen systematisch durchsucht werden können. Anschließend werden anhand von Anwendungskontexten Algorithmen für die Bestimmung kürzester Wege in einem Graphen sowie zur Konstruktion von minimalen Spannbäumen modelliert und zum Teil auch implementiert.
Die Datenstruktur Binärbaum mit den notwendigen Begrifflichkeiten wird als Spezialfall eines Graphen anhand von Anwendungskontexten erarbeitet. Die entsprechende Klasse `BinaryTree<ContentType>` der Vorgaben für das Zentralabitur NRW wird zur Modellierung und Implementierung verschiedener Problemstellungen verwendet. Unter anderem sollen die verschiedenen Baumtraversierungen (Pre-, Post- und In-Order) implementiert werden.
Mithilfe einer anwendungsorientierten Problemstellung werden die Operationen der Datenstruktur Suchbaum thematisiert und die Klasse `BinarySearchTree<ContentType>` (der Materialien für das Zentralabitur in NRW) modelliert und zumindest partiell implementiert.

Zeitbedarf: 40 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Analyse von Graphen in verschiedenen Kontexten (a) Grundlegende Begriffe (Graph, gerichtet – ungerichtet, Knoten, Kanten, Kantengewicht) (b) Aufbau und Darstellung von Graphen anhand von Graphenstrukturen in verschiedenen Kontexten (Adjazenzmatrix, Adjazenzliste)	Die Schülerinnen und Schüler • erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), • analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A), • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare und nichtlineare Datensammlungen zu (M), • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M),	*Beispiel:* „Das Haus vom Nikolaus“ Das Haus vom Nikolaus ist das bekannteste Beispiel für einen Graphen, für den ein Eulerweg, aber kein Eulerkreis existiert. Anhand dieses Beispiels werden die Grundbegriffe der Graphentheorie sowie die Darstellung eines Graphen als Adjazenzmatrix eingeführt. *Beispiel* Soziale Netzwerke Da es in dem Graph eines sozialen Netzwerks im Verhältnis zu den Knoten nur wenige Kanten gibt, bietet sich dieses Beispiel zur Einführung der Darstellung eines Graphen in Form von Adjazenzlisten an. *Materialien*: Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben LK-Q1.V (Download LK-Q1.V.1)

2. Die Datenstruktur `Graph` im Anwendungskontext unter Nutzung der Klassen `Graph`, `Vertex` und `Edge`. (a) Erarbeitung der Klassen `Graph`, `Vertex` und `Edge` und beispielhafte Anwendung der Operationen (b) Bestimmung von Wegen in Graphen im Anwendungskontext (Tiefensuche, Breitensuche) (c) Bestimmung von kürzesten Wegen in Graphen im Anwendungskontext (Backtracking, Dijkstra). (d) Bestimmung von minimalen Spannbäumen eines Graphen im Anwendungskontext.	• verwenden bei der Modellierung geeigneter Problemstellungen die Möglichkeiten der Polymorphie (M), • entwickeln iterative und rekursive Algorithmen unter Nutzung der Konstruktionsstrategien „Modularisierung“ und „Teilen und Herrschen“ und „Backtracking“(M), • entwickeln mit didaktisch orientierten Entwicklungsumgebungen einfache Benutzungsoberflächen zur Kommunikation mit einem Informatiksystem (M), • implementieren Operationen dynamischer (linearer oder nichtlinearer) Datenstrukturen (I), • implementieren und erläutern iterative und rekursive Such- und Sortierverfahren unterschiedlicher Komplexitätsklassen (Speicherbedarf und Laufzeitverhalten) (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme systematisch anhand von Beispielen und mithilfe von Testanwendungen(I),	*Beispiel:* Soziale Netzwerke Ausgehend von dem Problem der Berechnung der Dichte eines sozialen Netzwerkes werden die Funktionalitäten der Methoden der Klassen Graph, Vertex und Edge erarbeitet und erste Beispiele modelliert und implementiert: • Konstruktion eines Beispielgraphen • Anzahl der Knoten • Summe der Kantengewichte • Anzahl der Nachbarn eines Knotens *Beispiel:* Wegsuche Ausgehend von dem Problem der Suche eines beliebigen Weges zwischen zwei Knoten in einem Graphen wird der Backtracking-Algorithmus zur Tiefensuche erarbeitet. Durch Wegfall der Abbruchbedingung „Zielknoten gefunden“ lassen sich mit dem Algorithmus **alle** Wege zwischen Start und Zielknoten finden. Als Alternative wird der Algorithmus zur Breitensuche erarbeitet, der als Ergebnis eine Liste aller Knoten, die auf dem Weg vom Start- zum Zielknoten gefunden wurde, zurückgibt. Ausgehend vom Zielknoten kann durch Vorgängersuch in dieser Liste ein Weg

	• wenden didaktisch orientierte Entwicklungsumgebungen zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I), • stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D), • nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zur Erschließung, Aufbereitung und Präsentation fachlicher Inhalte (D), • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Daten, zur Organisation von Arbeitsabläufen sowie zur Verteilung und Zusammenführung von Arbeitsanteilen (K).	vom Start- zum Zielknoten gefunden werden. Damit wird das Verfahren beim Dijkstra-Algorithmus vorbereitet. *Beispiel*: Kürzeste Wege Ausgehend vom Backtracking-Algorithmus zur Bestimmung aller Wege von einem Start- zu einem Zielknoten in einem Graphen wird ein Algorithmus zur Bestimmung des kürzesten Weges erarbeitet. Aufwandbetrachtungen führen zu der Frage nach einem effizienteren Algorithmus. Der Dijkstra-Algorithmus kann durch geschickte Aufgabenstellung, ggf. unter Einbeziehung des in den Materialien enthaltenen Programms *GraphTool* von der Lerngruppe selbstständig erarbeitet und auf mehrere Beispiele angewandt werden. Die Implementierung erfolgt in der Lerngruppe arbeitsteilig unter Vorgabe einer Benutzungsoberfläche. Der Vergleich der beiden Algorithmen unter Effizienzaspekten ist Bestandteil des Unterrichts. *Beispiel:* Versorgungsnetz Die Problemstellung, Verbraucher eines Dorfes möglichst kostengünstig an ein Versorgungsnetz (Kabel, Gas, Telefon) anzuschließen, motiviert die Behandlung des minimalen Spannbaumes eines Graphen.

		Die Definition eines Baumes und eines Spannbaumes als Spezialfälle von Graphen bereiten die nächste Unterrichtssequenz vor. *Materialien*: Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben LK-Q1.V (Download LK-Q1.V.2)
3. Die Datenstruktur Binärbaum als Spezialfall eines Graphen im Anwendungskontext unter Nutzung der Klasse `BinaryTree<ContentType>` (a) Definition eines Binärbaums und grundlegende Begriffe (b) Erarbeitung der Klasse `BinaryTree<ContentType>` und beispielhafte Anwendung der Operationen (c) Implementierung der Traversierung eines Binärbaums im Pre-, In- und Postorderdurchlauf (d) Modellierung und Implementierung einer Anwendung unter Verwendung der Datenstruktur Binärbaum		*Beispiel*: MorseBaum Morse hat Buchstaben als Folge von Punkten und Strichen codiert. Diese Codierungen können in einem Binärbaum dargestellt werden, sodass ein Übergang zum linken Teilbaum einem Punkt und ein Übergang zum rechten Teilbaum einem Strich entspricht. Anhand dieses Beispiels werden die Definition eines Binärbaums als Spezialfall eines Graphen und eine rekursive Definition erarbeitet. Die Methoden der generischen Klasse `BinaryTree<ContentType>` eingeführt und zur Implementation des Morsecodierers und -dekodierers genutzt. Wenn man bei der Wurzel startet und durch Übergänge zu linken oder rechten Teilbäumen einen Pfad zum gewünschten Buchstaben sucht, erhält man den Morsecode des Buchstabens. Im Leistungskurs wird auch ein rekursiver Algorithmus zur Dekodierung von Morsezeichen entwickelt.

Beispiel: Termbaum
Arithmetische Ausdrücke werden in einem Binärbaum dargestellt. Mit den Traversierungsalgorithmen lassen sich die Terme in Pre, Post- und In-Order-Darstellung ausgeben. Gegebenenfalls kann ein Interpreter für einen arithmetischen Term mit den vier Grundrechenarten entwickelt werden. Im Unterrichtsvorhaben (Q2-III) kann das Beispiel unter dem Thema „Parsen eines einfachen Terms und die Erzeugung eines Termbaums“ wieder aufgegriffen werden.

