

**Schulinternen Lehrplan
zum Kernlehrplan für die Qualifikationsphase**

Informatik

Qualifikationsphase (Q1 und Q2) - GRUNDKURS

Qualifikationsphase 1	
<p><u>Unterrichtsvorhaben Q1-I</u></p> <p>Thema: <i>Wiederholung der objektorientierten Modellierung und Programmierung anhand einer kontextbezogenen Problemstellung</i></p> <p>Zentrale Kompetenzen:</p> <ul style="list-style-type: none">• Argumentieren• Modellieren• Implementieren• Darstellen und Interpretieren• Kommunizieren und Kooperieren <p>Inhaltsfelder:</p> <ul style="list-style-type: none">• Daten und ihre Strukturierung• Algorithmen• Formale Sprachen und Automaten• Informatiksysteme <p>Inhaltliche Schwerpunkte:</p> <ul style="list-style-type: none">• Objekte und Klassen• Analyse, Entwurf und Implementierung von Algorithmen• Syntax und Semantik einer Programmiersprache• Nutzung von Informatiksystemen <p>Zeitbedarf: 8 Stunden</p>	<p><u>Unterrichtsvorhaben Q1-II</u></p> <p>Thema: <i>Modellierung und Implementierung von Anwendungen mit dynamischen, linearen Datenstrukturen</i></p> <p>Zentrale Kompetenzen:</p> <ul style="list-style-type: none">• Argumentieren• Modellieren• Implementieren• Darstellen und Interpretieren• Kommunizieren und Kooperieren <p>Inhaltsfelder:</p> <ul style="list-style-type: none">• Daten und ihre Strukturierung• Algorithmen• Formale Sprachen und Automaten <p>Inhaltliche Schwerpunkte:</p> <ul style="list-style-type: none">• Objekte und Klassen• Analyse, Entwurf und Implementierung von Algorithmen• Algorithmen in ausgewählten informatischen Kontexten• Syntax und Semantik einer Programmiersprache <p>Zeitbedarf: 20 Stunden</p>

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

Qualifikationsphase (Q1 und Q2) - LEISTUNGSKURS

Qualifikationsphase 1	
<p><u>Unterrichtsvorhaben Q1-I</u></p> <p>Thema:</p> <p><i>Wiederholung der objektorientierten Modellierung und Programmierung anhand einer kontextbezogenen Problemstellung unter Berücksichtigung der Gestaltung einer Benutzungsoberfläche</i></p> <p>Zentrale Kompetenzen:</p> <ul style="list-style-type: none"> • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren <p>Inhaltsfelder:</p> <ul style="list-style-type: none"> • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten • Informatiksysteme <p>Inhaltliche Schwerpunkte:</p> <ul style="list-style-type: none"> • Objekte und Klassen • Analyse, Entwurf und Implementierung von Algorithmen • Syntax und Semantik einer Programmiersprache • Nutzung von Informatiksystemen <p>Zeitbedarf: 15 Stunden</p>	<p><u>Unterrichtsvorhaben Q1-II</u></p> <p>Thema:</p> <p><i>Modellierung und Implementierung von dynamischen linearen Datenstrukturen und von Anwendungen mit dynamischen linearen Datenstrukturen in kontextbezogenen Problemstellungen</i></p> <p>Zentrale Kompetenzen:</p> <ul style="list-style-type: none"> • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren <p>Inhaltsfelder:</p> <ul style="list-style-type: none"> • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten <p>Inhaltliche Schwerpunkte:</p> <ul style="list-style-type: none"> • Objekte und Klassen • Analyse, Entwurf und Implementierung von Algorithmen • Algorithmen in ausgewählten informatischen Kontexten • Syntax und Semantik einer Programmiersprache <p>Zeitbedarf: 25 Stunden</p>

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

Qualifikationsphase 1	
<p><u>Unterrichtsvorhaben Q1-V</u></p> <p>Thema:</p> <p><i>Modellierung und Implementierung von dynamischen nicht-linearen Datenstrukturen und von Anwendungen mit dynamischen nicht-linearen Datenstrukturen in kontextbezogenen Problemstellungen.</i></p> <p>Zentrale Kompetenzen:</p> <ul style="list-style-type: none"> • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren <p>Inhaltsfelder:</p> <ul style="list-style-type: none"> • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten <p>Inhaltliche Schwerpunkte:</p> <ul style="list-style-type: none"> • Objekte und Klassen • Analyse, Entwurf und Implementierung von Algorithmen • Algorithmen in ausgewählten informatischen Kontexten • Syntax und Semantik einer Programmiersprache <p>Zeitbedarf: 40 Stunden</p>	<p><u>Unterrichtsvorhaben Q1-VI</u></p> <p>Thema:</p> <p><i>Projektorientierte Softwareentwicklung am Beispiel einer Anwendung mit Datenbankbindung</i></p> <p>Zentrale Kompetenzen:</p> <ul style="list-style-type: none"> • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren <p>Inhaltsfelder:</p> <ul style="list-style-type: none"> • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten • Informatik, Mensch und Gesellschaft <p>Inhaltliche Schwerpunkte:</p> <ul style="list-style-type: none"> • Objekte und Klassen • Analyse, Entwurf und Implementierung von Algorithmen • Datenbanken • Algorithmen in ausgewählten informatischen Kontexten • Syntax und Semantik einer Programmiersprache • Sicherheit <p>Zeitbedarf: 15 Stunden</p>
<p>Summe Qualifikationsphase 1: 135 Stunden</p>	

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

Konkretisierte Unterrichtsvorhaben

Im Folgenden sollen die im Vorfeld aufgeführten Unterrichtsvorhaben konkretisiert werden. 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)

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 verschie-

denen 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
<p>1. Wiederholung und Erweiterung der objektorientierten Modellierung und Programmierung durch Analyse und Erweiterung eines kontextbezogenen Beispiels</p> <p>(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</p>	<p>Die Schülerinnen und Schüler</p> <ul style="list-style-type: none"> • 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 Pro- 	<p><i>Beispiel: Wetthuepfen</i> 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.</p> <p>oder</p> <p><i>Beispiel: Tannenbaum</i> 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.</p> <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.1-Wiederholung (Download Q1-I.1)</p>

	grammen (I), <ul style="list-style-type: none"> • 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
<p>1. Die Datenstruktur Schlange im Anwendungskontext unter Nutzung der Klasse <code>Queue</code></p> <p>(a) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen</p> <p>(b) Erarbeitung der Funktionalität der Klasse <code>Queue</code></p> <p>(c) Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse <code>Queue</code></p>	<p>Die Schülerinnen und Schüler</p> <ul style="list-style-type: none"> • 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 nicht-lineare 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), • interpretieren Fehlermeldungen und korri- 	<p><i>Beispiel:</i> Patientenwarteschlange (jeder kennt seinen Nachfolger bzw. alternativ: seinen Vorgänger)</p> <p>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.</p> <p>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 <code>Queue</code>.</p> <p>Anschließend wird der Funktionsumfang der Anwendung erweitert: Patienten können sich zusätzlich in die Warteschlange zum Blutdruckmessen einreihen. Objekte werden von zwei Schlangen verwaltet.</p> <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.2 – Warteschlange (Download Q1-II.1)</p>
<p>2. Die Datenstruktur Stapel im Anwen-</p>		<p><i>Beispiel:</i> Heftstapel</p>

