Konkretisierte Unterrichtsvorhaben: Informatik in der Qualifikationsphase (Stand 23.06.2015)

Dieser Lehrplan hat den Anspruch, alle im Kernlehrplan geforderten Kompetenzen abzubilden.

Die Umsetzung des Lehrplans folgt pro Unterrichtsvorhaben jeweils auf einer Übersichts- und einer dazugehörigen Konkretisierungsebene. Die Übersichtsebene ist für die Kolleginnen und Kollegen verpflichtend, die hier dargestellten Konkretisierungen können anhand von Arbeitsblättern, Projekten und Aufgaben umfassend umgesetzt werden.

Didaktische Lernumgebung

Die Unterrichtsvorhaben in Informatik hängen in ihrer konkreten Umsetzung von der gewählten Programmierumgebung ab.

In diesem Lehrplan wird als Lernumgebung zur Implementation *BlueJ*, *JavaEditor* und/oder *Eclipse* vorgeschlagen, wobei auch andere (möglichst auf allen üblichen Plattformen) frei verfügbare Entwicklungsumgebungen für Java denkbar sind.
Die Verwendung von wenigstens zwei der obigen Entwicklungsumgebungen innerhalb der Qualifikationsphase ist empfehlenswert.

Als Lern-/Lehrplattform zum Austausch von Unterrichtsmaterialien, Aufgaben und zur Kommunikation wird die schulinterne Moodle-Plattform intensiv genutzt.

Kommunizieren und Kooperieren

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) [im Grundkurs],
- Leistungskurs: nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Daten zur Organisation von Arbeitsabläufen sowie zur Verteilung und Zusammenführung von Arbeitsanteilen (K) [im Leistungskurs],
- 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).

Übersichtsraster Unterrichtsvorhaben in der Qualifikationsphase - Q1-Grundkurs – Teil 1

<table>
<tr><th colspan="2">Qualifikationsphase – Q1-Grundkurs</th></tr>
<tr>
<td><u>Unterrichtsvorhaben Q1G-I ('Klassendiagramme')</u>

Thema: Wiederholung und Vertiefung der objektorientierten Modellierung

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

Inhaltsfelder:
- Daten und ihre Strukturierung
- Algorithmen

Inhaltliche Schwerpunkte:
- Objekte und Klassen

Zeitbedarf: 12 Stunden</td>
<td><u>Unterrichtsvorhaben Q1G-II ('Lineare Datenstrukturen')</u>

Thema: Organisation und Verarbeitung von Daten I – Modellierung und Implementierung von Anwendungen mit dynamischen und linearen Datenstrukturen

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

Inhaltsfelder:
- Daten und ihre Strukturierung
- Algorithmen

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

Zeitbedarf: 20 Stunden</td>
</tr>
<tr>
<td><u>Unterrichtsvorhaben Q1G-III ('Suchen & Sortieren')</u>

Thema: Algorithmen zum Suchen und Sortieren auf linearen Datenstrukturen

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

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: 18 Stunden</td>
<td><u>Unterrichtsvorhaben Q1G-IV ('Automaten')</u>

Thema: Automaten und formale Sprachen

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

Inhaltsfelder:
- Formale Sprachen und Automaten
- Informatiksysteme

Inhaltliche Schwerpunkte:
- Syntax und Semantik einer Programmiersprache
- Endliche Automaten
- Grammatiken regulärer Sprachen
- Möglichkeiten und Grenzen von Automaten und formalen Sprachen

Zeitbedarf: 14 Stunden</td>
</tr>
</table>

Übersichtsraster Unterrichtsvorhaben in der Qualifikationsphase - Q1-Grundkurs – Teil 2

<table>
<tr><th colspan="2">Qualifikationsphase – Q1-Grundkurs</th></tr>
<tr><td rowspan="3"><u>Unterrichtsvorhaben Q1G-V ('Technische Informatik')</u>

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: 10 Stunden</td><td></td></tr>
<tr><td>Summe Qualifikationsphase 1
Grundkurs: 74 Stunden</td></tr>
<tr><td></td></tr>
</table>

Übersichtsraster Unterrichtsvorhaben in der Qualifikationsphase – Q2-Grundkurs

<table>
<tr><th colspan="2">Qualifikationsphase – Q2-Grundkurs</th></tr>
<tr><td><u>Unterrichtsvorhaben Q2G-I ('Bäume')</u>

Thema: Organisation und Verarbeitung von Daten II – Modellierung und Implementierung von Anwendungen mit dynamischen nicht-linearen Datenstrukturen

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

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

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

Zeitbedarf: 20 Stunden</td><td><u>Unterrichtsvorhaben Q2G-II ('Netzwerke & Krypto')</u>

Thema: Aufbau von und Kommunikation in Netzwerken

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: 16 Stunden</td></tr>
<tr><td rowspan="3"><u>Unterrichtsvorhaben Q2G-III ('Datenbanken')</u>

Thema: Nutzung und Modellierung 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</td><td></td></tr>
<tr><td>Summe Qualifikationsphase 2 Grundkurs: 56 Stunden</td></tr>
<tr><td></td></tr>
</table>

Übersichtsraster Unterrichtsvorhaben in der Qualifikationsphase - Q1-Leistungskurs – Teil 1

<table>
<tr><th colspan="2">Qualifikationsphase – Q1-Leistungskurs</th></tr>
<tr>
<td><u>Unterrichtsvorhaben Q1L-I ('Klassendiagramme')</u>

Thema: Wiederholung und Vertiefung der objektorientierten Modellierung

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

Inhaltsfelder:
- Daten und ihre Strukturierung
- Algorithmen

Inhaltliche Schwerpunkte:
- Objekte und Klassen

Zeitbedarf: 15 Stunden</td>
<td><u>Unterrichtsvorhaben Q1L-II ('Lineare Datenstrukturen')</u>

Thema: Organisation und Verarbeitung von Daten I – Modellierung und Implementierung von Anwendungen mit dynamischen und linearen Datenstrukturen

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

Inhaltsfelder:
- Daten und ihre Strukturierung
- Algorithmen

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

Zeitbedarf: 30 Stunden</td>
</tr>
<tr>
<td><u>Unterrichtsvorhaben Q1L-III ('Suchen & Sortieren')</u>

Thema: Algorithmen zum Suchen und Sortieren auf linearen Datenstrukturen

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

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: 25 Stunden</td>
<td><u>Unterrichtsvorhaben Q1L-IV ('(Keller-)Automaten')</u>

Thema: Automaten und formale Sprachen

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

Inhaltsfelder:
- Formale Sprachen und Automaten
- Informatiksysteme

Inhaltliche Schwerpunkte:
- Syntax und Semantik einer Programmiersprache
- Endliche Automaten und Kellerautomaten
- Grammatiken regulärer und kontextfreier Sprachen
- Möglichkeiten und Grenzen von Automaten und formalen Sprachen
- Scanner, Parser und Interpreter für eine reguläre Sprache

Zeitbedarf: 30 Stunden</td>
</tr>
</table>

Übersichtsraster Unterrichtsvorhaben in der Qualifikationsphase - Q1-Leistungskurs – Teil 2

<table>
<tr><th colspan="2">Qualifikationsphase – Q1-Leistungskurs</th></tr>
<tr><td rowspan="3"><u>Unterrichtsvorhaben Q1L-V ('Bäume')</u>

Thema: Organisation und Verarbeitung von Daten II – Modellierung und Implementierung von Anwendungen mit dynamischen nicht-linearen Datenstrukturen (Bäume)

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

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

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

Zeitbedarf: 25 Stunden</td><td></td></tr>
<tr><td>Summe Qualifikationsphase 1
Leistungskurs: 125 Stunden</td></tr>
<tr><td></td></tr>
</table>

Übersichtsraster Unterrichtsvorhaben in der Qualifikationsphase – Q2-Leistungskurs

<table>
<tr><th colspan="2">Qualifikationsphase – Q2-Leistungskurs</th></tr>
<tr><td><u>Unterrichtsvorhaben Q2L-I ('Graphen')</u>

Thema: Organisation und Verarbeitung von Daten III – Modellierung und Implementierung von Anwendungen mit dynamischen nicht-linearen Datenstrukturen (Graphen)

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

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

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

Zeitbedarf: 25 Stunden</td><td><u>Unterrichtsvorhaben Q2L-II ('Netzwerke & Krypto')</u>

Thema: Aufbau von und Kommunikation in Netzwerken

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: 25 Stunden</td></tr>
<tr><td><u>Unterrichtsvorhaben Q2L-III ('Datenbanken')</u>

Thema: Nutzung und Modellierung 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: 25 Stunden</td><td><u>Unterrichtsvorhaben Q2L-IV ('Technische Informatik')</u>

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: 15 Stunden

Summe Qualifikationsphase 2 Leistungskurs: 90 Stunden</td></tr>
</table>

Unterrichtsvorhaben Q1G-I

Thema: Wiederholung und Vertiefung der objektorientierten Modellierung

Leitfragen:
Wie wird aus einem anwendungsbezogenen Sachkontext ein informatisches Klassenmodell entwickelt? Wie werden Attribute, Methoden und Beziehungen identifiziert, den Klassen zugeordnet und dargestellt?

Vorhabenbezogene Konkretisierung:
Der bereits bekannte objektorientierte Zugang zu informatischer Modellierung wird von einer allgemeinen Betrachtung dieses informatischen Konzepts auf eine konkrete Problematik übertragen. Anhand dieser wird eine anwendungsbezogene Implementation Schritt für Schritt von der Objektidentifikation über das Entwurfs- und Implementationsdiagramm durchlaufen.
Grundlegende Modellierungskonzepte wie Sichtbarkeiten, Assoziationen, Vererbung sowie deren Darstellung in Entwurfs- und Klassendiagrammen und Dokumentationen werden wiederholt. Ebenso wird erneut die grafische Darstellung von Objektkommunikation thematisiert.
Anhand von Gütekriterien und Eigenschaften von Modellierung entwickeln und bewerten die Schülerinnen und Schüler Klassenentwürfe.
Das Konzept der objektorientierten Modellierung wird um die Idee der abstrakten Klasse und des Interfaces erweitert.