Beispiel: Infomatikerbaum als Binärbaum
In einem *binären Baum* werden die Namen und die Geburtsdaten von Informatikern lexikographisch geordnet abgespeichert. Alle Namen, die nach dieser Ordnung vor dem Namen im aktuellen Teilbaum stehen, sind in dessen linkem Teilbaum, alle die nach dem Namen im aktuellen Teilbaum stehen, sind in dessen rechtem Teilbaum. (Dies gilt für alle Teilbäume.)
Folgende Funktionalitäten werden benötigt:

- Einfügen der Informatiker-Daten in den Baum
- Suchen nach einem Informatiker über den Schlüssel Name
- Ausgabe des kompletten Datenbestandes in nach Namen sortierter Reihenfolge

Anhand des Beispiels werden die Eigenschaften und die Methoden eines binären Suchbaums entwickelt und implementiert.

		Materialien: Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben LK-Q1.V (Download LK-Q1-V.3)
4. Erarbeitung, Implementierung und Verwendung der Datenstruktur binärer Suchbaum im Anwendungskontext (a) Erarbeitung der Eigenschaften eines binären Suchbaums im Anwendungskontext (b) Erarbeitung der Attribute und Methoden der generischen Klasse `BinarySearchTree<ContentType>` und des Interfaces `ComparableContent` (c) Implementierung des Konstruktors und der Methode `search` der Klasse `BinarySearchTree<ContentType>` (d) Implementierung eines Anwendungsbeispiels einschließlich der sortierten Ausgabe eines binären Suchbaumes		*Beispiel*: Informatikerbaum binärer Suchbaum Das Beispiel wird wieder aufgegriffen und diesmal mit der Klasse `BinarySearchTree<ContentType>` implementiert. Durch Modifikation der implementierten Methoden des Interfaces `ComparableContent` wird der Suchbaum nach den Geburtsdaten der Informatiker sortiert. *Beispiel*: Buchindex Als weiteres Anwendungsbeispiel, das mehrere dynamische Datenstrukturen miteinander verknüpft, soll ein Programm modelliert und implementiert werden, das das Stichwortregister eines Buches verwaltet. Die Wörter werden in einem binären Suchbaum verwaltet, die zugehörigen Seitenzahlen als lineare Listen. *Materialien*: Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben LK-Q1.VI (Download LK-Q1.V.4)

Unterrichtsvorhaben Q1-VI

Thema: Projektorientierte Softwareentwicklung am Beispiel einer Anwendung mit Datenbankanbindung

Leitfrage: *Wie kann ein komplexeres Java-Projekt unter Verwendung einer Datenbank realisiert werden?*

Vorhabenbezogene Konkretisierung:
Der Schwerpunkt des vorliegenden Unterrichtsvorhabens liegt in der schülerorientierten Erarbeitung eines Java-Projekts, das einer Verknüpfung zwischen der Modellierung und Abfrage von Datenbanken sowie der Entwicklung von Problemlösungen mit Hilfe dynamischer Datenstrukturen wie zum Beispiel der linearen Liste oder dem Graphen herstellt. Dabei soll ein möglichst vollständiger Softwareentwicklungszyklus durchlaufen und sowohl die Arbeit mit Datenbanken als auch mit dynamischen Datenstrukturen vertieft bzw. geübt werden.
Dazu wird zunächst durch die Schülerinnen und Schüler eine lebensweltnahe Problemstellung entwickelt, die sich auch direkt aus den Anforderungen des Schulalltags ergeben und zu einem real einsetzbaren Softwareprodukt führen kann. Dabei könnte es sich zum Beispiel um ein Ausleihsystem für die Schülerbibliothek, eine Datenbank für Fehlstunden oder die Verwaltung von Ergebnissen vom Sportfest handeln. Da es nicht immer einfach ist, eine so praxisorientierte Problemstellung zu finden, sind aber auch andere Projekte denkbar.
Die Wahl der Problemstellung sollte je nach Lerngruppe einen Schwerpunkt auf die Datenbankmodellierung und Abfrage oder aber auf die Arbeit mit dynamischen Datenstrukturen legen. Soll sich das Projekt auf Datenbanken konzentrieren, ist z.B. ein Quizspiel denkbar, das schulweit Fragen aus einer zentralen Datenbank abruft, Ergebnisse und Ranglisten aller Spielerinnen und Spieler verwaltet und ggf. diese auch gegeneinander antreten lässt, indem ihnen die gleichen Fragen gestellt werden und sie somit in einen direkten Vergleich treten. Will man den Schwerpunkt des Projekts auf dynamische Datenstrukturen legen, wäre anknüpfend an das Unterrichtsvorhaben „Q1-V: Graphen“ die Entwicklung eines Routenplaners, zum Beispiel basierend auf einer Datenbank mit echten Daten des deutschen Autobahnnetzes, ein geeignetes Projekt.

Der Aufbau dieses Projekts orientiert sich an einer vereinfachten Version des Wasserfallmodells der Softwareentwicklung, bestehend aus Analyse, Modellierung, Implementierung und Test (und ggf. auch Installation und Schulung). Insbesondere die Modellierung und Implementierung beziehen sich dabei gegebenenfalls auf die Entwicklung einer geeigneten Datenbank und deren Abfrage und Manipulation mit SQL und auf die Entwicklung eines entsprechenden Java-Programms, das die Datenbank nutzt. Bei Projekten, die Datenbanken mit einer großen Anzahl von Datensätzen erfordern, sollten die Datensätze in geeigneter elektronischer Form vorgegeben werden. Zur Abfrage und Manipulation der Daten kommen didaktisch vereinfachte Klassen zur Einbindung einer Datenbank in ein Java-Programm zum Einsatz (siehe Abiturklassen zu Datenbanken).

Um den normalerweise hohen Zeitbedarf für Projektarbeiten möglichst gering zu halten, sollte arbeitsteilig vorgegangen werden und auch auf das Prinzip des Prototypings, d.h. die Vervollständigung eines vom Lehrenden vorgegebenen Teilprogramms, zurückgegriffen werden. So sollte zum Beispiel die zeitaufwändige, aber wenig ergiebige Implementation einer grafischen Benutzungsoberfläche – nicht jedoch deren Design – den Schülerinnen und Schülern abgenommen werden.

Zeitbedarf: 15 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
4. Analyse einer lebensweltnahen Problemstellung im Hinblick auf die Entwicklung eines Java-Programms mit Datenbankanbindung (a) Entwicklung einer Programmidee (b) Analyse des Problembereichs (c) Entwicklung eines Anforderungskatalogs für das zu entwickelnde Programm	Die Schülerinnen und Schüler • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modellieren Klassen mit ihren Attributen, Methoden und Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare und nichtlineare Datensammlungen zu (M), • ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M), • stellen die Kommunikation zwischen Objekten grafisch dar (D), • stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen (D),	*Beispiel*: Quizspiel Verwaltung von Quizfragen, Antworten und Ranglisten *Beispiel*: Navigationssystem Ermittlung von kürzesten Wegen im deutschen Autobahnnetz *Beispiel*: Verwaltung der Schülerbücherei Verwaltungsprogramm für das Einpflegen und Ausleihe von Büchern der Schülerbibliothek *Beispiel*: Fehlstundenverwaltung Verwaltungsprogramm für die Fehlstunden von Schülerinnen und Schülern *Beispiel*: Sportfestverwaltung Verwaltung von Aufgaben, Sportereignissen und Ergebnissen des Schulsportfestes *Beispiel*: Materialverwaltung