<p>dungskontext unter Nutzung der Klasse Stack</p> <p>(a) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen</p> <p>(b) Erarbeitung der Funktionalität der Klasse <code>Stack</code></p> <p>(c) Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse <code>Stack</code></p>	<p>gieren den Quellcode (I),</p> <ul style="list-style-type: none"> • testen Programme systematisch anhand von Beispielen (I), • stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D). 	<p>In einem Heftstapel soll das Heft einer Schülerin gefunden werden.</p> <p>oder</p> <p><i>Beispiel:</i> 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.</p>
<p>3. Die Datenstruktur lineare Liste im Anwendungskontext unter Nutzung der Klasse List</p> <p>(a) Erarbeitung der Vorteile der Klasse <code>List</code> im Gegensatz zu den bereits bekannten linearen Strukturen</p> <p>(b) Modellierung und Implementierung einer kontextbezogenen Anwendung unter Verwendung der Klasse <code>List</code>.</p>		<p><i>Beispiel:</i> 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.</p> <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.2 - Listen (Download Q1-II.2)</p>
<p>4. Vertiefung - Anwendungen von Listen, Stapeln oder Schlangen in mindestens einem weiteren Kontext</p>		<p><i>Beispiel:</i> Skispringen Ein Skispringer 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</p>

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 Halte-

punkt 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
<p>1. Suchen von Daten in Listen und Arrays</p> <p>(a) Lineare Suche in Listen und in Arrays</p> <p>(b) Binäre Suche in Arrays als Beispiel für rekursives Problemlösen</p> <p>(c) Untersuchung der beiden Suchverfahren hinsichtlich ihrer Effizienz (Laufzeitverhalten, Speicherbedarf)</p>	<p>Die Schülerinnen und Schüler</p> <ul style="list-style-type: none"> • 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 Implementierung und zur Analyse von Programmen 	<p><i>Beispiel:</i> Karteiverwaltung Für ein Adressverwaltungsprogramm soll eine Methode zum Suchen einer Adresse geschrieben werden.</p> <p>oder</p> <p><i>Beispiel:</i> 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.</p> <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.3 - Suchen und Sortieren (Download Q1-III.1)</p>

<p>2. Sortieren in Listen und Arrays - Entwicklung und Implementierung von iterativen und rekursiven Sortierverfahren</p> <p>(a) Entwicklung und Implementierung eines einfachen Sortierverfahrens für eine Liste</p> <p>(b) Implementierung eines einfachen Sortierverfahrens für ein Feld</p> <p>(c) Entwicklung eines rekursiven Sortierverfahrens für ein Feld (z.B. Sortieren durch Mischen)</p>	<p>(I),</p> <ul style="list-style-type: none"> • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme systematisch anhand von Beispielen (I), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D). 	<p><i>Beispiel:</i> Karteiverwaltung (s.o.)</p> <p>oder</p> <p><i>Beispiel:</i> Bundesjugendspiele (s.o.)</p> <p><i>Materialien:</i> (s.o.)</p>
<p>3. Untersuchung der Effizienz der Sortierverfahren „Sortieren durch direktes Einfügen“ und „Quicksort“ auf linearen Listen</p> <p>(a) Grafische Veranschaulichung der Sortierverfahren</p> <p>(b) Untersuchung der Anzahl der Vergleichsoperationen und des Speicherbedarf bei beiden Sortierverfahren</p> <p>(c) Beurteilung der Effizienz der beiden Sortierverfahren</p>		<p><i>Beispiel:</i> Karteiverwaltung (s.o.)</p> <p>oder</p> <p><i>Beispiel:</i> Bundesjugendspiele (s.o.)</p> <p><i>Materialien:</i> (s.o.)</p>

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
<p>1. Nutzung von relationalen Datenbanken</p> <p>(a) Aufbau von Datenbanken und Grundbegriffe</p> <ul style="list-style-type: none"> Entwicklung von Fragestellungen zur vorhandenen Datenbank Analyse der Struktur der vorgegebenen Datenbank und Erarbeitung der Begriffe Tabelle, Attribut, Datensatz, Datentyp, Primärschlüssel, Fremdschlüssel, Datenbankschema <p>(b) SQL-Abfragen</p> <ul style="list-style-type: none"> Analyse vorgegebener SQL-Abfragen und Erarbeitung der Sprachelemente von SQL (<code>SELECT (DISTINCT) ...FROM, WHERE, AND, OR, NOT</code>) auf einer Tabelle Analyse und Erarbeitung von SQL-Abfragen auf einer und mehrerer Tabelle zur Beantwortung der Fragestellungen (<code>JOIN, UNION, AS, GROUP BY, ORDER BY, ASC, DESC, COUNT, MAX, MIN, SUM, Arithmetische Operatoren: +, -, *, /, (...), Vergleichsoperatoren: =, <>, >, <, >=, <=, LIKE, BETWEEN, IN, IS NULL</code>) <p>(c) Vertiefung an einem weiteren Datenbankbeispiel</p>	<p>Die Schülerinnen und Schüler</p> <ul style="list-style-type: none"> 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 einem Datenbanksystem zu extrahieren (I), 	<p><i>Beispiel: VideoCenter</i></p> <p>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.</p> <p><i>Beispiel: Schulbuchausleihe</i></p> <p>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.</p>
<p>2. Modellierung von relationalen Datenbanken</p>		<p><i>Beispiel: Fahrradverleih</i></p> <p>Der Fahrradverleih <i>BTR (BikesToRent)</i> verleiht</p>

<p>(a) Entity-Relationship-Diagramm</p> <ul style="list-style-type: none"> • 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 <p>(b) Entwicklung einer Datenbank aus einem Datenbankentwurf</p> <ul style="list-style-type: none"> • Modellierung eines relationalen Datenbankschemas zu einem Entity-Relationship-Diagramm inklusive der Bestimmung von Primär- und Sekundärschlüsseln <p>(c) Redundanz, Konsistenz und Normalformen</p> <ul style="list-style-type: none"> • 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) 	<ul style="list-style-type: none"> • 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). 	<p>unterschiedliche Typen von Fahrrädern diverser Firmen an seine Kunden. Die Kunden sind bei <i>BTR</i> registriert (Name, Adresse, Telefon). <i>BTR</i> kennt von den Fahrradfirmen den Namen und die Telefonnummer. Kunden von <i>BTR</i> können CityBikes, Treckingräder und Mountainbikes ausleihen.</p> <p><i>Beispiel: Reederei</i> 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 Relationalalgebra eingegangen wird.</p> <p><i>Beispiel: Buchungssystem</i> 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.</p> <p><i>Beispiel: Schulverwaltung</i> In einer Software werden die Schulhalbjahre, Jahrgangsstufen, Kurse, Klassen, Schüler, Lehrer</p>
---	---	--