Zeitbedarf: 12 Stunden

Sequenzierung des Unterrichtsvorhabens: siehe nächste Seite

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Aufgaben und Materialien
1. Wiederholung der grundlegenden Konzepte der objektorientierten Programmierung a) Sichtweise der objektorientierten Informatik auf die Welt b) OOP als informatikspezifische Modellierung der Realität c) Schritte der Softwareentwicklung	Die Schülerinnen und Schüler … - analysieren und erläutern objektorientierte Modellierungen (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), - nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), - wenden eine didaktisch orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I), - stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D), - dokumentieren Klassen (D), - stellen die Kommunikation zwischen Objekten grafisch dar (D)	Objekte in der Realität vs. Objekte in Java: Eigenschaften und Fähigkeiten realer Objekte (Hund, Haus, Auto,...) in Java-Klassen übersetzen
2. Erweiterung der objektorientierten Programmierung a) Umsetzung einer Anforderung in Entwurfs- und Klassendiagramm b) Abstrakte Klassen und Interfaces b) Objektkommunikation im Sequenzdiagramm c) Klassendokumentation d) Umsetzung von Teilen der Modellierung e) Wiederholung/Vertiefung der Datenstruktur Array, Eigenschaften der Datenstruktur, Standardoperationen für ein- und zweidimensionale Arrays		Modellierung eines Autohauses:(abstrakte) Fahrzeug-Klasse mit Unterklassen; begrenzter Stellplatz für Fahrzeuge (→ Fahrzeug-Array) Angeschlossene Werkstatt kann u.a. Fahrzeuge reparieren (aber auch einzelne Bestandteile von Fahrzeugen) (→ Interface "Reparierbar")
3. Übung und Vertiefung der OOM / OOP		Weitere Mini-Projekte modellieren und implementieren Prüfungsvorbereitung

Unterrichtsvorhaben Q1G-II

Thema:
Organisation und Verarbeitung von Daten I – Modellierung und Implementierung von Anwendungen mit dynamischen und linearen Datenstrukturen

Leitfragen:
Wie müssen Daten linear strukturiert werden, um in den gestellten Anwendungsszenarien eine beliebige Anzahl von Objekten verwalten zu können?

Vorhabensbezogene Konkretisierung:
Ausgehend von einigen Alltagsbeispielen werden als Erstes die Anforderungen an eine Datenstruktur erschlossen. Anschließend werden die Möglichkeiten des Arrays untersucht, lineare Daten zu verwalten und über deren Grenzen/Probleme die Vorteile einer dynamischen linearen Struktur am Beispiel der Struktur Queue erarbeitet (Anwendungskontext Warteschlange). Die Klasse *Queue* selbst wird vorgegeben, die Operationen erläutert. Zur Vertiefung der Kenntnisse wird ein weiteres Anwendungsszenario eingeführt, dessen Lösung modelliert und implementiert wird. Darauf folgt die Erarbeitung der Struktur Stack, die mithilfe eines einfachen Anwendungsszenarios eingeführt (bspw. Palindrome) wird. Auch hier wird die Klasse *Stack* selbst vorgegeben und die Operationen erläutert. Weitere Aufgaben dienen der Vertiefung und Sicherung.

Um die Unterschiede der beiden Prinzipien FIFO und LIFO zu verstehen, werden zur Lösung der Aufgaben sowohl der Stack als auch die Queue benötigt.

Als letzte lineare dynamische Datenstruktur wird die Liste eingeführt. In dieser Sequenz liegt der Fokus auf der Möglichkeit, auf jedes Element zugreifen zu können. Nachdem die umfangreicheren Standardoperationen dieser Datenstruktur in einem einführenden Beispiel (bspw. Vokabeltrainer) erarbeitet und in einem weiteren Beispiel vertieft wurden, werden abschließend in einem Anwendungskontext verschiedene lineare Datenstrukturen angewendet. Die Modellierung erfolgt beim gesamten Vorhaben in Entwurfs- und Implementationsdiagrammen.

Zeitbedarf: 20 Stunden

Sequenzierung des Unterrichtsvorhabens: siehe nächste Seite

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Aufgaben und Materialien
1. Die Datenstruktur **Schlange** a) Nachteile von Arrays; Modellierung und Implementierung der dynamischen Verknüpfung von Objekten b) Generische Typen, Trennung von Verwaltung und Inhalt dyn. DS. c) Erläuterung von Problemstellungen, die nach dem FIFO-Prinzip bearbeitet werden d) Funktionalität der Schlange unter Verwendung der Klasse *Queue*; Erschließen der Standardoperationen e) Modellierung und Implementierung einer Anwendung auf der Basis einer Anforderungsbeschreibung mit Objekten der Klasse *Queue* Optional: f) Einfache graphische Benutzeroberflächen erstellen g) Das Model-View-Controller-Konzept in der Praxis	Die Schülerinnen und Schüler… - 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), - interpretieren Fehlermeldungen und korrigieren den Quellcode (I), - stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D) - erläutern Operationen dynamischer (linearer und nicht-linearer) Datenstrukturen (A), - modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M) - ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M) - dokumentieren Klassen (D) - implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I)	Wartezimmer-Projekt: Die Reihenfolge des Eintreffens von Patienten im Wartezimmer einer Arztpraxis soll verwaltet werden. (→ FIFO; Schlange) Optional: Für die Verwaltung soll eine einfache graphische Benutzeroberfläche erstellt werden (→ Java-Swing) Das Design-Pattern "Model-View-Controller" soll verwendet werden, um den Aufbau des Programms zu strukturieren. (siehe Informatik 2, Schoeningh)
3. Die Datenstruktur **Stapel** a) Erläuterung von Problemstellungen, die nach dem LIFO-Prinzip bearbeitet werden b) Funktionalität der Klasse Stapel unter Verwendung der Klasse *Stack* c) Modellierung und Implementierung einer Anwendung auf Basis einer Anforderungsbeschreibung mit Objekten der Klasse *Queue* d) Modellierung und Implementierung einer Anwendung unter Verwendung verschiedener Datenstrukturen (Objekte der Klassen *Queue*, *Stack* und *Array (Palindrom)*)		Klammern-Tester-Projekt: Mithilfe eines Stapels sollen Klammernterme (Bspw. "((())())" oder "[{()<>}]") auf Korrektheit überprüft werden. Optional: Auf Basis der GUI aus dem Wartezimmer-Projekt lässt sich dieses Projekt auch zur Übung für Benutzeroberflächen einsetzen. Weitere Projektidee: "Palindrome" Handelt es sich bei übergebenen Wörtern um Palindrome oder nicht.
4. Die Datenstruktur **Liste** a) Analyse der Möglichkeiten bisheriger Datenstrukturen zwecks Bestimmung notwendiger Funktionalitäten für komplexere Anwendungen (Abgrenzung zu *Stack*/*Queue*, zusätzliche		Projektideen: - Verwaltung einer Musik-Sammlung (CDs) - Vokabeltrainer (Vokabelkarten) - Ski-Abfahrtslauf (versetze

Fähigkeiten der Klasse *List*) b) Erarbeitung der Funktionalität der Liste unter Verwendung der Klasse *List* c) Modellierung und Implementierung einer Anwendung mit Objekten der Klasse *List* d) Modellierung und Implementierung einer Anwendung unter Verwendung verschiedener Datenstrukturen (Stack, Queue, List)		Starts; Läufer nach Laufzeit einsortieren)
5. Übungen und Vertiefungen zur Verwendung linearer und dynamischer Datenstrukturen anhand weiterer Problemstellungen		Projektidee: "Ein Array voller Listen – Eine Liste voller Arrays" Prüfungsvorbereitung

Unterrichtsvorhaben Q1G-III

Thema: Algorithmen zum Suchen und Sortieren auf linearen Datenstrukturen

Leitfragen:
Nach welchen Grundprinzipien können Algorithmen strukturiert werden? Welche Qualitätseigenschaften sollten Algorithmen erfüllen? Wie können mithilfe von Such- und Sortieralgorithmen Daten in linearen Strukturen effizient (wieder-)gefunden werden?

Vorhabenbezogene Konkretisierung:
Zunächst werden anhand eines Anwendungsbeispiels übergreifende Algorithmuseigenschaften (wie Korrektheit, Effizienz und Verständlichkeit) erarbeitet und Schritte der Algorithmusentwicklung wiederholt. Dabei kommen Struktogramme zur Darstellung von Algorithmen zum Einsatz.
Als besondere Struktur von Algorithmen wird die Rekursion an Beispielen veranschaulicht und gegenüber der Iteration abgegrenzt. Rekursive Algorithmen werden von den Schülerinnen und Schülern analysiert und selbst entwickelt.
In der zweiten Unterrichtssequenz geht es um die Frage, wie Daten in linearen Strukturen (lineare Liste und Array) (wieder-)gefunden werden können. Die lineare Suche als iteratives und die binäre Suche als rekursives Verfahren werden veranschaulicht und implementiert. Die Bewertung der Algorithmen erfolgt, indem jeweils die Anzahl der Vergleichsoperationen und der Speicherbedarf ermittelt wird.
Möchte man Daten effizient in einer linearen Struktur wiederfinden, so rückt zwangsläufig die Frage nach einer Sortierstrategie in den Fokus. Es wird mindestens ein iteratives und ein rekursives Sortierverfahren erarbeitet und implementiert sowie ihre Effizienz bewertet.