	• analysieren und erläutern objektorientierte Modellierungen (A), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), • ermitteln für anwendungsbezogene Problemstellungen Entitäten, zugehörige Attribute, Relationen und Kardinalitäten (M), • stellen Entitäten mit ihren Attributen und die Beziehungen zwischen Entitäten mit Kardinalitäten in einem Entity-Relationship-Diagramm grafisch dar (D), • modellieren zu einem Entity-Relationship-Diagramm ein relationales Datenbankschema (M), • bestimmen Primär- und Sekundärschlüssel (M), • implementieren ein relationales Datenbankschema als Datenbank (I), • analysieren und erläutern eine Datenbankmodellierung (A), • erläutern die Eigenschaften normalisierter Datenbankschemata (A), • überprüfen Datenbankschemata auf vorgegebene Normalisierungseigenschaften (D), • überführen Datenbankschemata in die 1. bis 3. Normalform (M), • analysieren und erläutern Algorithmen und Programme (A),	Materialien für Vertretungsstunden sollen verwaltet werden.
5. Modellierung einer datenbankgestützten Problemlösung unter Berücksichtigung des MVC-Prinzips (a) Strukturierung nach dem MVC-Prinzip (b) Modellierung der Datenbank (ER-Diagramm, Datenbankschema) (c) Modellierung der Kontrollklassen und Entwicklung von SQL-Anweisungen entsprechend des Anforderungskatalogs (d) Modellierung einer grafischen Benutzungsoberfläche		*Materialien:* Ergänzungsmaterialien zum Lehrplannavigator – Dokumentation der Abiturklassen zur Datenbankanbindung (Download LK-Q1-VI.1) Ergänzungsmaterialien zum Lehrplannavigator – Projekt Navigationssystem (Download LK-Q1-VI.2) Ergänzungsmaterialien zum Lehrplannavigator – Projekt Quizspiel (Download LK-Q1-VI.3)
6. Umsetzung des Softwareprojektes (a) Umsetzung der Datenbank in einem Datenbankmanagementsystem (b) Implementierung der Kontrollklassen mit Anbindung an die Datenbank unter Verwendung didaktischer Klassen (c) Integration in den Prototypen der Benutzeroberfläche		*Materialien:* Ergänzungsmaterialien zum Lehrplannavigator – Dokumentation der Abiturklassen zur Datenbankanbindung (Download LK-Q1-VI.1) Ergänzungsmaterialien zum Lehrplannavigator – Projekt Navigationssystem (Download LK-Q1-VI.2) Ergänzungsmaterialien zum Lehrplannavigator – Projekt Quizspiel (Download LK-Q1-VI.3)

7. *Optional*: Einführung und Evaluation (a) Installation und Test des Endprodukts im konkreten Anwendungskontext (b) Schulung von Anwendern an der neuen Software (c) Evaluation des Projekts	• modifizieren Algorithmen und Programme (I), • testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I), • ermitteln Ergebnisse von Datenbankabfragen über mehrere verknüpfte Tabellen (D), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • analysieren und erläutern die Syntax und Semantik einer Datenbankabfrage (A), • verwenden die Syntax und Semantik einer Datenbankabfragesprache, um Informationen aus einem Datenbanksystem zu extrahieren (I),nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Daten, zur Organisation von Arbeitsabläufen sowie zur Verteilung und Zusammenführung von Arbeitsanteilen (K), • wenden didaktisch orientierte Entwicklungsumgebungen zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I), • entwickeln mit didaktisch orientierten Entwicklungsumgebungen einfache	-

	Benutzungsoberflächen zur Kommunikation mit einem Informatiksystem (M), • erläutern Eigenschaften und Aufbau von Datenbanksystemen unter dem Aspekt der sicheren Nutzung (A), • untersuchen und bewerten anhand von Fallbeispielen Auswirkungen des Einsatzes von Informatiksystemen sowie Aspekte der Sicherheit von Informatiksystemen, des Datenschutzes und des Urheberrechts (A).	

Unterrichtsvorhaben Q2-I:

Thema: Sicherheit und Datenschutz in Informatiksystemen sowie Grenzen und Auswirkungen der Automatisierung

Leitfragen: Welche moralische und rechtliche Verantwortung tragen Informatikerinnen und Informatiker hinsichtlich des Datenschutzes, des Urheberrechts und der gesellschaftlichen Auswirkungen informatischer Systeme? Wie kann Datensicherheit z. B. mit Hilfe von Verschlüsselungsverfahren gewährleistet werden? Wo liegen die Grenzen der Automatisierbarkeit?

Vorhabenbezogene Konkretisierung:

Die Schülerinnen und Schüler werden mit Hilfe eines einprägsamen Fallbeispiels für die Problembereiche des Datenschutzes, des Urheberrechts und der moralischen Verantwortung von Informatikerinnen und Informatikern sensibilisiert. Ein mögliches Fallbeispiel stellt das selbstfahrende Automobil dar, das aus der Perspektive unterschiedlicher Interessensgruppen zu betrachten ist. Im Bereich des Datenschutzes könnte das Erstellen von Bewegungsprofilen thematisiert werden, im Bereich des Urheberrechts, in wieweit Quellcode zur Steuerung des Fahrzeugs geschützt und ggf. sogar nicht einsehbar sein darf, obwohl das Leben von Menschen von seiner fehlerfreien Funktion abhängt. Bezogen auf eine moralische Dimension könnte thematisiert werden, wie ein solches Fahrzeug bei Unfällen reagieren sollte, bei denen ein Personenschaden nicht abzuwenden ist, d.h. z.B. entschieden werden muss, ob ein Ausweichmanöver zur Rettung von Passanten gefahren werden soll, obwohl dabei der Insasse des Fahrzeugs in Lebensgefahr gebracht wird. Auch allgemeine gesellschaftliche Auswirkungen können thematisiert werden, wie z. B. die Konsequenzen für den Arbeitsmarkt, wenn selbstfahrende Autos, LKWs und Züge die Norm werden. Schülerinnen und Schüler sollen dieses Fallbeispiel zunächst hinsichtlich dieser Problemstellungen analysieren und erste Lösungsansätze auf dem Hintergrund ihres Vorwissens erarbeiten.

Anschließend werden die Themen Datenschutz, Urheberrecht und moralische Verantwortung systematisiert und vertieft. Im Bereich Datenschutz werden grundlegende Begriffe (z. B. personenbezogene Daten, informationelle Selbstbestimmung, Datensparsamkeit usw.) eingeführt und an weiteren Fallbeispielen verdeutlicht. Im Bereich Urheberrecht sollte mindestens ein verbreitetes Lizenzsystem thematisiert werden (z. B. Creative-Commons-Lizenzen) und anhand von Beispielen verdeutlicht werden. Im Bereich der moralischen Verantwortung sollte ein Bewertungsmaßstab für moralische Fragen erarbeitet werden (z. B. Grundidee des klassischen Handlungsutilitarismus nach Jeremy Bentham, Grundidee der Verantwortungsethik nach Hans Jonas). Auch diese Positionen werden auf weitere Beispiele angewendet. Die Erarbeitung der Schwerpunkte Datenschutz, Urheberrecht und moralische Verantwortung kann dabei sequenziell mit der gesamten Lerngruppe oder parallel in zieldifferenten Teilgruppen erfolgen. In beiden Fällen müssen die Ergebnisse zusammengefasst in der gesamten Lerngruppe gesichert werden.

Abschließend wird das Eingangsproblem, in diesem Fall das selbstfahrende Automobil, auf Grundlage der neu erarbeiteten Positionen abschließend bewertet. In diesem Zusammenhang wird insbesondere das Verfassen einer Stellungnahme im Sinne einer reflektierten Darstellung der eigenen Position eingeübt. Eine allgemeingültige und unbestreitbare Bewertung ist aufgrund der Ambivalenz der Beispiele nicht möglich. Der Blick auf die gesamtgesellschaftlichen Konsequenzen selbstfahrender Automobile führt zu der Frage, welche Grenzen der Automatisierung allgemein gesetzt sind.

Nachdem der erste Teil des Unterrichtsvorhabens die Notwendigkeit verdeutlicht hat, insbesondere personenbezogene Daten zu schützen, wird im zweiten Teil des Vorhabens dieses Problem aufgegriffen. Dabei werden die grundlegenden Sicherheitsziele *Vertraulichkeit*, *Integrität* und *Verfügbarkeit* und weitere Sicherheitsziele eingeführt und an Fallbeispielen diskutiert und gegenübergestellt. Anschließend werden symmetrische und asymmetrische Verschlüsselungsverfahren zur Umsetzung von Sicherheitszielen eingeführt. Dabei werden die grundlegenden Verfahren dieser Verschlüsselungen und die Möglichkeiten eines Angriffs auf sie in den Mittelpunkt gestellt. Im Kontext der symmetrischen Verschlüsselungsverfahren sollte das Problem des Schlüsselaustausches angesprochen werden. Im Anschluss an asymmetrische Verschlüsselungsverfahren wird das Prinzip des Signierens thematisiert.