		<p>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.</p>
--	--	--

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
<p>1. Daten in Netzwerken und Sicherheitsaspekte in Netzen sowie beim Zugriff auf Datenbanken (a) Beschreibung eines Datenbankzugriffs</p>	<p>Die Schülerinnen und Schüler</p> <ul style="list-style-type: none"> • beschreiben und erläutern Topologien, 	<p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben, Verschlüsselung Q1.5 - Zugriff auf Daten in Netzwerken</p>

<p>im Netz anhand eines Anwendungskontextes und einer Client-Server-Struktur zur Klärung der Funktionsweise eines Datenbankzugriffs</p> <p>(b) Netztopologien als Grundlage von Client-Server-Strukturen und TCP/IP-Schichtenmodell als Beispiel für eine Paketübermittlung in einem Netz</p> <p>(c) Vertraulichkeit, Integrität, Authentizität in Netzwerken sowie symmetrische und asymmetrische kryptografische Verfahren (Cäsar-, Vigenère-, RSA-Verfahren) als Methoden Daten im Netz verschlüsselt zu übertragen</p>	<p>die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken (A),</p> <ul style="list-style-type: none"> • 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). 	<p>(Download Q1-V.1)</p>
<p>2. Fallbeispiele zur Datenschutzproblematik und zum Urheberrecht</p>		<p><i>Materialien:</i> <i>Ergänzungsmaterialien zum Lehrplannavigators Unterrichtsvorhaben Q1 5 - Datenschutz beim Videocenter, Materialblatt-Datenschutzgesetz</i> (Download Q1-V.2)</p>

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 Umkehrbarkeit der Zuordnung Binärbaum \rightarrow 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
<p>1. Analyse von Baumstrukturen in verschiedenen Kontexten</p> <p>(a) Grundlegende Begriffe (Grad, Tiefe, Höhe, Blatt, Inhalt, Teilbaum, Ebene, Vollständigkeit)</p> <p>(b) Aufbau und Darstellung von binären Bäumen anhand von Baumstrukturen in verschiedenen Kontexten</p>	<p>Die Schülerinnen und Schüler</p> <ul style="list-style-type: none"> • erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), • analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit 	<p><i>Beispiel:</i> Termbaum Der Aufbau von Termen wird mit Hilfe von binären Baumstrukturen verdeutlicht.</p> <p>oder</p> <p><i>Beispiel:</i> Ahnenbaum Die binäre Baumstruktur ergibt sich daraus, dass jede Person genau einen Vater und eine Mutter</p>

	<p>und die Funktionalität von Programmen (A),</p> <ul style="list-style-type: none"> • 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), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), 	<p>hat.</p> <p><i>Weitere Beispiele für Anwendungskontexte für binäre Bäume:</i></p> <p><i>Beispiel:</i> 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.)</p> <p><i>oder</i></p> <p><i>Beispiel:</i> 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.</p> <p><i>oder</i></p> <p><i>Beispiel:</i> 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 Strich entspricht. Wenn man im Gesamtbaum startet und durch Übergänge zu linken oder rech-</p>
--	--	---

	<ul style="list-style-type: none"> • 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). 	<p>ten Teilbäumen einen Pfad zum gewünschten Buchstaben sucht, erhält man die Morsecodierung des Buchstabens.</p> <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.1 – Binärbaum (Download Q2-I.1)</p>
<p>2. Die Datenstruktur Binärbaum im Anwendungskontext unter Nutzung der Klasse <code>BinaryTree</code></p> <p>(a) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen im Anwendungskontext</p> <p>(b) Modellierung eines Entwurfsdiagramms und Entwicklung eines Implementationsdiagramms</p> <p>(c) Erarbeitung der Klasse <code>BinaryTree</code> und beispielhafte Anwendung der Operationen</p> <p>(d) Implementierung der Anwendung oder von Teilen der Anwendung</p> <p>(e) Traversierung eines Binärbaums im Pre-, In- und Postorderdurchlauf</p>		<p><i>Beispiel: Informatikerbaum als binärer Baum</i> In einem <i>binären Baum</i> 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.)</p> <p>Folgende Funktionalitäten werden benötigt:</p> <ul style="list-style-type: none"> • Einfügen der Informatiker-Daten in den Baum • Suchen nach einem Informatiker über den Schlüssel Name • Ausgabe des kompletten Datenbestands in nach Namen sortierter Reihenfolge <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.1 – Binärbaum (Download Q2-I.2)</p>
<p>3. Die Datenstruktur binärer Suchbaum im Anwendungskontext unter Verwendung der Klasse <code>BinarySearchTree</code></p>		<p><i>Beispiel: Informatikerbaum als Suchbaum</i> In einem binären <i>Suchbaum</i> werden die Namen und die Geburtsdaten von Informatikern lexiko-</p>

<p>(a) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen</p> <p>(b) Modellierung eines Entwurfsdiagramms und Entwicklung eines Implementationsdiagramm, grafische Darstellung eines binären Suchbaums und Erarbeitung der Struktureigenschaften</p> <p>(c) Erarbeitung der Klasse <code>BinarySearchTree</code> und Einführung des Interface <code>Item</code> zur Realisierung einer geeigneten Ordnungsrelation</p> <p>(d) Implementierung der Anwendung oder von Teilen der Anwendung inklusive einer sortierten Ausgabe des Baums</p>		<p>graphisch 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.)</p> <p>Folgende Funktionalitäten werden benötigt:</p> <ul style="list-style-type: none"> • Einfügen der Informatiker-Daten in den Baum • Suchen nach einem Informatiker über den Schlüssel <code>Name</code> • Ausgabe des kompletten Datenbestands in nach Namen sortierter Reihenfolge <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.1 – Binärer Suchbaum (Download Q2-I.3)</p>
<p>4. Übung und Vertiefungen der Verwendung von Binärbäumen oder binären Suchbäumen anhand weiterer Problemstellungen</p>		<p><i>Beispiel:</i> Codierungsbäume (s.o.) oder Huffman-Codierung</p> <p><i>oder</i></p> <p><i>Beispiel:</i> 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 <code>Buchindex</code> als Suchbaum (Objekt der Klasse <code>BinarySearchTree</code>) 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</p>

		<p>aktuellen Teilbaum stehen, sind in dessen rechtem Teilbaum. (Dies gilt für alle Teilbäume.)</p> <p><i>oder</i></p> <p><i>Beispiel:</i> Entscheidungsbäume (s.o.)</p> <p><i>oder</i></p> <p><i>Beispiel:</i> Termbaum (s.o.)</p> <p><i>oder</i></p> <p><i>Beispiel:</i> Ahnenbaum (s.o.)</p> <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.1 – Anwendung Binärbaum (Download Q2-I.4)</p>
--	--	---