Zeitbedarf: 18 Stunden

Sequenzierung des Unterrichtsvorhabens: siehe nächste Seite

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Aufgaben und Materialien
1. Eigenschaften von Algorithmen a) Qualitätseigenschaften von Algorithmen b) Strukturierung von Algorithmen mit Hilfe der Strategien „Modularisierung“ und „Teile und Herrsche“; Darstellung durch Struktogramme c) Analyse und Entwicklung von rekursiven Algorithmen	Die Schülerinnen und Schüler - analysieren und erläutern Algorithmen und Programme (A), - modifizieren Algorithmen und Programme (I), - stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D), - entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“ und „Teilen und Herrschen“ (M), - implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), - testen Programme systematisch anhand von Beispielen (I). - implementieren und erläutern iterative und rekursive Such- und Sortierverfahren (I), - beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A), - beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), - nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), - interpretieren Fehlermeldungen und korrigieren den Quellcode (I)	Projektideen: - ggT - Fibonacci-Zahlen - Türme von Hanoi
2. Suchen in Listen und Arrays a) Lineare Suche in Listen und Arrays b) Binäre Suche in einem Array c) Untersuchung der beiden Verfahren bzgl. Laufzeit und Speicherplatzbedarf		Lineare Suche in Listen und Arrays wurde schon in Q1G-II.4 bzw. Q1G-I.2 behandelt und muss hier nur kurz aufgefrischt werden.
3. Sortieren auf Listen und Arrays a) Entwicklung und Implementierung eines iterativen Sortierverfahrens für eine Liste (bspw. Bubble- oder Min-Sort) b) Entwicklung und Implementierung eines rekursiven Sortierverfahrens für ein Array (bspw. Merge- oder Quicksort) c) Untersuchung der beiden Verfahren bzgl. Laufzeit und Speicherplatzbedarf (Anzahl Vergleiche/Vertauschungen) Optional: d) Laufzeiten im O-Kalkül		Zunächst Sortieren ganzer Zahlen. Möglicher Wiederaufgriff der CD-Verwaltung oder des Vokabeltrainers aus Q1G-II.4 (→ CDs bzw. Vokabeln nach bestimmten Kriterien sortieren lassen)

Unterrichtsvorhaben Q1G- IV

Thema: Automaten und formale Sprachen

Leitfragen:
Wie lassen sich reale Automaten durch ein Modell formal beschreiben? Wie kann die Art und Weise, wie ein Computer Zeichen (Eingaben) verarbeitet, durch Automaten dargestellt werden? Welche Eigenschaften besitzen Automaten und was können sie leisten? Wie werden sie dargestellt? Wie werden reguläre Sprachen durch eine Grammatik beschrieben? In welchem Verhältnis stehen endliche Automaten und Grammatiken? Welche Anwendungsfälle können durch endliche Automaten und Grammatiken regulärer Sprachen beschrieben werden und welche nicht?

Vorhabensbezogene Konkretisierung:
Ausgehend von der Beschreibung und Untersuchung realer Automaten wird das formale Modell eines endlichen Automaten entwickelt. Neben dem Mealy-Automaten geht es vor allem um den erkennenden endlichen Automaten. Auf die Erarbeitung der Beschreibung folgt die Modellierung eigener Automaten und die Untersuchung bestehender, um die Eigenschaften und Grenzen eines endlichen Automaten zu erkennen. Hierbei wird dessen Verhalten auf bestimmte Eingaben analysiert.
An den Themenkomplex *Endliche Automaten* schließt sich die Erarbeitung von Grammatiken regulärer Sprachen an. Die Untersuchung beginnt bei die Erschließung der formalen Beschreibung und wird mit der Entwicklung von Grammatiken zu regulären Sprachen fortgeführt. Hierbei wird auch die Beziehung von Grammatiken regulärer Sprachen zu endlichen Automaten an Beispielen erarbeitet und analysiert. Hierzu gehört auch die Untersuchung, welche Problemstellungen durch endliche Automaten und reguläre Grammatiken beschrieben werden können und welche nicht.

Zeitbedarf: 14 Std.

Sequenzierung des Unterrichtsvorhabens: siehe nächste Seite

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Aufgaben und Materialien
1. Endliche Automaten a) Erarbeitung der formalen Beschreibung eines Mealy-Automaten und der Darstellungsformen b) Erarbeitung der formalen Beschreibung eines deterministischen endlichen Automaten (DEA) als 5-Tupel sowie dessen graphische Darstellung; Erschließung der Fachbegriffe Alphabet, Wort, (akzeptierte) Sprache (Darstellung durch reg. Ausdrücke und Mengenschreibweise), Determinismus c) Analyse der Eigenschaften von DEA durch die Modellierung eines Automaten zu einer gegebenen Problemstellung, der Modifikation eines Automaten sowie die Überführung der gegebenen Darstellungsform in eine andere d) Grenzen von DEA	Die Schülerinnen und Schüler - analysieren und erläutern die Eigenschaften endlicher Automaten einschließlich ihres Verhaltens bei bestimmten Eingaben (A) - ermitteln die Sprache, die ein endlicher Automat akzeptiert (D) - entwickeln und modifizieren zu einer Problemstellung endlicher Automaten (M) - stellen endliche Automaten in Tabellen und Graphen dar und überführen sie in die jeweils andere Darstellungsform (D) - entwickeln zur Grammatik einer regulären Sprache einen zugehörigen endlichen Automaten (M) - analysieren und erläutern Grammatiken regulärer Sprachen (A) - modifizieren Grammatiken regulärer Sprachen (M) - entwickeln zu einer regulären Sprache eine Grammatik, die die Sprache erzeugt (M) - entwickeln zur akzeptierten Sprache eines Automaten eine zugehörige Grammatik (M) - beschreiben an Beispielen den Zusammenhang zwischen Automaten und Grammatiken (D) - zeigen die Grenzen endlicher Automaten und regulärer Grammatiken im Anwendungszusammenhang auf (A) - ermitteln die formale Sprache, die durch eine Grammatik erzeugt wird (A)	Modellierung eines Cola-Automaten DEA zu verschiedenen Sprachen erstellen. Bspw.: $L(A) = \{a^n b \mid n \in \mathbb{N}\}$ $L(A) = \{awb \mid w \in \{a,b\}^*\}$ $L(A) = \{w \mid w \in \{a,b\}^*$ $und\ w\ enthält$ $eine\ gerade\ Anzahl\ a's\}$ Sprachen vorgegebener DEA herausfinden $L(A) = \{a^n b^n \mid n \in \mathbb{N}\}$
2. Grammatiken regulärer Sprachen a) Erarbeitung der formalen Beschreibung einer regulären Grammatik; Zusammenhang zu erzeugter Sprache b) Analyse der Eigenschaften einer regulären Grammatik durch deren Entwicklung und Modellierung zu einer gegebenen Problemstellung.		Grammatiken regulärer Sprachen (wie Beispiele s.o.)
3. Übungen und Vertiefungen Verwendung endlicher Automaten und Grammatiken regulärer Sprachen Optional: Kellerautomaten – Konzept und Funktionsweise		Prüfungsvorbereitung $L(A) = \{a^n b^n \mid n \in \mathbb{N}\}$ mit Kellerautomaten akzeptieren Palindrome erkennen

Unterrichtsvorhaben Q1G-V:

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. Mithilfe theoretischer Überlegungen zur Mächtigkeit von Methoden wird bewiesen, dass es unmöglich ist, ein Informatiksystem zu entwickeln, dass für jedes beliebige Computerprogramm und jede beliebige Eingabe entscheidet ob das Programm mit der Eingabe terminiert oder nicht (Halteproblem). Anschließend werden Vor- und Nachteile der Grenzen der Automatisierbarkeit angesprochen und der Einsatz von Informatiksystemen hinsichtlich prinzipieller Möglichkeiten und prinzipieller Grenzen beurteilt.

Zeitbedarf: 10 Stunden

Sequenzierung des Unterrichtsvorhabens: siehe nächste Seite

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Medien oder Materialien
1. Von-Neumann-Architektur und die Ausführung maschinennaher Programme a) prinzipieller Aufbau einer von Neumann-Architektur mit CPU, Rechenwerk, Steuerwerk, Register und Hauptspeicher b) Analyse und Erläuterung der Funktionsweise eines einfachen maschinennahen Programms c) Schreiben eigener einfacher Programme	Die Schülerinnen und Schüler - erläutern die Ausführung eines einfachen maschinennahen Programms sowie die Datenspeicherung auf einer „Von-Neumann-Architektur“ (A), - untersuchen und beurteilen Grenzen des Problemlösens mit Informatiksystemen (A).	Bspw. "Know-How-Rechner" zur Simulation maschinennahen Programmierens Bspw.: - Addieren - Multiplizieren - Sortieren
2. Grenzen der Automatisierbarkeit a) Vorstellung des Halteproblems b) Unlösbarkeit des Halteproblems c) Beurteilung des Einsatzes von Informatiksystemen hinsichtlich prinzipieller Möglichkeiten und prinzipieller Grenzen		Beispiel: Halteproblem inkl. Beweis der Unlösbarkeit Praktische Berechenbarkeit für wichtige Problemstellungen (NP-schwere Probleme): - Travelling-Salesman - Rucksackproblem

Unterrichtsvorhaben Q2G-I

Thema: Organisation und Verarbeitung von Daten II – Modellierung und Implementierung von Anwendungen mit dynamischen nicht-linearen Datenstrukturen

Leitfragen:
Wie können Daten mithilfe von Baumstrukturen verwaltet werden? Wie können mit binären Suchbäumen Inhalte sortiert verwaltet werden und welche Vor- und Nachteile bietet dies?

Vorhabenbezogene Konkretisierung:
Anhand des Anwendungskontextes Spielbäume werden zunächst der generelle Aufbau von Baumstrukturen (auch nicht-binäre) und wichtige Grundbegriffe erarbeitet. Die Darstellung von Bäumen mit Knoten und Kanten wird eingeführt.

Anschließend rückt der Fokus auf die binären Bäume, deren rekursiver Aufbau für die Traversierung der Datenstruktur genutzt wird. Die Preorder-Traversierung wird verwendet, um einen gespeicherten Inhalt in einem Binärbaum zu finden (Tiefensuche).
Der Anwendungskontext Ahnenbaum wird mithilfe der Klasse BinaryTree (der Materialien für das Zentralabitur in NRW) modelliert und (ggf. in Teilen) implementiert. Dabei wird u. a. die Erzeugung eines Binärbaums mithilfe der beiden Konstruktoren der Klasse BinaryTree thematisiert.