Zeitbedarf: 20 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Einführung in die Problemfelder Datenschutz, Urheberrecht und moralische Verantwortung a. Vorstellung eines komplexen Fallbeispiels b. Erarbeitung von Interesse verschiedener Interessensgruppen im Hinblick auf die Problemfelder c. Erster Bewertungsversuch auf Grundlage des Vorwissens von Schülerinnen und Schülern	Die Schülerinnen und Schüler • untersuchen und bewerten anhand von Fallbeispielen Auswirkungen des Einsatzes von Informatiksystemen sowie Aspekte der Sicherheit von Informatiksystemen, des Datenschutzes und des Urheberrechts (A). • untersuchen und bewerten Problemlagen, die sich aus dem Einsatz von Informatiksystemen ergeben hinsichtlich rechtlicher	*Beispiel: autonomes Fahren* Ein namhaftes Softwareunternehmen möchte den Automobilmarkt mit selbstfahrenden Autos revolutionieren. Der Quellcode zu Steuerung der Fahrzeuge ist Firmengeheimnis, zur Verbesserung der Fahrleistung werden Bewegungsprofile erstellt und noch ist nicht klar, wie das Fahrzeug bei drohenden Unfällen mit Personenschaden reagieren soll. Eine erfolgreiche Einführung der Technologie würde den Straßenverkehr sicherer, schneller und ökologischer machen, allerdings

	Vorgaben, ethischer Aspekte und gesellschaftlicher Werte unter Berücksichtigung unterschiedlicher Interessenlagen (A). • analysieren und erläutern Eigenschaften, Funktionsweisen und Einsatzbereiche symmetrischer und asymmetrischer Verschlüsselungsverfahren (A). • untersuchen und beurteilen Grenzen des Problemlösens mit Informatiksystemen (A). • nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zur Erschließung, Aufbereitung und Präsentation fachlicher Inhalte (D).	auch zu tausenden von Arbeitslosen durch Wegfall ganzer Berufszweige führen. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.I – Fallbeispiel „autonomes Fahren“ (Download LK-Q2-I.1)
2. Erarbeitung grundlegender Positionen (ggf. in zieldifferenten Gruppen) a. Erarbeitung von Grundideen des Datenschutzes (z.B. personenbezogene Daten, informationelle Selbstbestimmung, Datensparsamkeit usw.) b. Erarbeitung von Grundideen des Urheberrechts anhand eines verbreiteten Lizenzsystems c. Erarbeitung eines einfachen moralischen Bewertungsmaßstabes d. Anwendung auf Fallbeispiele		*Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.I – Grundlagen „Informatik, Mensch und Gesellschaft“ (Download LK-Q2-I.2) *Beispiel: Creative-Commons-Lizenzen* URL: http://de.creativecommons.org (Abgerufen am 19.06.2016)
3. Zusammenfassung und Sicherung a. Darstellung der relevanten Aspekte zum Datenschutz, Urheberrecht und zur moralischen Bewertung b. Demonstration an jeweils einem einfachen Beispiel		*Beispiel: Stellwände* Eine Zusammenfassung und Sicherung kann in Form von Plakatstellwänden erfolgen. Diese können auch im Sinne einer Dauerausstellung im Schulgebäude präsentiert werden.

Anmerkung: Dieser Schritt ist entscheidend, wenn in die Punkte 2 bis 4 arbeitsteilig umgesetzt wurden.		
4. Stellungnahme zu einem komplexen Fallbeispiel a. Reflexion des Einstiegsbeispiels des Unterrichtsvorhabens auf Grundlage der erarbeiteten Positionen zum Datenschutz, Urheberrecht und zur moralischen Bewertung b. Einführung in das Verfassen von Stellungnahmen c. Erarbeitung von Stellungnahmen zum Einstiegsbeispiel d. Diskussion und Würdigung von Stellungnahmen mit Blick auf die Grenzen der Automatisierbarkeit		*Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.I – Materialblatt „Stellungnahme“ (Download LK-Q2-I.3)
5. Sicherheitsziele und der kryptographische Ansatz a. Beschreibung und Gegenüberstellung von Sicherheitszielen anhand von Fallbeispielen b. Problematisierung von Sicherheitszielen in Bezug auf die Kommunikation über offene Kanäle c. Einführung in die Kryptologie – Erreichen von Sicherheitszielen durch		*Beispiel:* Sicherheitsziele *Zugriffskontrolle* und *Authentifikation* anhand des „Keyless Entry-System“ bei Autos *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.I – Sicherheitsziele und Kommunikationsrollenspiel (Download LK-Q2.I.4)

Verschlüsselung		
6. Verschlüsselungsverfahren und deren Sicherheit a. Monoalphabetische Verschlüsselungsverfahren und Angriffsmöglichkeiten auf diese Verfahren b. Polyalphabetische Verschlüsselungsverfahren und Angriffsmöglichkeiten auf diese Verfahren c. Schlüsseltausch als Grundproblem symmetrischer Verschlüsselungsverfahren (Diffie-Hellman) d. Das Schlüsselpaar und die Einwegfunktion als zentrale Konzepte der asymmetrischen Verschlüsselung e. Probleme der asymmetrischen Verschlüsselung f. Signaturen als Anwendungsgebiet asymmetrischer Kryptographie		*Beispiel: Caesar-Verschlüsselung* Die Cäsarchiffre als Beispiel für ein monoalphabetisches Verfahren und die Exhaustionsmethode (Brute-Force-Methode) zur Verdeutlichung der Bedeutsamkeit eines starken Schlüssels *Beispiel: Vigenère-Verschlüsselung* Die Vigenère-Chiffre als Verbesserung einer monoalphabetischen Verschlüsselung und der Kasiski-Test als Grundlage eines verbesserten Angriffsverfahrens *Beispiel: One-Time-Pad* Der One-Time-Pad als Spezialfall eines sicheren polyalphabetischen Verschlüsselungsverfahrens *Beispiel:* Das RSA-Verfahren als Beispiel für eine asymmetrische Verschlüsselung mit einer Einwegfunktion *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator – Verschlüsselungsverfahren (Download LK-Q2.I.5) *Materialien:* URL*:* *https://www.cryptool.org/de/* *(Abgerufen am 19.06.2016)*

Unterrichtsvorhaben Q2-II

Thema: Grundlagen der Netzwerkkommunikation sowie Modellierung und Implementierung von Client-Server-Anwendungen in kontextbezogenen Problemstellungen

Leitfragen*: Wie werden Daten in Netzen übermittelt? Wie entwickelt man ein Client-Server-System im Anwendungskontext? Welche Algorithmen sind zu implementieren?*

Vorhabenbezogene Konkretisierung:
Zunächst werden die Grundlagen von Datenübertragung in Netzwerken erarbeitet. Den Einstieg bildet ein Vergleich der Kommunikation in Netzen mit der physikalischen Zustellung von Sendungen durch Postunternehmen, der zu einem Schichtenmodell als Strukturierungsprinzip für Netzwerkkommunikation führt. Im Anschluss werden von den Schülerinnen und Schülern Antworten auf grundsätzliche Herausforderungen im Bereich Netzwerkkommunikation erarbeitet: die Wahl einer geeigneten Codierung, Vor- und Nachteile verschiedener Topologien, Adressierung/Routing in IP-Netzen sowie die Gestaltung von Protokollen für die Anwendungsebene.

In einer zweiten Phase werden zunächst Clients für vorhandene Server-Dienste entwickelt. Darauf aufbauend können anschließend eigene Server modelliert und implementiert werden.

In einer dritten Phase modellieren und implementieren die die Schülerinnen und Schülern schließlich ein Client-Server-System. Dieses macht u.a. ein Verständnis von Nebenläufigkeit notwendig, da ein Server parallel Nachrichten von mehreren Clients empfangen und verarbeiten können muss.

Zeitbedarf: 20 Stunden

Sequenzierung des Unterrichtsvorhabens

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Grundlagen den Datenübertragung in Netzwerken (a) Schichtenmodell • Erarbeitung eines standardisierten Schichtenmodells für Netzwerkkommunikation (b) Grundlagen der Codierung • Erarbeitung einer eigenen, vereinfachten Codierung für die Bitübertragung (c) Topologien • Erarbeitung und Vergleich ausgewählter Netzwerktopologien (d) Routing • Analyse von Grundlagen der Adressierung in IP-Netzwerken (e) Protokolle • Erarbeitung des beispielhaften Aufbaus eines Protokoll auf der Anwendungsschicht	Die Schülerinnen und Schüler • beschreiben und erläutern Netzwerk-Topologien, die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken (A), • analysieren und erläutern Algorithmen und Programme (A), • analysieren und erläutern Protokolle zur Kommunikation in einem Client-Server-Netzwerk (A), • entwickeln und erweitern Protokolle zur Kommunikation in einem Client-Server-Netzwerk (M), • modifizieren Algorithmen und Programme (I) • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihr Operationen und Beziehungen (M), • modellieren Klassen mit ihren Attributen, Methoden und Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare und nichtlineare Datensammlungen zu (M),	*Material:* Aufgabensammlung Anhand einer Sammlung von Aufgabenblättern (teils inkl. implementierter Begleitwerkzeuge) erarbeiten die Schülerinnen und Schüler im Anwendungskontext Grundlagen der Inhaltspunkte „OSI-Referenzmodell“, „Codierung“, „Topologien“, „Routing“ und „Protokolle“
2. Analyse, Modellierung und Implementierung von Netzwerkanwendungen in Client-Server-Struktur (a) Nutzung einfacher Server-Dienste mittels Client		*Beispiel:* Echo- bzw. Daytime-Clients und –Server sowie eigene Erweiterungen Anhand der Echo- und Daytime-Dienste (z.B. lokal im Schulnetz durch den Lehrenden zur