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
<p>1. Endliche Automaten</p> <p>(a) Vom Automaten in den Schülerinnen und Schülern bekannten Kontexten zur formalen Beschreibung eines endlichen Automaten</p> <p>(b) Untersuchung, Darstellung und Entwicklung endlicher Automaten</p>	<p>Die Schülerinnen und Schüler</p> <ul style="list-style-type: none"> • 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), 	<p><i>Beispiele:</i> Cola-Automat, Geldspielautomat, Roboter, Zustandsänderung eines Objekts „Auto“, Akzeptor für bestimmte Zahlen, Akzeptor für Teilwörter in längeren Zeichenketten, Akzeptor für Terme</p> <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.2 – Endliche Automaten, Formale Sprachen (Download Q2-II.1)</p>
<p>2. Untersuchung und Entwicklung von Grammatiken regulärer Sprachen</p> <p>(a) Erarbeitung der formalen Darstellung regulärer Grammatiken</p> <p>(b) Untersuchung, Modifikation und Entwicklung von Grammatiken</p> <p>(c) Entwicklung von endlichen Automaten zum Erkennen regulärer Sprachen die durch Grammatiken gegeben werden</p> <p>(d) Entwicklung regulärer Grammatiken zu endlichen Automaten</p>	<ul style="list-style-type: none"> • 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), • entwickeln zu einer regulären Sprache eine Grammatik, die die Sprache erzeugt (M), • stellen endliche Automaten in Tabellen 	<p><i>Beispiele:</i> reguläre Grammatik für Wörter mit ungerader Parität, Grammatik für Wörter, die bestimmte Zahlen repräsentieren, Satzgliederungsgrammatik</p> <p><i>Materialien:</i> (s.o.)</p>
<p>3. Grenzen endlicher Automaten</p>	<ul style="list-style-type: none"> • entwickeln zu einer regulären Sprache eine Grammatik, die die Sprache erzeugt (M), • stellen endliche Automaten in Tabellen 	<p><i>Beispiele:</i> Klammerausdrücke, $a^n b^n$ im Vergleich zu $(ab)^n$</p>

	oder Graphen dar und überführen sie in die jeweils andere Darstellungsform (D), <ul style="list-style-type: none"> • 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, das 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	Die Schülerinnen und Schüler	<i>Beispiel:</i>

<p>Ausführung maschinennaher Programme</p> <p>a) prinzipieller Aufbau einer von Neumann-Architektur mit CPU, Rechenwerk, Steuerwerk, Register und Hauptspeicher</p> <p>b) einige maschinennahe Befehle und ihre Repräsentation in einem Binär-Code, der in einem Register gespeichert werden kann</p> <p>c) Analyse und Erläuterung der Funktionsweise eines einfachen maschinennahen Programms</p>	<ul style="list-style-type: none"> • 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). 	<p>Addition von 4 zu einer eingegebenen Zahl mit einem Rechnermodell</p> <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.3 –Von-Neumann-Architektur und maschinennahe Programmierung (Download Q2-III.1)</p>
<p>2. Grenzen der Automatisierbarkeit</p> <p>a) Vorstellung des Halteproblems</p> <p>b) Unlösbarkeit des Halteproblems</p> <p>c) Beurteilung des Einsatzes von Informatiksystemen hinsichtlich prinzipieller Möglichkeiten und prinzipieller Grenzen</p>		<p><i>Beispiel:</i> Halteproblem</p> <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.3 - Halteproblem (Download Q2-III.2)</p>

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
<p>2. Entwicklung eines Anforderungskatalogs</p> <p>(g) Analyse der Problemstellung (h) Entwicklung von Anwendungsfällen (use cases) (i) Formulierung eines Anforderungskatalogs</p>	<p>Die Schülerinnen und Schüler</p> <ul style="list-style-type: none"> 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), 	<p><i>Beispiel:</i> 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.</p>
<p>3. Entwurf einer grafischen Benutzungsoberfläche</p> <p>(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</p>	<ul style="list-style-type: none"> 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), 	<p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.1 (Download LK-Q1-I.1)</p>
<p>4. Klassenmodellierung und Visualisierung von Objektkommunikation</p> <p>(a) Entwurf eines Implementationsdiagramms unter Berücksichtigung des Anforderungskatalogs und der Benutzungsoberfläche (b) Grafische Darstellung der Kommunika-</p>	<ul style="list-style-type: none"> wenden eine didaktisch orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I), interpretieren Fehlermeldungen und 	

<p>tion zwischen mindestens zwei Objekten (c) Dokumentation von Klassen</p>	<p>korrigieren den Quellcode (I),</p> <ul style="list-style-type: none"> • testen Programme systematisch anhand von Beispielen und mithilfe von Testanwendungen(I), 	
<p>5. Projektentwicklung und Test (ggf. arbeitsteilig) (a) Implementierung der Anwendung mit grafischer Benutzungsoberfläche (b) Test einzelner Klassen der Gesamtanwendung mit Hilfe von Testanwendungen</p>	<ul style="list-style-type: none"> • 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). 	<p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.1 (Download LK-Q1-I.2)</p>

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 sequenziell 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
<p>5. Modellierung und Implementation einer Datenstruktur Schlange, die die Daten nach dem FIFO-Prinzip im Anwendungskontext verwaltet.</p>	<p>Die Schülerinnen und Schüler</p> <ul style="list-style-type: none"> • 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, ihre Operationen und ihre Beziehungen 	<p><i>Beispiel:</i> (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.</p> <p>Erster Lösungsansatz: Die FIFO-Datenstruktur wird problembezogen modelliert und implementiert.</p> <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.II (Download LK-Q1-II.1)</p>

<p>6. Die Datenstruktur Schlange im Anwendungskontext unter Nutzung der Klasse <code>Queue</code></p> <p>(d) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen</p> <p>(e) Erarbeitung der Funktionalität der Klasse <code>Queue</code> und des Konzepts der parametrisierten Klassen</p> <p>Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse <code>Queue</code></p>	<p>(M),</p> <ul style="list-style-type: none"> • 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). 	<p>Zweiter Lösungsansatz: Bei der Modellierung und Implementation wird der generische Datentyp <code>Queue</code> verwendet.</p> <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.II (Download LK-Q1-II.2)</p>
<p>7. Die Datenstruktur Stapel im Anwendungskontext unter Nutzung der Klasse <code>Stack</code></p> <p>(d) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen</p> <p>(e) Erarbeitung der Funktionalität der Klasse <code>Stack</code></p> <p>(f) Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse <code>Stack</code></p>	<p>(M),</p> <ul style="list-style-type: none"> • 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). 	<p>Zweiter Lösungsansatz: Bei der Modellierung und Implementation wird der generische Datentyp <code>Stack</code> verwendet.</p> <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.II (Download LK-Q1-II.3)</p> <p><i>Beispiel:</i> (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 <code>Stack</code>.</p> <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.II (Download LK-Q1-II.3)</p>