Möchte man Daten geordnet speichern, so bietet sich die Struktur des binären Suchbaums an. An Beispielen wird zunächst das Prinzip des binären Suchbaums erarbeitet. Die Operationen des Suchens, Einfügens, Löschens und der sortierten Ausgabe werden thematisiert.
Um Daten in einem Anwendungskontext mithilfe eines binären Suchbaums verwalten zu können, müssen sie in eine Ordnung gebracht werden können, d. h. sie müssen vergleichbar sein. Diese Vorgabe wird mithilfe des Interfaces Item realisiert, das alle Klassen, dessen Objekte in einem Suchbaum verwaltet werden sollen, implementieren müssen. Auf diese Weise wird ein Anwendungskontext (Benutzerverwaltung) mithilfe der Klassen BinarySearchTree und Item modelliert und (ggf. in Teilen) implementiert.

Zeitbedarf: 20 Stunden

Sequenzierung des Unterrichtsvorhabens: siehe nächste Seite

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Aufgaben und Materialien
1. Aufbau von Baumstrukturen und Grundbegriffe a) Erarbeitung der Begriffe Wurzel, Knoten, Blatt, Kante, Grad eines Knotens und eines Baumes, Pfad, Tiefe, Ebene, Teilbaum b) Aufbau und Darstellung von Baumstrukturen in verschiedenen Anwendungskontexten	Die Schülerinnen und Schüler - stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D), - erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), - analysieren und erläutern Algorithmen und Programme (A), - stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D). - beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), - ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), - ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M), - modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M), - verwenden bei der Modellierung geeigneter Problemstellungen die Möglichkeiten der Polymorphie (M), - entwickeln iterative und rekursive Algorithmen unter Nutzung der Konstruktionsstrategien „Modularisierung“ und „Teilen und Herrschen“ (M), - implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), - modifizieren Algorithmen und Programme (I), - nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), - interpretieren Fehlermeldungen und korrigieren den Quellcode (I), - testen Programme systematisch anhand von Beispielen (I),	Möglicher Zugang: Spielbäume für einfaches Zweispielerspiel (bspw. "Würfeldrehen" → Ein Würfel wird abwechselnd auf eine neue Seite gekippt und die Augenzahl zur eigenen addiert. Gewonnen hat, wer auf eine vorher festgelegte Punktzahl exakt erreicht.)
2. Binäre Bäume a) rekursiver Aufbau eines binären Baums b) Traversierungen (pre-, in-, postorder) c) Modellierung eines Binärbaums in einem Anwendungskontext mit Hilfe der Klasse BinaryTree (als Entwurfs- und Implementationsdiagramm) d) Implementation einer Anwendung der Datenstruktur binärer Baum (ggf. in Teilen)		Projektideen: - Ahnenbaum aufbauen und Verwandtschaftsbeziehungen auswerten - Codierungsbäume: bspw. Morsezeichen als innere Knoten codieren, in Blättern stehen Buchstaben
3. Binäre Suchbäume a) Prinzip des binären Suchbaums, Ordnungsrelation b) Operationen auf dem binären Suchbaum (Suchen, Einfügen, Löschen, sortierte Ausgabe) c) Modellierung eines binären Suchbaums in einem Anwendungskontext mit Hilfe der Klasse BinarySearchTree (als Entwurfs- und Implementationsdiagramm) und dem Interface Item d) Implementation einer Anwendung der Datenstruktur binärer Suchbaum (ggf. in Teilen)		Projektideen: - "Schülerverwaltung" (→ Sortierung bspw. nach Name) - CD-Verwaltung (Rückgriff auf Q1G-II.4) -…
4. Übung und Vertiefungen der Verwendung von Binärbäumen oder binären Suchbäumen anhand weiterer Problemstellungen		Prüfungsvorbereitung

Unterrichtsvorhaben Q2G-II

Thema: Aufbau von und Kommunikation in Netzwerken

Leitfragen:
Was macht menschliche Kommunikation aus? Welchen Stellenwert haben technische/ informatische Hilfsmittel für die Kommunikation? Wie werden Daten in einem Netzwerk zwischen den Kommunikationspartnern übertragen? Wie ist die Arbeitsteilung in Netzwerken gestaltet? Wie kann sicher in Netzwerken kommuniziert werden?

Vorhabenbezogene Konkretisierung:
Ausgehend von alltäglicher Face-to-Face-Kommunikation werden die Grundprinzipien sowie die Bewertungskriterien von Kommunikation erläutert. Das Netzwerk wird als vorteilhafte Kommunikationsstruktur dargestellt und anhand von Topologien kategorisiert. Ausgehend davon wird der Protokollbegriff entwickelt und anhand des TCP/IP-Schichtenmodells analysiert. Anschließend wird das Client-Server-Prinzip vorgestellt und angewandt.
Der Aufbau und die Administration von Netzwerken wird mithilfe der Software Filius erarbeitet und dabei IP-Adressbereiche, Kommunikationshardware und Client-/Server-Software kennengelernt, eingerichtet und getestet.
Sichere Kommunikation in Netzen ist nur dank kryptografischer Verfahren möglich. Stellvertretend werden zwei symmetrische und ein asymmetrisches Verfahren erläutert, angewandt und bewertet.

Zeitbedarf: 12 Stunden

Sequenzierung des Unterrichtsvorhabens: siehe nächste Seite

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Aufgaben und Materialien
1. Technische Kommunikation als Fortführung natürlicher Kommunikation a) Kommunikation im Shannon-Weaver-Modell b) Kriterien von technischen Kommunikationsarten c) Die Geschichte der technischen Kommunikation	Die Schülerinnen und Schüler - beschreiben und erläutern Topologien, die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken (A), - nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zum Erschließen, zur Aufbereitung und Präsentation fachlicher Inhalte (D), - analysieren und erläutern Eigenschaften und Einsatzbereiche symmetrischer und asymmetrischer Verschlüsselungsverfahren (A).	Projekt: Kommunikation im wilden Westen → Material von Daniel Garmann, Gymnasium Odenthal, Link: http://projekte.gymnasium-odenthal.de/informatik/ →Informatik → Jahrgangsstufe Q → 003 … → 13 Netzwerke → Demo
2. Aufbau von Netzwerken und Kommunikationsregeln a) Das Netzwerk als Organisationsprinzip der Kommunikation und Möglichkeiten der Ausformung b) Geregelte technische Kommunikation durch Protokolle in Schichtenmodellen		
3. Aufgabenteilung in Netzwerken durch Server und Client a) Aufbau und Aufgaben der Client-Server-Struktur b) Aufbau von Netzwerken mit Filius Verkabelung, Einrichtung von IP-Adressen, Routern, Client-/Serversoftware c) Protokolle zwischen Client und Server		Software "Filius": Rechnernetzwerke aufbauen, einrichten, Software installieren und testen Datenkommunikation zu verschiedenen Protokollen nachvollziehen (in Filius)
4. Kryptologie a) Veranschaulichen und Anwenden von symmetrischen und asymmetrischen kryptographischen Verfahren (Caesar, Vigenère, RSA) b) Bewertung der Verfahren hinsichtlich ihrer Sicherheit und ihrem Aufwand		Bspw. mithilfe von Webquests: http://www.matheprisma.uni-wuppertal.de/Module/Caesar/ und http://www.matheprisma.uni-wuppertal.de/Module/RSA/
5. Übung und Vertiefung des Aufbaus von und der Kommunikation in Netzwerken		Prüfungsvorbereitung

Unterrichtsvorhaben Q2G-III

Thema: Nutzung und Modellierung von relationalen Datenbanken in Anwendungskontexten

Leitfragen:
Was sind Datenbanken und wie kann man mit ihnen arbeiten? Wie entwickelt man selbst eine Datenbank für einen Anwendungskontext?

Vorhabenbezogene Konkretisierung:
Am Beispiel eines Online-Buchhandels wird der Aufbau einer Datenbank sowie wichtige Grundbegriffe erarbeitet. Die Schülerinnen und Schüler nehmen dabei zunächst die Sicht der Anwender an, die eine bestehende Datenbank beschreiben und analysieren und mithilfe von SQL-Abfragen Daten gezielt herausfiltern.

Nachdem die Lernenden in der ersten Sequenz mit Datenbanken vertraut gemacht wurden, nehmen sie nun die Rolle der Entwickler an, indem sie selbst Datenbanken von Grund auf modellieren und das Modell in ein Relationenschema überführen. Sie arbeiten mit Entity-Relationship-Diagrammen, um Entitäten, zugehörige Attribute, Relationen und Kardinalitäten in Anwendungskontexten darzustellen. Gegebene ER-Diagramme werden analysiert, erläutert und modifiziert.
Der bereits in der ersten Sequenz problematisierte Begriff der Redundanz wird am Ende des Unterrichtsvorhabens wieder aufgegriffen, um die Normalisierung von Datenbanken zu thematisieren. Bestehende Datenbankschemata werden hinsichtlich der 1. bis 3. Normalform untersucht und (soweit nötig) normalisiert.