• Modellierung und Implementierung von Clients für einfach Serverdienste (b) Anbieten von Diensten mittels Server • Analyse vorgegebener Implementationen einfacher Server • Modellierung und Implementierung eigener Server	• analysieren und erläutern objektorientierte Modellierungen (A), • stellen Klassen und ihre Beziehungen grafisch dar (D), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I).	Verfügung gestellt) erarbeiten die Schülerinnen und Schüler zunächst den die Funktionsweise bzw. den Aufbau einfacher Clients und verwenden dabei zunächst die Klasse Connection, später die Klasse Client. In einem zweiten Schritt implementieren die Schülerinnen und Schüler Daytime- und Echo-Client bzw. Erweiterungen/Abwandlungen derselben (ggf. mit Steigerung des Interaktionsgrades) selbst.
3. Entwicklung eines vollständigen Client-Server-Systems • Protokollentwurf, Dialogorientierung • Modellierung mittels Entwurfs- und Implementationsdiagramm • Bedeutung von Nebenläufigkeit • Implementierung		*Beispiel:* Messenger-Dienst Abschließend entwickeln die Schülerinnen und Schüler ein Client-Server-System zum Versenden von Nachrichten zwischen einzelnen Rechnern (einfacher Messenger), basierend auf selbst gewählten „Nicknames“.

Unterrichtsvorhaben Q2-III:

Thema: Grundlagen von Automaten und formalen Sprachen sowie die Modellierung und Implementierung eines Parsers zu einer formalen Sprache

Leitfragen: *Wie kann man (endliche) Automaten genau beschreiben? Wie können endliche Automaten und Kellerautomaten (in alltäglichen Kontexten oder zu informatischen Problemstellungen) modelliert werden? Wie können Sprachen durch Grammatiken beschrieben werden? Welche Zusammenhänge gibt es zwischen formalen Sprachen, Automaten und Grammatiken? Wie kann ein Parser für eine einfache formale Sprache entwickelt werden?*

Vorhabenbezogene Konkretisierung:
Anhand kontextbezogener Beispiele werden endliche Automaten entwickelt, untersucht und modifiziert. Dabei werden verschiedene Darstellungsformen für endliche Automaten ineinander überführt und die akzeptierten Sprachen endlicher Automaten ermittelt. An einem Beispiel wird ein nichtdeterministischer Akzeptor als Alternative zu einem entsprechenden deterministischen Akzeptor eingeführt. Auch die Umwandlung eines nichtdeterministischen Automaten in einen deterministischen Automaten wird thematisiert.

Anhand kontextbezogener Beispiele werden Grammatiken regulärer Sprachen entwickelt, untersucht und modifiziert. Der Zusammenhang zwischen regulären Grammatiken und endlichen Automaten wird verdeutlicht durch die Entwicklung von allgemeinen Verfahren zur Erstellung einer regulären Grammatik für die Sprache eines gegebenen endlichen Automaten bzw. zur Entwicklung eines endlichen Automaten, der genau die Sprache einer gegebenen regulären Grammatik akzeptiert. Zu einer einfachen regulären Sprache wird ein Parser in Form eines Java-Programms entwickelt.

Auch nicht-reguläre Grammatiken werden untersucht, entwickelt oder modifiziert. An einem Beispiel werden die Grenzen endlicher Automaten ausgelotet. Mit Blick auf diese Einschränkungen endlicher Automaten wird die Idee eines Automaten mit Speicher thematisiert und zu einem Kellerautomaten weiterentwickelt.

Zeitbedarf: 30 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien oder Materialien
4. Endliche Automaten (e) Erarbeitung der formalen Beschreibung eines endlichen Automaten auf der Grundlage von Automaten in bekannten Kontexten (f) Untersuchung, Darstellung und Entwicklung endlicher Automaten (g) Umwandlung nichtdeterministischer endlicher Automaten in deterministische endliche Automaten	Die Schülerinnen und Schüler • analysieren und erläutern die Eigenschaften endlicher Automaten und Kellerautomaten einschließlich ihres Verhaltens auf bestimmte Eingaben (A), • analysieren und erläutern Grammatiken regulärer und kontextfreier Sprachen (A), • erläutern die Grenzen endlicher Automaten und regulärer Sprachen im Anwendungszusammenhang (A), • ermitteln die formale Sprache, die durch eine Grammatik erzeugt wird (A), • entwickeln und modifizieren zu einer Problemstellung endliche Automaten oder Kellerautomaten (M), • entwickeln zur akzeptierten Sprache eines Automaten die zugehörige Grammatik (M), • entwickeln zur Grammatik einer regulären oder kontextfreien Sprache einen zugehörigen endlichen Automaten oder einen Kellerautomaten (M), • modifizieren Grammatiken regulärer und kontextfreier Sprachen (M), • entwickeln zu einer regulären oder kontextfreien Sprache eine Grammatik, die die Sprache erzeugt (M), • stellen endliche Automaten in Tabellen oder Graphen dar und überführen sie in die jeweils andere Darstellungsform (D),	*Beispiele:* Cola-Automat, Geldspielautomat, Roboter, Zustandsänderung eines Objekts „Auto“, Akzeptor für bestimmte Zahlen, Akzeptor für Teilwörter in längeren Zeichenketten, Akzeptor für Terme *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.III (Download LK-Q2-III.1)
5. Untersuchung und Entwicklung von Grammatiken regulärer Sprachen (a) Erarbeitung der formalen Darstellung regulärer Grammatiken (b) Untersuchung, Modifikation und Entwicklung von Grammatiken (c) Entwicklung von endlichen Automaten zum Erkennen regulärer Sprachen die durch Grammatiken gegeben werden (d) Entwicklung regulärer Grammatiken zu endlichen Automaten (e) Entwicklung eines Parsers für eine einfache reguläre Sprache		*Beispiele**:*** reguläre Grammatik für Wörter mit ungerader Parität, Grammatik für Wörter, die bestimmte Zahlen repräsentieren, Satzgliederungsgrammatik *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.III (Download LK-Q2-III.2)

	• ermitteln die Sprache, die ein endlicher Automat oder ein Kellerautomat akzeptiert (D), • beschreiben an Beispielen den Zusammenhang zwischen Automaten und Grammatiken (D), entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“, „Teilen und Herrschen“ und „Backtracking“ (M).	
6. Grenzen endlicher Automaten		*Beispiele:* Klammerausdrücke, $a^n b^n$ im Vergleich zu $(ab)^n$ *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.III (Download LK-Q2-III.3)
7. Entwicklung eines Kellerautomaten als Antwort auf die Grenzen endlicher Automaten (a) Erweiterung eines DEA um eine einzelne Speichervariable zum Zählen von Eingabezeichen (z.B. Klammern) und Problematisierung dieses Ansatzes (b) Entwicklung eines Automaten mit Kellerspeicher (c) Anwendung eines Kellerautomaten zur Syntaxüberprüfung auf Grundlage von nicht-regulären Grammatiken (d) Implementierung eines Kellerautomaten zur Syntaxüberprüfung (Backtracking)		*Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.III (Download LK-Q2-III.4)

Unterrichtsvorhaben Q2-IV:

Thema: Prinzipielle Arbeitsweise eines Computers sowie Modellierung und Implementierung eines Scanners, Parsers und Interpreters für eine einfache maschinennahe Programmiersprache

Leitfragen: *Was sind die strukturellen Hauptbestandteile eines Computers und wie kann man sich die Ausführung eines maschinenahen Programms mit diesen Komponenten vorstellen? Wie werden Programme aus höheren Programmiersprachen für den Computer verständlich und wie werden sie in eine tiefere Sprachebene übersetzt und interpretiert?*

Vorhabenbezogene Konkretisierung:
Anhand einer von-Neumann-Architektur und einem maschinennahen Programm wird die prinzipielle Arbeitsweise von Computern verdeutlicht. Grundlegenden Begrifflichkeiten bei der maschinellen Übersetzung von einer Hochsprache in eine maschinenverständliche Sprache werden definiert, veranschaulicht und zum Vorwissen aus dem Unterrichtsvorhaben Q2-III in Beziehung gesetzt.
Ausgehend von einer einfachen formalen Sprache (z. B. eine Konstruktionssprache für geometrische Figuren) werden die Bestandteile eines Compilers dargestellt:
Der Scanner eines Compilers wird in Form eines endlichen Automaten modelliert und implementiert. Die Begriffe *Symboltabelle* und *Tokenliste* werden inhaltlich gefüllt.
Die dem Parser des Compilers zugrunde liegende Grammatik wird in Form einer regulären oder kontextfreien Grammatik definiert und zugehörige Parser-Methoden werden implementiert.
Zum Abschluss wird ein Interpreter-Modul entwickelt, welches die einfache formale Sprache in eine andere Sprachebene übersetzt.