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
<p>4. Suchen von Daten in Listen und Arrays</p> <p>(d) Lineare Suche in Listen und in Arrays</p> <p>(e) Binäre Suche in Arrays als Beispiel für rekursives Problemlösen</p> <p>(f) Untersuchung der beiden Suchverfahren hinsichtlich ihrer Effizienz (Speicherbedarf, Anzahl der Vergleiche)</p>	<p>Die Schülerinnen und Schüler</p> <ul style="list-style-type: none"> • 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), 	<p><i>Beispiel:</i> Benutzerverwaltung für einen Fotokopierer</p> <p>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.</p> <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.III - Suchen und Sortieren (Download Q1-III.1)</p>
<p>5. Sortieren in Listen und Arrays - Entwicklung und Implementierung von iterativen und rekursiven Sortierverfahren</p> <p>(d) Entwicklung und Implementierung eines einfachen Sortierverfahrens für eine Liste</p> <p>(e) Implementierung eines einfachen Sortierverfahrens für ein Feld</p> <p>(f) Entwicklung eines rekursiven Sortierverfahrens für ein Feld (z.B. Sortieren durch Mischen)</p>	<ul style="list-style-type: none"> • 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), • testen Programme systematisch anhand 	<p><i>Beispiel:</i> Benutzerverwaltung für einen Fotokopierer (s.o.)</p> <p>Die Benutzungsdaten sollen nach den Kriterien Kopierverbrauch, Namen und PIN-Nummern geordnet ausgegeben werden können.</p> <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.III - Suchen und Sortieren (Download Q1-III.2)</p>

<p>6. Untersuchung der Effizienz der Sortierverfahren „Sortieren durch direktes Einfügen“ und „Quicksort“ auf linearen Listen</p> <p>(d) Grafische Veranschaulichung der Sortierverfahren</p> <p>(e) Untersuchung der Anzahl der Vergleichsoperationen bei beiden Sortierverfahren</p> <p>(f) Beurteilung der Effizienz der beiden Sortierverfahren, untere Schranke für die Laufzeit von Sortieralgorithmen</p>	<p>von Beispielen und mit Hilfe von Testanwendungen (I),</p> <ul style="list-style-type: none"> stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D). 	<p><i>Beispiel:</i> Test- und Analyseumgebung für Sortieralgorithmen</p> <p>Listen mit unterschiedlich vielen Listenelementen und unterschiedlicher Vorsortierung werden bezüglich der Anzahl der Vergleiche von Listenelementen analysiert.</p> <p>Die untere Schranke für die Laufzeit wird bestimmt.</p> <p><i>Materialien:</i></p> <p>Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.III - Suchen und Sortieren (Download Q1-III.3)</p>
---	--	---

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
<p>3. Nutzung von relationalen Datenbanken</p> <p>(d) Aufbau von Datenbanken und Grundbegriffe</p> <ul style="list-style-type: none"> Entwicklung von Fragestellungen zur vorhandenen Datenbank Analyse der Struktur der vorgegebenen Datenbank und Erarbeitung der Begriffe Tabelle, Attribut, Datensatz, Datentyp, Primärschlüssel, Fremdschlüssel, Relationenschema, Datenbankschema <p>(e) SQL-Abfragen</p> <ul style="list-style-type: none"> Analyse vorgegebener SQL-Abfragen und Erarbeitung der Sprachelemente von SQL (<code>SELECT (DISTINCT) ...FROM, WHERE, AND, OR, NOT</code>) auf einer Tabelle Analyse und Erarbeitung von SQL-Abfragen auf einer und mehreren Tabellen zur Beantwortung der Fragestellungen (<code>JOIN</code> und Varianten, <code>UNION, AS, GROUP BY, ORDER BY, ASC, DESC, COUNT, MAX, MIN, SUM, AVG</code>, Arithmetische Operatoren: <code>+, -, *, /, (...)</code>, Vergleichsoperatoren: <code>=, <>, >, <, >=, <=, LIKE, BETWEEN, IN, IS NULL</code>) 	<p>Die Schülerinnen und Schüler</p> <ul style="list-style-type: none"> 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), 	<p><i>Beispiel:</i> 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.</p> <p><i>Beispiel:</i> 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.</p>

<p>(f) Vertiefung an einem weiteren Datenbankbeispiel</p>	<ul style="list-style-type: none"> • überführen Datenbankschemata in die 1. bis 3. Normalform (M), • ermitteln Ergebnisse von Datenbankabfragen über mehrere verknüpfte Tabellen (D), 	
<p>4. Modellierung von relationalen Datenbanken</p> <p>(d) Entity-Relationship-Diagramm</p> <ul style="list-style-type: none"> • 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 <p>(e) Entwicklung einer Datenbank aus einem Datenbankentwurf</p> <ul style="list-style-type: none"> • Modellierung eines relationalen Datenbankschemas zu einem Entity-Relationship-Diagramm inklusive der Bestimmung von Primär- und Sekundärschlüsseln <p>(f) Redundanz, Konsistenz und Normalformen</p> <ul style="list-style-type: none"> • Untersuchung einer Datenbank hinsichtlich Konsistenz und Redundanz in einer Anwendungssituation • Überprüfung von Datenbankschemata 	<ul style="list-style-type: none"> • 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). 	<p><i>Beispiel: Fahrradverleih</i> Der Fahrradverleih <i>BTR (BikesToRent)</i> verleiht unterschiedliche Typen von Fahrrädern diverser Firmen an seine Kundinnen und Kunden. Die Kundinnen und Kunden sind bei <i>BTR</i> registriert (Name, Adresse, Telefon). <i>BTR</i> kennt von den Fahrradfirmen den Namen und die Telefonnummer. Kundinnen und Kunden von <i>BTR</i> können CityBikes, Trekkingräder und Mountainbikes ausleihen.</p> <p><i>Beispiel: Reederei</i> 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.</p> <p><i>Beispiel: Buchungssystem</i> In dem Online-Buchungssystem einer Schule können die Lehrenden Medienräume, Beamer, Laptops, Kameras, usw. für einen bestimmten Zeitpunkt buchen, der durch Datum und die</p>

<p>hinsichtlich der 1. bis 3. Normalform und Normalisierung (um Redundanzen zu vermeiden und Konsistenz zu gewährleisten)</p>		<p>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.</p> <p><i>Beispiel: Schulverwaltung</i> 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.</p>
---	--	---

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
<p>1. Analyse von Graphen in verschiedenen Kontexten</p> <p>(a) Grundlegende Begriffe (Graph, gerichtet – ungerichtet, Knoten, Kanten, Kantengewicht)</p> <p>(b) Aufbau und Darstellung von Graphen anhand von Graphenstrukturen in verschiedenen Kontexten (Adjazenzmatrix, Adjazenzliste)</p>	<p>Die Schülerinnen und Schüler</p> <ul style="list-style-type: none"> • 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), • verwenden bei der Modellierung geeigneter Problemstellungen die Möglichkeiten der Polymorphie (M), 	<p><i>Beispiel:</i> „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.</p> <p><i>Beispiel</i> 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.</p> <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben LK-Q1.V (Download LK-Q1.V.1)</p>

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.

- 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),
- wenden didaktisch orientierte Entwicklungsumgebungen zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I),
- stellen lineare und nichtlineare Strukturen

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 vom Start- zum Zielknoten gefunden werden. Damit wird das Verfahren beim Dijkstra-

	<p>grafisch dar und erläutern ihren Aufbau (D),</p> <ul style="list-style-type: none"> • 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). 	<p>Algorithmus vorbereitet. <i>Beispiel: Kürzeste Wege</i> 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 <i>GraphTool</i> 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.</p> <p><i>Beispiel: Versorgungsnetz</i> 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</p>
--	--	--

bereiten die nächste Unterrichtssequenz vor.