Zeitbedarf: 20 Stunden

Sequenzierung des Unterrichtsvorhabens: siehe nächste Seite

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Aufgaben und Materialien
1. Nutzung von relationalen Datenbanken a) Aufbau von Datenbanksystemen und Grundbegriffe - Aufgaben und Eigenschaften eines Datenbanksystems - Erarbeitung der Begriffe Tabelle, Attribut, Attributwert, Datensatz, Datentyp, Primärschlüssel, Datenbankschema - Problematisierung von Redundanzen, Anomalien und Inkonsistenzen b) SQL-Abfragen - Erarbeitung der grundlegenden Sprachelemente von SQL (SELECT(DISTINCT), FROM, WHERE, JOIN) - Analyse und Erarbeitung von SQL-Abfragen (AND, OR, NOT, UNION, AS, GROUP BY,ORDER BY, ASC, DESC, COUNT, MAX, MIN, SUM, Arithmetische Operatoren: +, -, *, /, (…), Vergleichsoperatoren: =, <>, >, <, >=, <=, LIKE, BETWEEN, IN, IS NULL, geschachtelte Select-Ausdrücke) c) Vertiefung an einem weiteren Datenbankbeispiel	Die Schülerinnen und Schüler - erläutern die Eigenschaften, Funktionsweisen und 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), - überführen Datenbankschemata in die 1. bis 3. Normalform (M), - verwenden die Syntax und Semantik einer Datenbankabfragesprache, um Informationen aus einen Datenbanksystem zu extrahieren (I), - 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). - 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 Vorgaben, ethischer Aspekte und gesellschaftlicher Werte unter Berücksichtigung unterschiedlicher Interessenlagen (A).	Mögliche Literatur: - *Informatik 3*, Schoeningh - *Informatik für Schule und Ausbildung*, Pearson - *Duden Informatik Lehrbuch II*, Duden Paetec Schulbuchverlag Mögliche Software: - mySQL-Datenbank - HeidiSQL Datenbank-Ideen: - Fahrradhandlung - Schulverwaltung - Drogerielager
2. Modellierung von relationalen Datenbanken a) Datenbankentwurf durch ER-Diagramme - Ermittlung von Entitäten, zugehörigen Attributen, Beziehungen und Kardinalitäten in Anwendungssituationen und Modellierung eines Datenbankentwurfs in Form eines Entity-Relationship-Diagramms - Erläuterung und Erweiterung einer Datenbankmodellierung b) Entwicklung eines relationalen Modells aus einem Datenbankentwurf - Überführung eines Entity-Relationship-Diagramms in ein relationales Datenbankschema		

inklusive der Bestimmung von Primär- und Fremdschlüsseln c) Normalformen - Überprüfung von Datenbankschemata hinsichtlich der 1. bis 3. Normalform und Normalisierung (um Redundanzen zu vermeiden und Konsistenz zu gewährleisten)		
3. Wirkungen der Automatisierung: Datenschutzaspekte a) Sicherheit von Daten in Datenbanken b) Datenschutzbestimmungen in Deutschland c) Fallbeispiel: Facebook		Aktuelle Vorfälle: Datenklau bei großen Firmen (Sony, GMX, …). Wie sollte man (als Schüler) mit seinen Daten umgehen? Was ist erlaubt, was nicht? (Bilder hochladen, etc…) Wie wird das bei Facebook gehandhabt?
4. Übung und Vertiefung der Nutzung und Modellierung von relationalen Datenbanken		Prüfungsvorbereitung

Unterrichtsvorhaben Q1L-I

Thema: Wiederholung und Vertiefung der objektorientierten Modellierung

Leitfragen:
Wie wird aus einem anwendungsbezogenen Sachkontext ein informatisches Klassenmodell entwickelt? Wie werden Attribute, Methoden und Beziehungen identifiziert, den Klassen zugeordnet und dargestellt?

Vorhabenbezogene Konkretisierung:
Der bereits bekannte objektorientierte Zugang zu informatischer Modellierung wird von einer allgemeinen Betrachtung dieses informatischen Konzepts auf eine konkrete Problematik übertragen. Anhand dieser wird eine anwendungsbezogene Implementation Schritt für Schritt von der Objektidentifikation über das Entwurfs- und Implementationsdiagramm durchlaufen.
Grundlegende Modellierungskonzepte wie Sichtbarkeiten, Assoziationen, Vererbung sowie deren Darstellung in Entwurfs- und Klassendiagrammen und Dokumentationen werden wiederholt. Ebenso wird erneut die grafische Darstellung von Objektkommunikation thematisiert.
Anhand von Gütekriterien und Eigenschaften von Modellierung entwickeln und bewerten die Schülerinnen und Schüler Klassenentwürfe.
Das Konzept der objektorientierten Modellierung wird um die Idee der abstrakten Klasse und des Interfaces erweitert.

Zeitbedarf: 15 Stunden

Sequenzierung des Unterrichtsvorhabens: siehe nächste Seite

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Aufgaben und Materialien
1. Wiederholung der grundlegenden Konzepte der objektorientierten Programmierung a) Sichtweise der objektorientierten Informatik auf die Welt b) OOP als informatikspezifische Modellierung der Realität c) Schritte der Softwareentwicklung	Die Schülerinnen und Schüler … - analysieren und erläutern objektorientierte Modellierungen (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), - nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), - wenden eine didaktisch orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I), - stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D), - dokumentieren Klassen (D), - stellen die Kommunikation zwischen Objekten grafisch dar (D)	Objekte in der Realität vs. Objekte in Java: Eigenschaften und Fähigkeiten realer Objekte (Hund, Haus, Auto,...) in Java-Klassen übersetzen
2. Erweiterung der objektorientierten Programmierung a) Umsetzung einer Anforderung in Entwurfs- und Klassendiagramm b) Abstrakte Klassen und Interfaces b) Objektkommunikation im Sequenzdiagramm c) Klassendokumentation d) Umsetzung von Teilen der Modellierung e) Wiederholung/Vertiefung der Datenstruktur Array, Eigenschaften der Datenstruktur, Standardoperationen für ein- und zweidimensionale Arrays		Modellierung eines Autohauses:(abstrakte) Fahrzeug-Klasse mit Unterklassen; begrenzter Stellplatz für Fahrzeuge (→ Fahrzeug-Array) Angeschlossene Werkstatt kann u.a. Fahrzeuge reparieren (aber auch einzelne Bestandteile von Fahrzeugen) (→ Interface "Reparierbar") Weitere Objekte einbinden (bspw. Hersteller, Kunde → Interface "Kontaktierbar")
3. Übung und Vertiefung der OOM / OOP		Weitere Mini-Projekte modellieren und implementieren Prüfungsvorbereitung

Unterrichtsvorhaben Q1L-II

Thema:
Organisation und Verarbeitung von Daten I – Modellierung und Implementierung von Anwendungen mit dynamischen und linearen Datenstrukturen

Leitfragen:
Wie müssen Daten linear strukturiert werden, um in den gestellten Anwendungsszenarien eine beliebige Anzahl von Objekten verwalten zu können?

Vorhabensbezogene Konkretisierung:
Ausgehend von einigen Alltagsbeispielen werden als Erstes die Anforderungen an eine Datenstruktur erschlossen. Anschließend werden die Möglichkeiten des Arrays untersucht, lineare Daten zu verwalten und über deren Grenzen/Probleme die Vorteile einer dynamischen linearen Struktur am Beispiel der Struktur Queue erarbeitet (Anwendungskontext Warteschlange). Die Klasse wird selbst implementiert und später mit der vorgegebenen Klasse *Queue* verglichen. Zur Vertiefung der Kenntnisse wird ein weiteres Anwendungsszenario eingeführt, dessen Lösung modelliert und implementiert wird. Darauf folgt die Erarbeitung der Struktur Stack, die mithilfe eines einfachen Anwendungsszenarios eingeführt (bspw. Palindrome) wird. Hier wird die Klasse *Stack* selbst vorgegeben und die Operationen erläutert. Weitere Aufgaben dienen der Vertiefung und Sicherung.

Um die Unterschiede der beiden Prinzipien FIFO und LIFO zu verstehen, werden zur Lösung der Aufgaben sowohl der Stack als auch die Queue benötigt.

Als letzte lineare dynamische Datenstruktur wird die Liste eingeführt. In dieser Sequenz liegt der Fokus auf der Möglichkeit, auf jedes Element zugreifen zu können. Nachdem die umfangreicheren Standardoperationen dieser Datenstruktur in einem einführenden Beispiel (bspw. Vokabeltrainer) erarbeitet und in einem weiteren Beispiel vertieft wurden, werden abschließend in einem Anwendungskontext verschiedene lineare Datenstrukturen angewendet. Die Modellierung erfolgt beim gesamten Vorhaben in Entwurfs- und Implementationsdiagrammen.

Um motivierendere Aufgabenstellungen zu ermöglichen, werden graphische Benutzeroberflächen (mit Java Swing) eingeführt und inhaltsbezogene Anwendungsaufgaben mithilfe von einfachen Oberflächen gelöst.

Zeitbedarf: 30 Stunden

Sequenzierung des Unterrichtsvorhabens: siehe nächste Seite

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Aufgaben und Materialien
1. Die Datenstruktur **Schlange** a) Nachteile von Arrays; Modellierung und Implementierung der dynamischen Verknüpfung von Objekten b) Generische Typen, Trennung von Verwaltung und Inhalt dyn. DS. c) Erläuterung von Problemstellungen, die nach dem FIFO-Prinzip bearbeitet werden d) Funktionalität der Schlange unter Verwendung der Klasse *Queue*; Implementieren der Standardoperationen; Abgleich mit der Abiturklasse *Queue* e) Modellierung und Implementierung einer Anwendung auf der Basis einer Anforderungsbeschreibung mit Objekten der Klasse *Queue* f) Einfache graphische Benutzeroberflächen erstellen g) Das Model-View-Controller-Konzept in der Praxis	Die Schülerinnen und Schüler… - 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), - interpretieren Fehlermeldungen und korrigieren den Quellcode (I), - stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D) - erläutern Operationen dynamischer (linearer und nicht-linearer) Datenstrukturen (A), - implementieren Operationen dynamischer (linearer und nicht-linearer) Datenstrukturen (I), - modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), - ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M), - dokumentieren Klassen (D), - implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), - entwickeln mit didaktisch orientierten Entwicklungsumgebungen einfache Benutzeroberflächen zur Kommunikation mit einem Informatiksystem (M).	Wartezimmer-Projekt: Die Reihenfolge des Eintreffens von Patienten im Wartezimmer einer Arztpraxis soll verwaltet werden. (→ FIFO; Schlange) Für die Verwaltung soll eine einfache graphische Benutzeroberfläche erstellt werden (→ Java-Swing) Das Design-Pattern "Model-View-Controller" soll verwendet werden, um den Aufbau des Programms zu strukturieren. (siehe Informatik 2, Schoeningh)
3. Die Datenstruktur **Stapel** a) Erläuterung von Problemstellungen, die nach dem LIFO-Prinzip bearbeitet werden b) Funktionalität der Klasse Stapel unter Verwendung der Klasse *Stack* c) Modellierung und Implementierung einer Anwendung auf Basis einer Anforderungsbeschreibung mit Objekten der Klasse *Queue* d) Modellierung und Implementierung einer Anwendung unter Verwendung verschiedener Datenstrukturen (Objekte der Klassen *Queue*, *Stack* und *Array (Palindrom)*)		Klammern-Tester-Projekt: Mithilfe eines Stapels sollen Klammernterme (Bspw. "((())())" oder "[{()<>}]") auf Korrektheit überprüft werden. Auf Basis der GUI aus dem Wartezimmer-Projekt lässt sich dieses Projekt auch zur Übung für Benutzeroberflächen einsetzen. Weitere Projektidee: "Palindrome" Handelt es sich bei übergebenen Wörtern um Palindrome oder nicht.
4. Die Datenstruktur **Liste** a) Analyse der Möglichkeiten bisheriger Datenstrukturen zwecks Bestimmung notwendiger Funktionalitäten für komplexere Anwendungen (Abgrenzung zu *Stack*/*Queue*, zusätzliche		Projektideen: - Verwaltung einer Musik-Sammlung (CDs) - Vokabeltrainer (Vokabelkarten) - Ski-Abfahrtslauf (versetze