Zeitbedarf: 12 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Von-Neumann-Architektur und die Ausführung maschinennaher Programme (a) prinzipieller Aufbau einer von Neumann-Architektur mit CPU, Rechenwerk, Steuerwerk, Register und Hauptspeicher (b) einige maschinennahe Befehlen und ihre Repräsentation in einem Binär-Code, der in einem Register gespeichert werden kann (c) Analyse und Erläuterung der Funktionsweise eines einfachen maschinennahen Programms	Die Schülerinnen und Schüler • erläutern die Ausführung eines einfachen maschinennahen Programms sowie die Datenspeicherung auf einer „Von-Neumann-Architektur“ (A), • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M), • verwenden bei der Modellierung geeigneter Problemstellungen Möglichkeiten der Polymorphie (M), • ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M), • stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D), • analysieren und erläutern objektorientierte Modellierungen (A),	*Beispiel:* Addition von 4 zu einer eingegeben Zahl mit einem Rechnermodell *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.IV (Download LK-Q2-IV.1)
2. Simulation der Phasen eines Compilers (a) Prozesse beim Compiler: • scannen • parsen • übersetzen/interpretieren (b) Arten von Fehlern: • lexikalischer Fehler • syntaktischer Fehler • semantischer Fehler (c) Einordnung der neuen Begriffe in den Gesamtkontext der formalen Sprachen • Automaten • Grammatiken • Sprachen		*Beispiel:* Scanner, Parser und Interpreter einer Konstruktionssprache für geometrische Figuren Anhand einer einführenden Folienpräsentation werden die Begrifflichkeiten definiert. Ein kleiner Ausschnitt einer Pseudo-Programmiersprache zur Konstruktion von Zeichnungen wird betrachtet. Anfänglich besteht der Sprachumfang aus Programmen mit lediglich einer einzigen Zuweisung. Die grundlegenden Schritte eines Compilers werden am Beispiel dieser Grammatik nachvollzogen. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.IV (Download LK-Q2-IV.2)

3. Die Schritte eines Compilers (a) Scanner: • endlicher Automat als Grundlage • Vorgabe von Symboltabelle und Tokenliste zur Verwaltung und Erkennung des Quelltextes • Erweiterung des terminalen Alphabets der zu übersetzenden formalen Sprache • Implementierung als endlicher Automat (b) Parser: • reguläre (oder wahlweise kontextfreie) Grammatik als Grundlage • Vorgabe einer Grundversion des Parsers • Erweiterung des Sprachumfangs • Implementierung der Parsermethoden für die Produktionsregeln der kontextfreien Grammatik (c) Interpreter • Vorgabe einer Grundversion des Interpreters • Erweiterung des Sprachumfangs • Implementierung	• implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), • analysieren und erläutern Algorithmen und Programme (A), • modifizieren Algorithmen und Programme (I), • entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“ und „Teilen und Herrschen“ und „Backtracking“ (M), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I), • analysieren und erläutern die Eigenschaften endlicher Automaten und Kellerautomaten einschließlich ihres Verhaltens bei bestimmten Eingaben (A), • ermitteln die Sprache, die ein endlicher Automat oder ein Kellerautomat akzeptiert (D), • entwickeln und modifizieren zu einer Problemstellung endliche Automaten oder Kellerautomaten (M), • analysieren und erläutern Grammatiken regulärer und kontextfreier Sprachen (A),	*Beispiel:* Der Scanner-Automat zur Erkennung der einzelnen Symbole wird als endlicher Automat realisiert. Mithilfe der Symboltabelle wird die Vereinfachung des Automaten deutlich gemacht und der vereinfachte Scanner-Automat wird schrittweise erweitert und implementiert. In der zweiten Phase wird die der Pseudoprogrammiersprache zugrunde liegende Grammatik analysiert. Die Überprüfung der syntaktischen Korrektheit wird in Form eines Parsers modelliert und implementiert. Dabei wird der Sprachumfang der Pseudo-Programmiersprache schrittweise erweitert. In der dritten Phase wird ein zu Teilen bereits vorbereiteter Interpreter analysiert und erweitert, welcher Programme der Pseudo-Programmiersprache zur Konstruktion von Zeichnungen in eine Grafik übersetzt. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.IV (Download LK-Q2-IV.3)

	• modifizieren Grammatiken regulärer und kontextfreier Sprachen (M), • ermitteln die formale Sprache, die durch eine Grammatik erzeugt wird (A), • entwickeln zu einer regulären oder kontextfreien Sprache eine Grammatik, die die Sprache erzeugt (M), • modellieren und implementieren Scanner, Parser und Interpreter zu einer gegebenen regulären Sprache (I), • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Daten, zur Organisation von Arbeitsabläufen sowie zur Verteilung und Zusammenführung von Arbeitsanteilen (K), • untersuchen und beurteilen Grenzen des Problemlösens mit Informatiksystemen (A).	

Unterrichtsvorhaben Q2-V:

Thema: Entwicklung eines Netzwerkspiels mit Durchführung eines vollständigen Softwareentwicklungszyklus

Leitfragen
Wie ist ein Softwareentwicklungszyklus aufgebaut? Welches Protokoll ist für die vorgegebene Funktionalität angemessen bzw. garantiert eine fehlerfreie Kommunikation? Welche Reaktionen auf Kommunikationsereignisse sind server- und clientseitig zu entwickeln? Wie können verwendete Daten mit Hilfe von Graphen verwaltet werden?

Vorhabenbezogene Konkretisierung:
Anwendungskontext ist ein von den Schülerinnen und Schülern zu entwickelndes Zweipersonen-Netzwerkspiel. Anhand des Spiels sollen die grundlegende Prinzipien und Begrifflichkeiten der Kommunikation in Netzwerken, dem Aufbau einer Client-Server-Struktur und der Datenorganisation mit Hilfe von Graphen aus den Unterrichtsvorhaben Q1-V und Q2-II festigend wiederholt werden. Grundlage für das Softwareprojekt sind die Klassen zur Netzwerkkommunikation `Connection`, `Client` und `Server`. Der Begriff des Softwareentwicklungszyklus wird thematisiert und es werden alle Phasen der Entwicklung eines Softwareprojekts über Analyse, Modellierung und Implementierung anhand des Netzwerkspiels vollzogen. Das Projekt läuft in folgenden groben Phasen ab:

1. Projekteinstieg / -planung (Spielauswahl, Zeitmanagement)
2. Projektumsetzung (informatische Analyse, Modellierung, Implementierung, Test ➔ ggf. in Zyklen)
   2.1 Analyse des Spiels mit dem Ziel einer späteren informatischen Umsetzung als Netzwerkspiel
   2.2 Modellierung und Implementierung des Spiels als Netzwerkanwendung
   2.3 Reflexion des Softwareprodukts
3. Projektreflexion (Reflexion der Projektplanung, Präsentation)