Materialien:

Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben LK-Q1.V

(Download LK-Q1.V.2)

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

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

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 „Parse eines einfachen Terms und die Erzeugung eines Termbaums“ wieder aufgegriffen werden.

Beispiel: Informatikerbaum 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:

		<p>Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben LK-Q1.V</p> <p>(Download LK-Q1-V.3)</p>
<p>4. Erarbeitung, Implementierung und Verwendung der Datenstruktur binärer Suchbaum im Anwendungskontext</p> <p>(a) Erarbeitung der Eigenschaften eines binären Suchbaums im Anwendungskontext</p> <p>(b) Erarbeitung der Attribute und Methoden der generischen Klasse <code>BinarySearchTree<ContentType></code> und des Interfaces <code>ComparableContent</code></p> <p>(c) Implementierung des Konstruktors und der Methode <code>search</code> der Klasse <code>BinarySearchTree<ContentType></code></p> <p>(d) Implementierung eines Anwendungsbeispiels einschließlich der sortierten Ausgabe eines binären Suchbaumes</p>		<p><i>Beispiel:</i> Informatikerbaum binärer Suchbaum Das Beispiel wird wieder aufgegriffen und diesmal mit der Klasse <code>BinarySearchTree<ContentType></code> implementiert. Durch Modifikation der implementierten Methoden des Interfaces <code>ComparableContent</code> wird der Suchbaum nach den Geburtsdaten der Informatiker sortiert.</p> <p><i>Beispiel:</i> 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.</p> <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben LK-Q1.VI</p> <p>(Download LK-Q1.V.4)</p>

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 abrufen, 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
<p>1. Analyse einer lebensweltnahen Problemstellung im Hinblick auf die Entwicklung eines Java-Programms mit Datenbankbindung</p> <p>(a) Entwicklung einer Programmidee (b) Analyse des Problembereichs (c) Entwicklung eines Anforderungskatalogs für das zu entwickelnde Programm</p>	<p>Die Schülerinnen und Schüler</p> <ul style="list-style-type: none"> • 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), • analysieren und erläutern objektorientierte 	<p><i>Beispiel:</i> Quizspiel Verwaltung von Quizfragen, Antworten und Ranglisten</p> <p><i>Beispiel:</i> Navigationssystem Ermittlung von kürzesten Wegen im deutschen Autobahnnetz</p> <p><i>Beispiel:</i> Verwaltung der Schülerbücherei Verwaltungsprogramm für das Einpflegen und Ausleihe von Büchern der Schülerbibliothek</p> <p><i>Beispiel:</i> Fehlstundenverwaltung Verwaltungsprogramm für die Fehlstunden von Schülerinnen und Schülern</p> <p><i>Beispiel:</i> Sportfestverwaltung Verwaltung von Aufgaben, Sportereignissen und Ergebnissen des Schulsportfestes</p> <p><i>Beispiel:</i> Materialverwaltung</p>

	<p>Modellierungen (A),</p> <ul style="list-style-type: none"> • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), 	<p>Materialien für Vertretungsstunden sollen verwaltet werden.</p>
<p>2. Modellierung einer datenbankgestützten Problemlösung unter Berücksichtigung des MVC-Prinzips</p> <p>(a) Strukturierung nach dem MVC-Prinzip</p> <p>(b) Modellierung der Datenbank (ER-Diagramm, Datenbankschema)</p> <p>(c) Modellierung der Kontrollklassen und Entwicklung von SQL-Anweisungen entsprechend des Anforderungskatalogs</p> <p>(d) Modellierung einer grafischen Benutzungsoberfläche</p>	<ul style="list-style-type: none"> • 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), 	<p><i>Materialien:</i></p> <p>Ergänzungsmaterialien zum Lehrplannavigator – Dokumentation der Abiturklassen zur Datenbankanbindung (Download LK-Q1-VI.1)</p> <p>Ergänzungsmaterialien zum Lehrplannavigator – Projekt Navigationssystem (Download LK-Q1-VI.2)</p> <p>Ergänzungsmaterialien zum Lehrplannavigator – Projekt Quizspiel (Download LK-Q1-VI.3)</p>
<p>3. Umsetzung des Softwareprojektes</p> <p>(a) Umsetzung der Datenbank in einem Datenbankmanagementsystem</p> <p>(b) Implementierung der Kontrollklassen mit Anbindung an die Datenbank unter Verwendung didaktischer Klassen</p> <p>(c) Integration in den Prototypen der Benutzeroberfläche</p>	<ul style="list-style-type: none"> • 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), • modifizieren Algorithmen und Programme (I), 	<p><i>Materialien:</i></p> <p>Ergänzungsmaterialien zum Lehrplannavigator – Dokumentation der Abiturklassen zur Datenbankanbindung (Download LK-Q1-VI.1)</p> <p>Ergänzungsmaterialien zum Lehrplannavigator – Projekt Navigationssystem (Download LK-Q1-VI.2)</p> <p>Ergänzungsmaterialien zum Lehrplannavigator – Projekt Quizspiel (Download LK-Q1-VI.3)</p>

4. **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

- 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 Infor-

	<p>matiksystem (M),</p> <ul style="list-style-type: none">• 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
<p>1. Einführung in die Problemfelder Datenschutz, Urheberrecht und moralische Verantwortung</p> <p>a. Vorstellung eines komplexen Fallbeispiels</p> <p>b. Erarbeitung von Interesse verschiedener Interessensgruppen im Hinblick auf die Problemfelder</p> <p>c. Erster Bewertungsversuch auf Grundlage des Vorwissens von Schülerinnen und Schülern</p>	<p>Die Schülerinnen und Schüler</p> <ul style="list-style-type: none"> • 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 	<p><i>Beispiel: autonomes Fahren</i></p> <p>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 auch zu tausenden von Arbeitslosen durch Weg-</p>

	<p>Vorgaben, ethischer Aspekte und gesellschaftlicher Werte unter Berücksichtigung unterschiedlicher Interessenlagen (A).</p> <ul style="list-style-type: none"> • analysieren und erläutern Eigenschaften, Funktionsweisen und Einsatzbereiche symmetrischer und asymmetrischer Verschlüsselungsverfahren (A). 	<p>fall ganzer Berufszweige führen.</p> <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.1 – Fallbeispiel „autonomes Fahren“ (Download LK-Q2-I.1)</p>
<p>2. Erarbeitung grundlegender Positionen (ggf. in zieldifferenten Gruppen)</p> <p>a. Erarbeitung von Grundideen des Datenschutzes (z.B. personenbezogene Daten, informationelle Selbstbestimmung, Datensparsamkeit usw.)</p> <p>b. Erarbeitung von Grundideen des Urheberrechts anhand eines verbreiteten Lizenzsystems</p> <p>c. Erarbeitung eines einfachen moralischen Bewertungsmaßstabes</p> <p>d. Anwendung auf Fallbeispiele</p>	<ul style="list-style-type: none"> • untersuchen und beurteilen Grenzen des Problemlösens mit Informatiksystemen (A). • nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zur Erschließung, Aufbereitung und Präsentation fachlicher Inhalte (D). 	<p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.1 – Grundlagen „Informatik, Mensch und Gesellschaft“ (Download LK-Q2-I.2)</p> <p><i>Beispiel: Creative-Commons-Lizenzen</i> URL: http://de.creativecommons.org (Abgerufen am 19.06.2016)</p>
<p>3. Zusammenfassung und Sicherung</p> <p>a. Darstellung der relevanten Aspekte zum Datenschutz, Urheberrecht und zur moralischen Bewertung</p> <p>b. Demonstration an jeweils einem einfachen Beispiel</p> <p><i>Anmerkung:</i> Dieser Schritt ist entscheidend, wenn in die Punkte 2 bis 4 arbeitsteilig umgesetzt wur-</p>		<p><i>Beispiel: Stellwände</i> Eine Zusammenfassung und Sicherung kann in Form von Plakatstellwänden erfolgen. Diese können auch im Sinne einer Dauerausstellung im Schulgebäude präsentiert werden.</p>