Fähigkeiten der Klasse *List*) b) Erarbeitung der Funktionalität der Liste unter Verwendung der Klasse *List* c) Modellierung und Implementierung einer Anwendung mit Objekten der Klasse *List* d) Modellierung und Implementierung einer Anwendung unter Verwendung verschiedener Datenstrukturen (Stack, Queue, List)		Starts; Läufer nach Laufzeit einsortieren)
5. Übungen und Vertiefungen zur Verwendung linearer und dynamischer Datenstrukturen anhand weiterer Problemstellungen		Projektidee: "Ein Array voller Listen – Eine Liste voller Arrays" Prüfungsvorbereitung

Unterrichtsvorhaben Q1L-III

Thema: Algorithmen zum Suchen und Sortieren auf linearen Datenstrukturen

Leitfragen:
Nach welchen Grundprinzipien können Algorithmen strukturiert werden? Welche Qualitätseigenschaften sollten Algorithmen erfüllen? Wie können mithilfe von Such- und Sortieralgorithmen Daten in linearen Strukturen effizient (wieder-)gefunden werden?

Vorhabenbezogene Konkretisierung:
Zunächst werden anhand eines Anwendungsbeispiels übergreifende Algorithmuseigenschaften (wie Korrektheit, Effizienz und Verständlichkeit) erarbeitet und Schritte der Algorithmusentwicklung wiederholt. Dabei kommen Struktogramme zur Darstellung von Algorithmen zum Einsatz.
Als besondere Struktur von Algorithmen wird die Rekursion an Beispielen veranschaulicht und gegenüber der Iteration abgegrenzt. Rekursive Algorithmen werden von den Schülerinnen und Schülern analysiert und selbst entwickelt.
In der zweiten Unterrichtssequenz geht es um die Frage, wie Daten in linearen Strukturen (lineare Liste und Array) (wieder-)gefunden werden können. Die lineare Suche als iteratives und die binäre Suche als rekursives Verfahren werden veranschaulicht und implementiert. Die Bewertung der Algorithmen erfolgt, indem jeweils die Anzahl der Vergleichsoperationen und der Speicherbedarf ermittelt wird. Das Hashing wird als Suchprinzip erforscht und angewendet.
Möchte man Daten effizient in einer linearen Struktur wiederfinden, so rückt zwangsläufig die Frage nach einer Sortierstrategie in den Fokus. Es wird mindestens ein iteratives und ein rekursives Sortierverfahren erarbeitet und implementiert sowie ihre Effizienz im O-Kalkül bewertet.

Zeitbedarf: 25 Stunden

Sequenzierung des Unterrichtsvorhabens: siehe nächste Seite

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Aufgaben und Materialien
1. Eigenschaften von Algorithmen a) Qualitätseigenschaften von Algorithmen b) Strukturierung von Algorithmen mit Hilfe der Strategien „Modularisierung“ und „Teile und Herrsche“; Darstellung durch Struktogramme c) Analyse und Entwicklung von rekursiven Algorithmen	Die Schülerinnen und Schüler - analysieren und erläutern Algorithmen und Programme (A), - modifizieren Algorithmen und Programme (I), - stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D), - entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“, „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). - implementieren und erläutern iterative und rekursive Such- und Sortierverfahren unterschiedlicher Komplexitätsklassen (Speicherbedarf und Laufzeitverhalten) (I), - beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A), - beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), - nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), - interpretieren Fehlermeldungen und korrigieren den Quellcode (I)	Projektideen: - ggT - Fibonacci-Zahlen - Türme von Hanoi
2. Suchen in Listen und Arrays a) Lineare Suche in Listen und Arrays b) Binäre Suche in einem Array c) Untersuchung der beiden Verfahren bzgl. Laufzeit und Speicherplatzbedarf d) Suchen mit Hashing-Verfahren		Lineare Suche in Listen und Arrays wurde schon in Q1L-II.4 bzw. Q1L-I.2 behandelt und muss hier nur kurz aufgefrischt werden.
3. Sortieren auf Listen und Arrays a) Entwicklung und Implementierung von zwei iterativen Sortierverfahren für eine Liste (bspw. Bubble- und Min-Sort) b) Entwicklung und Implementierung mindestens eines rekursiven Sortierverfahrens für ein Array (bspw. Merge- und/oder Quicksort) c) Laufzeiten im O-Kalkül d) Untersuchung der Verfahren bzgl. Laufzeit und Speicherplatzbedarf (Anzahl Vergleiche/Vertauschungen) im O-Kalkül		Zunächst Sortieren ganzer Zahlen. Möglicher Wiederaufgriff der CD-Verwaltung oder des Vokabeltrainers aus Q1L-II.4 (→ CDs bzw. Vokabeln nach bestimmten Kriterien sortieren lassen) Einfache Algorithmen (in Pseudocode) untersuchen und Laufzeit im O-Kalkül bestimmen

Unterrichtsvorhaben Q1L-IV

Thema: Automaten und formale Sprachen

Leitfragen:
Wie lassen sich reale Automaten durch ein Modell formal beschreiben? Wie kann die Art und Weise, wie ein Computer Zeichen (Eingaben) verarbeitet, durch Automaten dargestellt werden? Welche Eigenschaften besitzen Automaten und was können sie leisten? Wie werden sie dargestellt? Wie werden reguläre und kontextfreie Sprachen durch eine Grammatik beschrieben? In welchem Verhältnis stehen endliche (Keller-)Automaten und Grammatiken? Welche Anwendungsfälle können durch endliche Automaten und Grammatiken regulärer und kontextfreier Sprachen beschrieben werden und welche nicht? Wie kann eine eigene Programmiersprache eingelesen, interpretiert und ausgeführt werden?

Vorhabensbezogene Konkretisierung:
Ausgehend von der Beschreibung und Untersuchung realer Automaten wird das formale Modell eines endlichen Automaten entwickelt. Neben dem Mealy-Automaten geht es vor allem um den erkennenden endlichen Automaten. Auf die Erarbeitung der Beschreibung folgt die Modellierung eigener Automaten und die Untersuchung bestehender, um die Eigenschaften und Grenzen eines endlichen Automaten zu erkennen. Hierbei wird dessen Verhalten auf bestimmte Eingaben analysiert.
Als Erweiterung wird das Konzept der (nichtdeterministischen) Kellerautomaten kennengelernt. Kellerautomaten werden als Übergangsgraph mit einer graphischen Darstellung eines Stapels (wie in Q1L-II) dargestellt.
An den Themenkomplex *Endliche Automaten* schließt sich die Erarbeitung von Grammatiken regulärer und kontextfreier Sprachen an. Die Untersuchung beginnt bei die Erschließung der formalen Beschreibung und wird mit der Entwicklung von Grammatiken zu regulären Sprachen fortgeführt. Hierbei wird auch die Beziehung von Grammatiken regulärer Sprachen zu endlichen Automaten an Beispielen erarbeitet und analysiert. Hierzu gehört auch die Untersuchung, welche Problemstellungen durch endliche Automaten und reguläre Grammatiken beschrieben werden können und welche nicht.
Ebenso wird der Zusammenhang zwischen Kellerautomaten, kontextfreier Grammatiken und kontextfreier Sprachen erarbeitet und die Grenzen von Kellerautomaten untersucht.
Für eine einfache Sprache wird ein Scanner, ein Parser und ein Interpreter modelliert und implementiert.

Zeitbedarf: 30 Std.