Zeitbedarf: 15 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Projekteinstieg / -planung (Spielauswahl, Zeitmanagement) **2. Projektumsetzung (informatische Analyse, Modellierung, Implementierung, Test ➔ ggf. in Zyklen)** *2.1. Analyse des Spiels mit dem Ziel einer späteren informatischen Umsetzung als Netzwerkspiel* (a) Spielen des Spiels in mehreren Kleingruppen in Form eines Rollenspiels nach vereinbarten Regeln: i. Ein Schüler oder eine Schülerin übernehmen die Rolle des Spielleiters oder der Spielleiterin. ii. Ein weiteres Gruppenmitglied protokolliert präzise die Kommunikation zwischen der Spielleiterin bzw. dem Spielleiter und den Spielerinnen und Spielern. (b) Formalisierung des Spielablaufs *2.2. Modellierung und Implementierung des Spiels als Netzwerkanwendung* (a) Entwurf eines Protokolls zur Kommunikation zwischen Spielserver und -client basierend auf den erarbeiteten Spielregeln	Die Schülerinnen und Schüler • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M), • verwenden bei der Modellierung geeigneter Problemstellungen Möglichkeiten der Polymorphie (M), • ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M), • stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D), • analysieren und erläutern objektorientierte Modellierungen (A), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), • analysieren und erläutern Algorithmen und Programme (A), • modifizieren Algorithmen und Programme (I), • entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“ und „Teilen und Herrschen“ und „Backtracking“ (M),	*Beispiel:* GraphColoringGame Zwei-Personen-Spiel, bei dem die beiden Personen abwechselnd am Zug sind. Das Spielfeld ist ein ungerichteter Graph ohne Mehrfachkanten. Die Knoten des Graphen können mit vorgegebenen Farben markiert werden. Die beiden Spielerinnen bzw. Spieler färben abwechselnd einen noch ungefärbten Knoten. Dabei muss beachtet werden, dass kein adjazenter Knoten mit derselben Farbe markiert wurde. Es verliert derjenige Spielerinn bzw. derjenige Spieler, die bzw. der unter diesen Bedingungen keinen Zug mehr machen kann. Sind alle Knoten korrekt gefärbt, endet das Spiel unentschieden. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2-V (LK) (Download LK-Q2-V.1)

(b) Entwicklung von Entwurfs- und Implementationsdiagrammen für den Spielserver (c) Implementation und Test der Spielserver-Klasse und von dieser benötigter Fachklassen (d) Modellierung, Implementierung und Test des Spielclients (e) Test der Zusammenarbeit von Spielserver und Spielclient *2.3. Reflexion des Softwareprodukts* (a) Identifikation der Stufen eines Softwareentwicklungszyklus (b) Wiederholende Darstellung der Entwicklungsschritte zum fertigen Produkt (c) Mögliche Erweiterungen: i. ggf. Implementierung einer grafischen Benutzeroberfläche (GUI) für den Client ii. ggf. Analyse, Modellierung und Implementierung alternativer Spielregeln iii. ggf. Implementierung einer KI für einen Computer-Gegenspieler . **3. Projektreflexion (Reflexion der Projektplanung, Präsentation)**	• implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I). • analysieren und erläutern Algorithmen und Methoden zur Client-Server-Kommunikation (A), • entwickeln und implementieren Algorithmen und Methoden zur Client-Server-Kommunikation (I). • beschreiben und erläutern Netzwerk-Topologien, die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken (A), • analysieren und erläutern Protokolle zur Kommunikation in einem Client-Server-Netzwerk (A), • entwickeln und erweitern Protokolle zur Kommunikation in einem Client-Server-Netzwerk (M).	

2.2 Grundsätze der fachmethodischen und fachdidaktischen Arbeit

In Absprache mit der Lehrerkonferenz sowie unter Berücksichtigung des Schulprogramms hat die Fachkonferenz Informatik des Konrad-Zuse-Gymnasiums die folgenden fachmethodischen und fachdidaktischen Grundsätze beschlossen. In diesem Zusammenhang beziehen sich die Grundsätze 1 bis 14 auf fächerübergreifende Aspekte, die auch Gegenstand der Qualitätsanalyse sind, die Grundsätze 15 bis 21 sind fachspezifisch angelegt.

Überfachliche Grundsätze:

1) Geeignete Problemstellungen zeichnen die Ziele des Unterrichts vor und bestimmen die Struktur der Lernprozesse.
2) Inhalt und Anforderungsniveau des Unterrichts entsprechen dem Leistungsvermögen der Schüler/innen.
3) Die Unterrichtsgestaltung ist auf die Ziele und Inhalte abgestimmt.
4) Medien und Arbeitsmittel sind schülernah gewählt.
5) Die Schüler/innen erreichen einen Lernzuwachs.
6) Der Unterricht fördert eine aktive Teilnahme der Schüler/innen.
7) Der Unterricht fördert die Zusammenarbeit zwischen den Schülern/innen und bietet ihnen Möglichkeiten zu eigenen Lösungen.
8) Der Unterricht berücksichtigt die individuellen Lernwege der einzelnen Schüler/innen.
9) Die Schüler/innen erhalten Gelegenheit zu selbstständiger Arbeit und werden dabei unterstützt.
10) Der Unterricht fördert strukturierte und funktionale Partner- bzw. Gruppenarbeit.
11) Der Unterricht fördert strukturierte und funktionale Arbeit im Plenum.
12) Die Lernumgebung ist vorbereitet; der Ordnungsrahmen wird eingehalten.
13) Die Lehr- und Lernzeit wird intensiv für Unterrichtszwecke genutzt.
14) Es herrscht ein positives pädagogisches Klima im Unterricht.

Fachliche Grundsätze:

15) Der Unterricht unterliegt der Wissenschaftsorientierung und ist dementsprechend eng verzahnt mit seiner Bezugswissenschaft.
16) Der Unterricht ist problemorientiert und soll von realen Problemen ausgehen und sich auf solche rückbeziehen.
17) Der Unterricht folgt dem Prinzip der Exemplarizität und soll ermöglichen, informatische Strukturen und Gesetzmäßigkeiten in den ausgewählten Problemen und Projekten zu erkennen.
18) Der Unterricht ist anschaulich sowie gegenwarts- und zukunftsorientiert und gewinnt dadurch für die Schülerinnen und Schüler an Bedeutsamkeit.
19) Der Unterricht ist handlungsorientiert, d.h. projekt- und produktorientiert angelegt.
20) Im Unterricht werden sowohl für die Schule didaktisch reduzierte als auch reale Informatiksysteme aus der Wissenschafts-, Berufs- und Lebenswelt eingesetzt.

21) Der Unterricht beinhaltet reale Begegnung mit Informatiksystemen.

2.3 Grundsätze der Leistungsbewertung und Leistungsrückmeldung

Hinweis: Sowohl die Schaffung von **Transparenz bei Bewertungen** als auch die Vergleichbarkeit von Leistungen sind das Ziel, innerhalb der gegebenen Freiräume Vereinbarungen zu Bewertungskriterien und deren Gewichtung zu treffen.

Auf der Grundlage von §13 - §16 der APO-GOSt sowie Kapitel 3 des Kernlehrplans Informatik für die gymnasiale Oberstufe hat die Fachkonferenz des Konrad-Zuse-Gymnasiums im Einklang mit dem entsprechenden schulbezogenen Konzept die nachfolgenden Grundsätze zur Leistungsbewertung und Leistungsrückmeldung beschlossen. Die nachfolgenden Absprachen stellen die Minimalanforderungen an das lerngruppenübergreifende gemeinsame Handeln der Fachgruppenmitglieder dar. Bezogen auf die einzelne Lerngruppe kommen ergänzend weitere der in den Folgeabschnitten genannten Instrumente der Leistungsüberprüfung zum Einsatz.

2.3.1 Beurteilungsbereich Klausuren

Verbindliche Absprachen:
Bei der Formulierung von Aufgaben werden die für die Abiturprüfungen geltenden Operatoren des Faches Informatik schrittweise eingeführt, erläutert und dann im Rahmen der Aufgabenstellungen für die Klausuren benutzt.

Instrumente:

- Einführungsphase: 1 Klausur je Halbjahr
  Dauer der Klausur: 2 Unterrichtsstunden
- Grund- und Leistungskurse Q 1: 2 Klausuren je Halbjahr
  Dauer der Klausuren: 2 Unterrichtsstunden (Grundkurs), 3 (Leistungskurs)
- Grund- und Leistungskurse Q 2.1: 2 Klausuren
  Dauer der Klausuren: 3 Unterrichtsstunden (Grundkurs), 4 (Leistungskurs)
- Grund- und Leistungskurse Q 2.2: 1 Klausur unter Abiturbedingungen
- Anstelle einer Klausur kann gemäß dem Beschluss der Lehrerkonferenz in Q 1.2 eine Facharbeit geschrieben werden.

Die Aufgabentypen, sowie die Anforderungsbereiche I-III sind entsprechend den Vorgaben in Kapitel 3 des Kernlehrplans zu beachten.

Kriterien

Die Bewertung der schriftlichen Leistungen in Klausuren erfolgt über ein Raster mit Hilfspunkten, die im Erwartungshorizont den einzelnen Kriterien zugeordnet sind.

Spätestens ab der Qualifikationsphase orientiert sich die Zuordnung der Hilfspunktsumme zu den Notenstufen an dem Zuordnungsschema des Zentralabiturs.
Von diesem kann aber im Einzelfall begründet abgewichen werden, wenn sich z.B. besonders originelle Teillösungen nicht durch Hilfspunkte gemäß den Kriterien des Erwartungshorizontes abbilden lassen oder eine Abwertung wegen besonders schwacher Darstellung (APO-GOSt §13 (2)) angemessen erscheint.