den.		
<p>4. Stellungnahme zu einem komplexen Fallbeispiel</p> <ul style="list-style-type: none"> 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 		<p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.1 – Materialblatt „Stellungnahme“ (Download LK-Q2-I.3)</p>
<p>5. Sicherheitsziele und der kryptographische Ansatz</p> <ul style="list-style-type: none"> 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 Verschlüsse- 		<p><i>Beispiel:</i> Sicherheitsziele <i>Zugriffskontrolle</i> und <i>Authentifikation</i> anhand des „Keyless Entry-System“ bei Autos</p> <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.1 – Sicherheitsziele und Kommunikationsrollenspiel (Download LK-Q2.I.4)</p>

lung		
<p>6. Verschlüsselungsverfahren und deren Sicherheit</p> <ul style="list-style-type: none"> 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 		<p><i>Beispiel: Caesar-Verschlüsselung</i> Die Cäsarchiffre als Beispiel für ein monoalphabetisches Verfahren und die Exhaustionsmethode (Brute-Force-Methode) zur Verdeutlichung der Bedeutsamkeit eines starken Schlüssels</p> <p><i>Beispiel: Vigenère-Verschlüsselung</i> Die Vigenère-Chiffre als Verbesserung einer monoalphabetischen Verschlüsselung und der Kasiski-Test als Grundlage eines verbesserten Angriffsverfahrens</p> <p><i>Beispiel: One-Time-Pad</i> Der One-Time-Pad als Spezialfall eines sicheren polyalphabetischen Verschlüsselungsverfahrens</p> <p><i>Beispiel:</i> Das RSA-Verfahren als Beispiel für eine asymmetrische Verschlüsselung mit einer Einwegfunktion</p> <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator – Verschlüsselungsverfahren (Download LK-Q2.1.5)</p> <p><i>Materialien:</i> URL: https://www.cryptool.org/de/ (Abgerufen am 19.06.2016)</p>

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
<p>1. Grundlagen der Datenübertragung in Netzwerken</p> <p>(a) Schichtenmodell</p> <ul style="list-style-type: none"> • Erarbeitung eines standardisierten Schichtenmodells für Netzwerkkommunikation <p>(b) Grundlagen der Codierung</p> <ul style="list-style-type: none"> • Erarbeitung einer eigenen, vereinfachten Codierung für die Bitübertragung <p>(c) Topologien</p> <ul style="list-style-type: none"> • Erarbeitung und Vergleich ausgewählter Netzwerktopologien <p>(d) Routing</p> <ul style="list-style-type: none"> • Analyse von Grundlagen der Adressierung in IP-Netzwerken <p>(e) Protokolle</p> <ul style="list-style-type: none"> • Erarbeitung des beispielhaften Aufbaus eines Protokolls auf der Anwendungsschicht 	<p>Die Schülerinnen und Schüler</p> <ul style="list-style-type: none"> • 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), 	<p><i>Material:</i> Aufgabensammlung</p> <p>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“</p>
<p>2. Analyse, Modellierung und Implementierung von Netzwerkanwendungen in Client-Server-Struktur</p> <p>(a) Nutzung einfacher Server-Dienste mittels Client</p>		<p><i>Beispiel:</i> Echo- bzw. Daytime-Clients und – Server sowie eigene Erweiterungen</p> <p>Anhand der Echo- und Daytime-Dienste (z.B. lokal im Schulnetz durch den Lehrenden zur</p>

<ul style="list-style-type: none"> • Modellierung und Implementierung von Clients für einfach Serverdienste <p>(b) Anbieten von Diensten mittels Server</p> <ul style="list-style-type: none"> • Analyse vorgegebener Implementatio- nen einfacher Server • Modellierung und Implementierung ei- gener Server 	<ul style="list-style-type: none"> • analysieren und erläutern objektorientierte Modellierungen (A), • stellen Klassen und ihre Beziehungen gra- fisch dar (D), • implementieren Klassen in einer Pro- grammiersprache auch unter Nutzung do- kumentierter Klassenbibliotheken (I). 	<p>Verfügung gestellt) erarbeiten die Schüle- rin- nen und Schüler zunächst den die Funkti- onsweise 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. Erweiterun- gen/Abwandlungen derselben (ggf. mit Stei- gerung des Interaktionsgrades) selbst.</p>
<p>3. Entwicklung eines vollständigen Cli- ent-Server-Systems</p> <ul style="list-style-type: none"> • Protokollentwurf, Dialogorientierung • Modellierung mittels Entwurfs- und Im- plementationsdiagramm • Bedeutung von Nebenläufigkeit • Implementierung 		<p><i>Beispiel:</i> Messenger-Dienst</p> <p>Abschließend entwickeln die Schülerinnen und Schüler ein Client-Server-System zum Versenden von Nachrichten zwischen einzel- nen Rechnern (einfacher Messenger), basie- rend auf selbst gewählten „Nicknames“.</p>

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
<p>4. Endliche Automaten</p> <p>(e) Erarbeitung der formalen Beschreibung eines endlichen Automaten auf der Grundlage von Automaten in bekannten Kontexten</p> <p>(f) Untersuchung, Darstellung und Entwicklung endlicher Automaten</p> <p>(g) Umwandlung nichtdeterministischer endlicher Automaten in deterministische endliche Automaten</p>	<p>Die Schülerinnen und Schüler</p> <ul style="list-style-type: none"> 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), 	<p><i>Beispiele:</i> Cola-Automat, Geldspielautomat, Roboter, Zustandsänderung eines Objekts „Auto“, Akzeptor für bestimmte Zahlen, Akzeptor für Teilwörter in längeren Zeichenketten, Akzeptor für Terme</p> <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.III (Download LK-Q2-III.1)</p>
<p>5. Untersuchung und Entwicklung von Grammatiken regulärer Sprachen</p> <p>(a) Erarbeitung der formalen Darstellung regulärer Grammatiken</p> <p>(b) Untersuchung, Modifikation und Entwicklung von Grammatiken</p> <p>(c) Entwicklung von endlichen Automaten zum Erkennen regulärer Sprachen die durch Grammatiken gegeben werden</p> <p>(d) Entwicklung regulärer Grammatiken zu endlichen Automaten</p> <p>(e) Entwicklung eines Parsers für eine einfache reguläre Sprache</p>	<ul style="list-style-type: none"> 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), 	<p><i>Beispiele:</i> reguläre Grammatik für Wörter mit ungerader Parität, Grammatik für Wörter, die bestimmte Zahlen repräsentieren, Satzgliederungsgrammatik</p> <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.III (Download LK-Q2-III.2)</p>