Sequenzierung des Unterrichtsvorhabens: siehe nächste Seite

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Aufgaben und Materialien
1. Endliche Automaten a) Erarbeitung der formalen Beschreibung eines Mealy-Automaten und der Darstellungsformen b) Erarbeitung der formalen Beschreibung eines deterministischen endlichen Automaten (DEA) als 5-Tupel sowie dessen graphische Darstellungsform; Erschließung der Fachbegriffe Alphabet, Wort, (akzeptierte) Sprache (Darstellung durch reg. Ausdrücke und Mengenschreibweise), Determinismus c) Analyse der Eigenschaften von DEA durch die Modellierung eines Automaten zu einer gegebenen Problemstellung, der Modifikation eines Automaten sowie die Überführung der gegebenen Darstellungsform in eine andere d) Grenzen von DEA	Die Schülerinnen und Schüler - 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) - stellen endliche Automaten in Tabellen und Graphen dar und überführen sie in die jeweils andere Darstellungsform (D) - entwickeln zur Grammatik einer regulären oder kontextfreien Sprache einen zugehörigen endlichen Automaten oder einen Kellerautomaten (M) - analysieren und erläutern Grammatiken regulärer und kontextfreier Sprachen (A) - modifizieren Grammatiken regulärer und kontextfreier Sprachen (M) - entwickeln zu einer regulären oder kontextfreien Sprache eine Grammatik, die die Sprache erzeugt (M) - entwickeln zur akzeptierten Sprache eines Automaten eine zugehörige Grammatik (M) - beschreiben an Beispielen den Zusammenhang zwischen Automaten und Grammatiken (D) - erläutern die Grenzen endlicher Automaten und regulärer Grammatiken im Anwendungszusammenhang auf (A) - ermitteln die formale Sprache, die durch eine Grammatik erzeugt wird (A) - modellieren und implementieren Scanner, Parser und Interpreter zu einer gegebenen regulären Sprache	Modellierung eines Cola-Automaten DEA zu verschiedenen Sprachen erstellen. Bspw.: $L(A) = \{a^n b \mid n \in \mathbb{N}\}$ $L(A) = \{awb \mid w \in \{a,b\}^*\}$ $L(A) = \{w \mid w \in \{a,b\}^*$ $und\ w\ enthält$ $eine\ gerade\ Anzahl\ a's\}$ Sprachen vorgegebener DEA herausfinden $L(A) = \{a^n b^n \mid n \in \mathbb{N}\}$
2. Kellerautomaten a) Erarbeitung der Funktionsweise und der graphischen Darstellung von Kellerautomaten b) Analyse von Kellerautomaten und Angabe einer entsprechenden Sprache in Mengenschreibweise c) Kellerautomaten zu vorgegebenen Sprachen erstellen d) Grenzen der Kellerautomaten		$L(A) = \{a^n b^n \mid n \in \mathbb{N}\}$ $L(A) = \{w \mid w \in \{a,b\}^*$ $und\ w\ enthält\ gleich$ $viele\ a's\ wie\ b's\}$ Die Sprache, die alle Palindrome über $\{a,b\}$ enthält. $L(A) = \{a^n b^n c^n \mid n \in \mathbb{N}\}$
3. Grammatiken regulärer und kontextfreier Sprachen a) Erarbeitung der formalen Beschreibung einer linksregulären und einer kontextfreien Grammatik; Zusammenhang zu erzeugter Sprache b) Analyse der Eigenschaften einer linksregulären/kontextfreien Grammatik durch deren Entwicklung und Modellierung zu einer gegebenen Problemstellung.		Grammatiken (links-) regulärer/kontextfreier Sprachen (wie Beispiele s.o.)
4. Scanner, Parser und Interpreter a) Modellierung und Implementierung eines Scanners b) Entwicklung eines Parsers;		Projekt: Compilerbau → Material von Daniel Garmann, Gymnasium Odenthal, Link:

Verfahren des Parsens nachvollziehen und durchführen c) Parser implementieren d) Interpreter implementieren		http://projekte.gymnasium-odenthal.de/informatik/ →Informatik → Jahrgangsstufe Q → 003 … → 05 Compilerbau
5. Übungen und Vertiefungen Verwendung endlicher Automaten und Grammatiken regulärer Sprachen		Prüfungsvorbereitung

Unterrichtsvorhaben Q1L-V

Thema: Organisation und Verarbeitung von Daten II – Modellierung und Implementierung von Anwendungen mit dynamischen nicht-linearen Datenstrukturen (Bäume)

Leitfragen:
Wie können Daten mithilfe von Baumstrukturen verwaltet werden? Wie können mit binären Suchbäumen Inhalte sortiert verwaltet werden und welche Vor- und Nachteile bietet dies?

Vorhabenbezogene Konkretisierung:
Anhand des Anwendungskontextes Spielbäume werden zunächst der generelle Aufbau von Baumstrukturen (auch nicht-binäre) und wichtige Grundbegriffe erarbeitet. Die Darstellung von Bäumen mit Knoten und Kanten wird eingeführt.

Anschließend rückt der Fokus auf die binären Bäume, deren rekursiver Aufbau für die Traversierung der Datenstruktur genutzt wird. Die Preorder-Traversierung wird verwendet, um einen gespeicherten Inhalt in einem Binärbaum zu finden (Tiefensuche).
Der Anwendungskontext Ahnenbaum wird mithilfe der Klasse BinaryTree (der Materialien für das Zentralabitur in NRW) modelliert und (ggf. in Teilen) implementiert. Dabei wird u. a. die Erzeugung eines Binärbaums mithilfe der beiden Konstruktoren der Klasse BinaryTree thematisiert.

Möchte man Daten geordnet speichern, so bietet sich die Struktur des binären Suchbaums an. An Beispielen wird zunächst das Prinzip des binären Suchbaums erarbeitet. Die Operationen des Suchens, Einfügens, Löschens und der sortierten Ausgabe werden thematisiert und ihre Laufzeiten (auch bei entarteten Bäumen) untersucht.
Um Daten in einem Anwendungskontext mithilfe eines binären Suchbaums verwalten zu können, müssen sie in eine Ordnung gebracht werden können, d. h. sie müssen vergleichbar sein. Diese Vorgabe wird mithilfe des Interfaces Item realisiert, das alle Klassen, dessen Objekte in einem Suchbaum verwaltet werden sollen, implementieren müssen. Auf diese Weise wird ein Anwendungskontext (Benutzerverwaltung) mithilfe der Klassen BinarySearchTree und Item modelliert und implementiert.

Zeitbedarf: 25 Stunden

Sequenzierung des Unterrichtsvorhabens: siehe nächste Seite

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Aufgaben und Materialien
1. Aufbau von Baumstrukturen und Grundbegriffe a) Erarbeitung der Begriffe Wurzel, Knoten, Blatt, Kante, Grad eines Knotens und eines Baumes, Pfad, Tiefe, Ebene, Teilbaum b) Aufbau und Darstellung von Baumstrukturen in verschiedenen Anwendungskontexten	Die Schülerinnen und Schüler - stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D), - erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), - analysieren und erläutern Algorithmen und Programme (A), - stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D). - beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), - ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), - ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M), - modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M), - verwenden bei der Modellierung geeigneter Problemstellungen die Möglichkeiten der Polymorphie (M), - entwickeln iterative und rekursive Algorithmen unter Nutzung der Konstruktionsstrategien „Modularisierung“, „Teilen und Herrschen“ und "Backtracking" (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), - testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I),	Mögliche Zugang: Spielbäume für einfaches Zweispielerspiel (bspw. "Würfeldrehen" → Ein Würfel wird abwechselnd auf eine neue Seite gekippt und die Augenzahl zur eigenen addiert. Gewonnen hat, wer auf eine vorher festgelegte Punktzahl exakt erreicht.)
2. Binäre Bäume a) rekursiver Aufbau eines binären Baums b) Traversierungen (pre-, in-, postorder) c) Modellierung eines Binärbaums in einem Anwendungskontext mit Hilfe der Klasse BinaryTree (als Entwurfs- und Implementationsdiagramm) d) Implementation einer Anwendung der Datenstruktur binärer Baum (ggf. in Teilen)		Projektideen: - Ahnenbaum aufbauen und Verwandtschaftsbeziehungen auswerten - Codierungsbäume: bspw. Morsezeichen als innere Knoten codieren, in Blättern stehen Buchstaben
3. Binäre Suchbäume a) Prinzip des binären Suchbaums, Ordnungsrelation b) Operationen auf dem binären Suchbaum (Suchen, Einfügen, Löschen, sortierte Ausgabe) c) Modellierung eines binären Suchbaums in einem Anwendungskontext mit Hilfe der Klasse BinarySearchTree (als Entwurfs- und Implementationsdiagramm) und dem Interface Item d) Implementation einer Anwendung der Datenstruktur binärer Suchbaum (ggf. in Teilen) Optional: AVL-Bäume		Projektideen: - "Schülerverwaltung" (→ Sortierung bspw. nach Name) - CD-Verwaltung (Rückgriff auf Q1L-II.4) -… Laufzeit von entarteten Bäumen thematisieren. Rotationen anwenden und bessere Laufzeit thematisieren
4. Übung und Vertiefungen Verwendung von Binärbäumen oder binären Suchbäumen anhand weiterer Problemstellungen vertiefen		Prüfungsvorbereitung

Unterrichtsvorhaben Q2L-I:

Thema: Organisation und Verarbeitung von Daten II – Modellierung und Implementierung von Anwendungen mit dynamischen nicht-linearen Datenstrukturen (Graphen)

Leitfragen:
Wie kann man Objekte, die in mehrfacher Beziehung zueinander stehen, in einer Datenstruktur sinnvoll zusammenfassen? Warum lassen sich manche Strichzeichnungen ohne Abzusetzen zeichnen und manche nicht? Was ist ein minimaler Spannbaum, wofür ist er in der Praxis relevant und wie kann man einen solchen erstellen? Wie findet man den kürzesten Weg von einer Stadt zur anderen? Wie kommt man aus einem Labyrinth heraus oder findet dort einen versteckten Schatz? Welchen weiteren Praxisbezug haben Graphen und wofür kann man Algorithmen zu Graphen alles verwenden?

Vorhabenbezogene Konkretisierung:
Anhand des klassischen Beispiels des Königsberger Brückenproblems wird die Darstellung eines geographischen Problemszenarios mithilfe von Graphen eingeführt und ein Verfahren zum Bestimmen eines Eulerweges bzw. Eulerkreises erforscht und angewendet.
Weitere Eigenschaften von Graphen (gerichtet/ungerichtet, gewichtet/ungewichtet, zusammenhängend etc.) und mögliche formale Darstellungsformen (Adjazenzmatrix, Adjazenzliste) werden anhand von praxisnahen Aufgabenstellungen erarbeitet.
Algorithmen zum Bestimmen von minimalen Spannbäumen, zur Tiefen- und Breitensuche und zum Finden kürzester Wege werden von den Schülerinnen und Schülern erforscht, praktisch angewendet und mithilfe der vorgegebenen Graphen-Klassen zum Teil implementiert.