Die Note ausreichend (5 Punkte) soll bei Erreichen von 45 % der Hilfspunkte erteilt werden.

2.3.2 Beurteilungsbereich Sonstige Mitarbeit

Den Schülerinnen und Schülern werden die Kriterien zum Beurteilungsbereich „sonstige Mitarbeit“ zu Beginn des Schuljahres genannt.

Verbindliche Absprachen der Fachkonferenz

- Alle Schülerinnen und Schüler führen in der Einführungsphase in Kleingruppen ein Kurzprojekt durch und fertigen dazu eine Arbeitsmappe mit Arbeitstagebuch an. Dies wird in die Note für die Sonstige Mitarbeit einbezogen.

- In der Qualifikationsphase erstellen, dokumentieren und präsentieren die Schülerinnen und Schüler in Kleingruppen ein anwendungsbezogenes Softwareprodukt. Dies wird in die Note für die Sonstige Mitarbeit einbezogen.

Leistungsaspekte

<u>Mündliche Leistungen</u>
- Beteiligung am Unterrichtsgespräch
- Zusammenfassungen zur Vor- und Nachbereitung des Unterrichts
- Präsentation von Arbeitsergebnissen
- Referate

- Mitarbeit in Partner-/Gruppenarbeitsphasen

Praktische Leistungen am Computer
- Implementierung, Test und Anwendung von Informatiksystemen

Sonstige schriftliche Leistungen
- Arbeitsmappe und Arbeitstagebuch zu einem durchgeführten Unterrichtsvorhaben
- Lernerfolgsüberprüfung durch kurze schriftliche Übungen
  In Kursen, in denen höchstens 50% der Kursmitglieder eine Klausur schreiben, finden schriftliche Übungen mindestens einmal pro Kurshalbjahr statt, in anderen Kursen entscheidet über die Durchführung die Lehrkraft.
  Schriftliche Übung dauern ca. 20 Minuten und umfassen den Stoff der letzten ca. 4–6 Stunden.
- Bearbeitung von schriftlichen Aufgaben im Unterricht

Kriterien

Die folgenden allgemeinen Kriterien gelten sowohl für die mündlichen als auch für die schriftlichen Formen der sonstigen Mitarbeit.

Die Bewertungskriterien stützen sich auf
- die Qualität der Beiträge,
- die Quantität der Beiträge und
- die Kontinuität der Beiträge.

Besonderes Augenmerk ist dabei auf
- die sachliche Richtigkeit,
- die angemessene Verwendung der Fachsprache,
- die Darstellungskompetenz,
- die Komplexität und den Grad der Abstraktion,
- die Selbstständigkeit im Arbeitsprozess,
- die Präzision und
- die Differenziertheit der Reflexion zu legen.

Bei Gruppenarbeiten auch auf

- das Einbringen in die Arbeit der Gruppe,
- die Durchführung fachlicher Arbeitsanteile und
- die Qualität des entwickelten Produktes.

Bei Projektarbeit darüber hinaus auf

- die Dokumentation des Arbeitsprozesses,
- den Grad der Selbstständigkeit,
- die Reflexion des eigenen Handelns und
- die Aufnahme von Beratung durch die Lehrkraft.

Grundsätze der Leistungsrückmeldung und Beratung

Die Grundsätze der Leistungsbewertung werden zu Beginn eines jeden Halbjahres den Schülerinnen und Schülern transparent gemacht. Leistungsrückmeldungen können erfolgen

- nach einer mündlichen Überprüfung,
- bei Rückgabe von schriftlichen Leistungsüberprüfungen,
- nach Abschluss eines Projektes,
- nach einem Vortrag oder einer Präsentation,
- bei auffälligen Leistungsveränderungen,
- auf Anfrage,
- als Quartalsfeedback und
- zu Eltern- oder Schülersprechtagen.

Die Leistungsrückmeldung kann

- durch ein Gespräch mit der Schülerin oder dem Schüler,
- durch einen Feedbackbogen,
- durch die schriftliche Begründung einer Note oder
- durch eine individuelle Lern-/Förderempfehlung

erfolgen.

Leistungsrückmeldungen erfolgen auch in der Einführungsphase im Rahmen der kollektiven und individuellen Beratung zur Wahl des Faches Informatik als fortgesetztes Grund- oder Leistungskursfach in der Qualifikationsphase.

3 Entscheidungen zu fach- und unterrichtsübergreifenden Fragen

Fach- und aufgabenfeldbezogene sowie übergreifende Absprachen, z.B. zur Arbeitsteilung bei der Entwicklung crosscurricularer Kompetenzen (ggf. Methodentage, Projektwoche, Facharbeitsvorbereitung, Schulprofil usw.)

Die Fachkonferenz Informatik hat sich im Rahmen des Schulprogramms für folgende zentrale Schwerpunkte entschieden:

Zusammenarbeit mit anderen Fächern
Im Informatikunterricht werden Kompetenzen anhand informatischer Inhalte in verschiedenen Anwendungskontexten erworben, in denen Schülerinnen und Schülern aus anderen Fächern Kenntnisse mitbringen können. Diese können insbesondere bei der Auswahl und Bearbeitung von Softwareprojekten berücksichtigt werden und in einem hinsichtlich der informatischen Problemstellung angemessenem Maß in den Unterricht Eingang finden. Da im Inhaltsfeld Informatik, Mensch und Gesellschaft auch gesellschaftliche und ethische Fragen im Unterricht angesprochen werden, soll eine mögliche Zusammenarbeit mit den Fächern Sozialwissenschaften und Philosophie in einer gemeinsamen Fachkonferenz ausgelotet werden.

Projekttage
Alle zwei Jahre werden am Konrad-Zuse-Gymnasium Projekttage angeboten. Die Fachkonferenz Informatik bietet in diesem Zusammenhang mindestens ein Projekt für Schülerinnen und Schüler der gymnasialen Oberstufe an.

Vorbereitung auf die Erstellung der Facharbeit
Möglichst schon zweiten Halbjahr der Einführungsphase, spätestens jedoch im ersten Halbjahr des ersten Jahres der Qualifikationsphase werden im Unterricht an geeigneten Stellen Hinweise zur Erstellung von Facharbeiten gegeben. Das betrifft u. a. Themenvorschläge, Hinweise zu den Anforderungen und zur Bewertung. Es wird vereinbart, dass nur Facharbeiten vergeben werden, die mit der eigenständigen Entwicklung eines Softwareproduktes verbunden sind.

Exkursionen

In der Einführungsphase wird im Rahmen des Unterrichtsvorhabens „Geschichte der digitalen Datenverarbeitung und die Grundlagen des Datenschutzes“ eine Exkursion zum Heinz Nixdorf MuseumsForum durchgeführt. Die außerunterrichtliche Veranstaltung wird im Unterricht vor- und nachbereitet.

4 Qualitätssicherung und Evaluation

Das schulinterne Curriculum stellt keine starre Größe dar, sondern ist als „lebendes Dokument“ zu betrachten. Dementsprechend sind die Inhalte stetig zu überprüfen, um ggf. Modifikationen vornehmen zu können. Die Fachkonferenz (als professionelle Lerngemeinschaft) trägt durch diesen Prozess zur Qualitätsentwicklung und damit zur Qualitätssicherung des Faches bei.

Durch Diskussion der Aufgabenstellung von Klausuren in Fachdienstbesprechungen und eine regelmäßige Erörterung der Ergebnisse von Leistungsüberprüfungen wird ein hohes Maß an fachlicher Qualitätssicherung erreicht.

Das schulinterne Curriculum (siehe 2.1) ist zunächst bis 2017 für den ersten Durchgang durch die gymnasiale Oberstufe nach Erlass des Kernlehrplanes verbindlich. Erstmalig nach Ende der Einführungsphase im Sommer 2015, werden in einer Sitzung der Fachkonferenz Erfahrungen ausgetauscht und ggf. Änderungen für den nächsten Durchgang der Einführungsphase beschlossen, um erkannten ungünstigen Entscheidungen schnellstmöglich entgegenwirken zu können.

Nach Abschluss des Abiturs 2017 wir die Fachkonferenz Informatik auf der Grundlage ihrer Unterrichtserfahrungen eine Gesamtsicht des schulinternen Curriculums vornehmen und ggf. eine Beschlussvorlage für die erste Fachkonferenz des folgenden Schuljahres erstellen.