<p>6. Grenzen endlicher Automaten</p>	<ul style="list-style-type: none"> • 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). 	<p><i>Beispiele:</i> Klammerausdrücke, $a^n b^n$ im Vergleich zu $(ab)^n$</p> <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.III (Download LK-Q2-III.3)</p>
<p>7. Entwicklung eines Kellerautomaten als Antwort auf die Grenzen endlicher Automaten</p> <p>(a) Erweiterung eines DEA um eine einzelne Speichervariable zum Zählen von Eingabezeichen (z.B. Klammern) und Problematisierung dieses Ansatzes</p> <p>(b) Entwicklung eines Automaten mit Kellerspeicher</p> <p>(c) Anwendung eines Kellerautomaten zur Syntaxüberprüfung auf Grundlage von nicht-regulären Grammatiken</p> <p>(d) Implementierung eines Kellerautomaten zur Syntaxüberprüfung (Backtracking)</p>		<p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.III (Download LK-Q2-III.4)</p>



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 Konstruktionsprache 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
<p>1. Von-Neumann-Architektur und die Ausführung maschinennaher Programme</p> <p>(a) prinzipieller Aufbau einer von Neumann-Architektur mit CPU, Rechenwerk, Steuerwerk, Register und Hauptspeicher</p> <p>(b) einige maschinennahe Befehle und ihre Repräsentation in einem Binär-Code, der in einem Register gespeichert werden kann</p> <p>(c) Analyse und Erläuterung der Funktionsweise eines einfachen maschinennahen Programms</p>	<p>Die Schülerinnen und Schüler</p> <ul style="list-style-type: none"> • 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), 	<p><i>Beispiel:</i> Addition von 4 zu einer eingegeben Zahl mit einem Rechnermodell</p> <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.IV (Download LK-Q2-IV.1)</p>
<p>2. Simulation der Phasen eines Compilers</p> <p>(a) Prozesse beim Compiler:</p> <ul style="list-style-type: none"> • scannen • parsen • übersetzen/interpretieren <p>(b) Arten von Fehlern:</p> <ul style="list-style-type: none"> • lexikalischer Fehler • syntaktischer Fehler • semantischer Fehler <p>(c) Einordnung der neuen Begriffe in den Gesamtkontext der formalen Sprachen</p> <ul style="list-style-type: none"> • Automaten • Grammatiken • Sprachen 	<ul style="list-style-type: none"> • 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 Pro- 	<p><i>Beispiel:</i> Scanner, Parser und Interpreter einer Konstruktionsprache für geometrische Figuren Anhand einer einführenden Folienpräsentation werden die Begrifflichkeiten definiert.</p> <p>Ein kleiner Ausschnitt einer Pseudo-Programmiersprache zur Konstruktion von Zeichnungen wird betrachtet. Anfänglich besteht der Sprachumfang aus Programmen mit lediglich einer einzigen Zuweisung.</p> <p>Die grundlegenden Schritte eines Compilers werden am Beispiel dieser Grammatik nachvollzogen.</p> <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.IV (Download LK-Q2-IV.2)</p>

<p>3. Die Schritte eines Compilers</p> <p>(a) Scanner:</p> <ul style="list-style-type: none"> • 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 <p>(b) Parser:</p> <ul style="list-style-type: none"> • reguläre (oder wahlweise kontextfreie) Grammatik als Grundlage • Vorgabe einer Grundversion des Parsers • Erweiterung des Sprachumfangs • Implementierung der Parsermethoden für die Produktionsregeln der kontextfreien Grammatik <p>(c) Interpreter</p> <ul style="list-style-type: none"> • Vorgabe einer Grundversion des Interpreters • Erweiterung des Sprachumfangs • Implementierung 	<p>grammiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I),</p> <ul style="list-style-type: none"> • 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), • modifizieren Grammatiken regulärer und 	<p><i>Beispiel:</i> 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.</p> <p>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.</p> <p>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.</p> <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.IV (Download LK-Q2-IV.3)</p>
---	---	---

	<p>kontextfreier Sprachen (M),</p> <ul style="list-style-type: none">• 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
<p>1. Projekteinstieg / -planung (Spielauswahl, Zeitmanagement)</p> <p>2. Projektumsetzung (informatische Analyse, Modellierung, Implementierung, Test → ggf. in Zyklen)</p> <p>2.1. <i>Analyse des Spiels mit dem Ziel einer späteren informatischen Umsetzung als Netzwerkspiel</i></p> <p>(a) Spielen des Spiels in mehreren Kleingruppen in Form eines Rollenspiels nach vereinbarten Regeln:</p> <p>i. Ein Schüler oder eine Schülerin übernehmen die Rolle des Spielleiters oder der Spielleiterin.</p> <p>ii. Ein weiteres Gruppenmitglied protokolliert präzise die Kommunikation zwischen der Spielleiterin bzw. dem Spielleiter und den Spielerinnen und Spielern.</p> <p>(b) Formalisierung des Spielablaufs</p> <p>2.2. <i>Modellierung und Implementierung des Spiels als Netzwerkanwendung</i></p> <p>(a) Entwurf eines Protokolls zur Kommunikation zwischen Server und -client basierend auf den erarbeiteten Spielregeln</p>	<p>Die Schülerinnen und Schüler</p> <ul style="list-style-type: none"> • 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“ 	<p><i>Beispiel:</i> 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.</p> <p>Die beiden Spielerinnen bzw. Spieler färben abwechselnd einen noch ungefärbten Knoten. Dabei muss beachtet werden, dass kein adjazenter Knoten mit derselben Farbe markiert wurde.</p> <p>Es verliert derjenige Spielerin bzw. derjenige Spieler, die bzw. der unter diesen Bedingungen keinen Zug mehr machen kann. Sind alle Knoten korrekt gefärbt, endet das Spiel unentschieden.</p> <p><i>Materialien:</i> Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2-V (LK) (Download LK-Q2-V.1)</p>

<p>(b) Entwicklung von Entwurfs- und Implementationsdiagrammen für den Spielservers</p> <p>(c) Implementation und Test der Spielservers-Klasse und von dieser benötigter Fachklassen</p> <p>(d) Modellierung, Implementierung und Test des Spielclients</p> <p>(e) Test der Zusammenarbeit von Spielservers und Spielclient</p> <p>2.3. Reflexion des Softwareprodukts</p> <p>(a) Identifikation der Stufen eines Softwareentwicklungszyklus</p> <p>(b) Wiederholende Darstellung der Entwicklungsschritte zum fertigen Produkt</p> <p>(c) Mögliche Erweiterungen:</p> <ol style="list-style-type: none"> 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 <p>3. Projektreflexion (Reflexion der Projektplanung, Präsentation)</p>	<p>und „Backtracking“ (M),</p> <ul style="list-style-type: none"> • 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). 	
--	--	--