Zeitbedarf: 25 Stunden

Sequenzierung des Unterrichtsvorhabens: siehe nächste Seite

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Medien oder Materialien
1. Definition eines Graphen a) Geografisches Szenario als Graph darstellen (Begriffe Knoten und Kanten werden erarbeitet) b) Eulerweg/-kreis definieren, bedingende Eigenschaften erforschen und an Graphen anwenden c) Eigenschaften von Graphen ((un)gewichtet, (un)gerichtet, zusammenhängend) d) Formale Darstellung mit Adjazenzliste/-matrix und Umwandlung in beiden Richtungen	Die Schülerinnen und Schüler - stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D), - erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), - analysieren und erläutern Algorithmen und Programme (A), - stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D). - beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), - ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M),	Problemstellungen: Königsberger Brückenproblem und Figuren ohne Absetzen des Stifts zeichnen
2. Standardalgorithmen für Graphen a) Minimale Spannbäume bestimmen mit Prim/Kruskal - praktisch durchführen - einen der Algorithmen implementieren b) Tiefensuche und Breitensuche zum Durchlaufen bzw. zur Suche innerhalb eines Graphen - praktisch durchführen - einen der Algorithmen implementieren c) kürzeste Wege in Graphen (Algorithmus von Dijkstra) - praktisch durchführen - implementieren	- 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“, „Teilen und Herrschen“ und "Backtracking" (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), - testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I),	Mögliche Problemstellungen: - Brückennetz für die Galapagosinseln planen (→MST) - fehlerhafte Pumpstation in Kanalisation finden (→Tiefensuche) - Weg aus Labyrinth finden (→ Tiefensuche) - kürzesten Weg zum Schatz in einem Labyrinth planen (→ Breitensuche) - Navigationssystem unter die Lupe nehmen (→ Dijkstra) Möglichkeit zum Erforschen von Dijkstra: - Graphen mit Knete und Faden erstellen, durch Hochziehen kürzeste Verbindungen herausfinden und dokumentieren - Webquest "Dijkstra": https://www-m9.ma.tum.de/material/de/spp-dijkstra/
3. Übung und Vertiefung Anhand weiterer Beispiele den Umgang mit den Graphen-Klassen vertiefen		Prüfungsvorbereitung

Unterrichtsvorhaben Q2L-II

Thema: Aufbau von und Kommunikation in Netzwerken

Leitfragen:
Was macht menschliche Kommunikation aus? Welchen Stellenwert haben technische/ informatische Hilfsmittel für die Kommunikation? Wie werden Daten in einem Netzwerk zwischen den Kommunikationspartnern übertragen? Wie ist die Arbeitsteilung in Netzwerken gestaltet? Wie kann sicher in Netzwerken kommuniziert werden?

Vorhabenbezogene Konkretisierung:
Ausgehend von alltäglicher Face-to-Face-Kommunikation werden die Grundprinzipien sowie die Bewertungskriterien von Kommunikation erläutert. Das Netzwerk wird als vorteilhafte Kommunikationsstruktur dargestellt und anhand von Topologien kategorisiert. Ausgehend davon wird der Protokollbegriff entwickelt und anhand des TCP/IP-Schichtenmodells analysiert. Anschließend wird das Client-Server-Prinzip vorgestellt und angewandt.
Der Aufbau und die Administration von Netzwerken wird mithilfe der Software Filius erarbeitet und dabei IP-Adressbereiche, Kommunikationshardware und Client-/Server-Software kennengelernt, eingerichtet und getestet. Protokolle zur Kommunikation werden erforscht, analysiert und im Anwendungskontext selbst erstellt.
Auf Basis der vorgegebenen Klassen zur Kommunikation in Netzwerken (*Client, Server, Connection,…)* wird Client-Server-Software entwickelt und implementiert.
Sichere Kommunikation in Netzen ist nur dank kryptografischer Verfahren möglich. Stellvertretend werden zwei symmetrische und ein asymmetrisches Verfahren erläutert, angewandt und bewertet.

Zeitbedarf: 25 Stunden

Sequenzierung des Unterrichtsvorhabens: siehe nächste Seite

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Aufgaben und Materialien
1. Technische Kommunikation als Fortführung natürl. Kommun. a) Kommunikation im Shannon-Weaver-Modell b) Kriterien von technischen Kommunikationsarten c) Die Geschichte der technischen Kommunikation	Die Schülerinnen und Schüler - beschreiben und erläutern Topologien, die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken (A), - nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zum Erschließen, zur Aufbereitung und Präsentation fachlicher Inhalte (D), - analysieren und erläutern Protokolle zur Kommunikation in einem Client-Server-Netzwerk (A), - entwickeln und erweitern Protokolle zur Kommunikation in einem Client-Server-Netzwerk (M), - erläutern das Prinzip der Nebenläufigkeit (A), - analysieren und erläutern Algorithmen und Methoden zur Client-Server-Kommunikation (A), - entwickeln und implementieren Algorithmen und Methoden zur Client-Server-Kommunikation (I), - analysieren und erläutern Eigenschaften, Funktionsweisen und Einsatzbereiche symmetrischer und asymmetrischer Verschlüsselungsverfahren (A).	Projekt: Kommunikation im wilden Westen → Material von Daniel Garmann, Gymnasium Odenthal, Link: http://projekte.gymnasium-odenthal.de/informatik/ →Informatik → Jahrgangsstufe Q → 003 … → 13 Netzwerke → Demo
2. Aufbau von Netzwerken und Kommunikationsregeln a) Das Netzwerk als Organisationsprinzip der Kommunikation und Möglichkeiten der Ausformung b) Geregelte technische Kommunikation durch Protokolle in Schichtenmodellen		
3. Aufgabenteilung in Netzwerken durch Server und Client a) Aufbau und Aufgaben der Client-Server-Struktur b) Aufbau von Netzwerken mit Filius: Verkabelung, Einrichtung von IP-Adressen, Routern, Client-/Serversoftware c) Protokolle zw. Client und Server d) Protokolle für Netzwerkanwendungen analysieren und entwickeln		Software "Filius": Rechnernetzwerke aufbauen, einrichten, Software installieren und testen Datenkommunikation zu verschiedenen Protokollen nachvollziehen (in Filius) siehe alte Zentralabituraufgaben: "Mensaplan", "Filmausleihe",…
4. Entwicklung von Netzwerkanwendungen a) Konzept der Nebenläufigkeit (Threads) in Java b) Entwicklung und Implementation von Client- und Server-Software entsprechend vorgegebeneroder selbsterstellter Protokolle		Projektideen: - Uhrzeit-Server - Additions-Server - Mensaplan - Filmausleihe fertige Oberfläche zum Anpassen: http://projekte.gymnasium-odenthal.de/informatik/
5. Kryptologie a) Veranschaulichen und Anwenden von symmetrischen und asymmetrischen kryptographischen Verfahren (Caesar, Vigenère, RSA) b) Bewertung der Verfahren hinsichtlich ihrer Sicherheit und ihrem Aufwand		Bspw. mithilfe von Webquests: http://www.matheprisma.uni-wuppertal.de/Module/Caesar/ und http://www.matheprisma.uni-wuppertal.de/Module/RSA/
6. Übung und Vertiefung des Aufbaus von und der Kommunikation in Netzwerken		Prüfungsvorbereitung

Unterrichtsvorhaben Q2L-III

Thema: Nutzung und Modellierung von relationalen Datenbanken in Anwendungskontexten

Leitfragen:
Was sind Datenbanken und wie kann man mit ihnen arbeiten? Wie entwickelt man selbst eine Datenbank für einen Anwendungskontext?

Vorhabenbezogene Konkretisierung:
Am Beispiel eines Online-Buchhandels wird der Aufbau einer Datenbank sowie wichtige Grundbegriffe erarbeitet. Die Schülerinnen und Schüler nehmen dabei zunächst die Sicht der Anwender an, die eine bestehende Datenbank beschreiben und analysieren und mithilfe von SQL-Abfragen Daten gezielt herausfiltern.

Nachdem die Lernenden in der ersten Sequenz mit Datenbanken vertraut gemacht wurden, nehmen sie nun die Rolle der Entwickler an, indem sie selbst Datenbanken von Grund auf modellieren, das Modell in ein Relationenschema überführen und als Datenbank implementieren. Sie arbeiten mit Entity-Relationship-Diagrammen, um Entitäten, zugehörige Attribute, Relationen und Kardinalitäten in Anwendungskontexten darzustellen. Gegebene ER-Diagramme werden analysiert, erläutert und modifiziert.
Der bereits in der ersten Sequenz problematisierte Begriff der Redundanz wird am Ende des Unterrichtsvorhabens wieder aufgegriffen, um die Normalisierung von Datenbanken zu thematisieren. Bestehende Datenbankschemata werden hinsichtlich der 1. bis 3. Normalform untersucht und (soweit nötig) normalisiert.

Zeitbedarf: 25 Stunden

Sequenzierung des Unterrichtsvorhabens: siehe nächste Seite

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Aufgaben und Materialien
1. Nutzung von relationalen Datenbanken a) Aufbau von Datenbanksystemen und Grundbegriffe - Aufgaben und Eigenschaften eines Datenbanksystems - Erarbeitung der Begriffe Tabelle, Attribut, Attributwert, Datensatz, Datentyp, Primärschlüssel, Datenbankschema - Problematisierung von Redundanzen, Anomalien und Inkonsistenzen b) SQL-Abfragen - Erarbeitung der grundlegenden Sprachelemente von SQL (SELECT(DISTINCT), FROM, WHERE, JOIN) - Analyse und Erarbeitung von SQL-Abfragen (AND, OR, NOT, UNION, AS, GROUP BY,ORDER BY, ASC, DESC, COUNT, MAX, MIN, SUM, Arithmetische Operatoren: +, -, *, /, (…), Vergleichsoperatoren: =, <>, >, <, >=, <=, LIKE, BETWEEN, IN, IS NULL, geschachtelte Select-Ausdrücke) c) Vertiefung an einem weiteren Datenbankbeispiel	Die Schülerinnen und Schüler - erläutern die Eigenschaften und 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), - implementieren ein relationales Datenbankschema als Datenbank (I), - 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), - überführen Datenbankschemata in die 1. bis 3. Normalform (M), - verwenden die Syntax und Semantik einer Datenbankabfragesprache, um Informationen aus einen Datenbanksystem zu extrahieren (I), - 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). - 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 Vorgaben, ethischer Aspekte und gesellschaftlicher Werte unter Berücksichtigung unterschiedlicher Interessenlagen (A).	Mögliche Literatur: - *Informatik 3*, Schoeningh - *Informatik für Schule und Ausbildung*, Pearson - *Duden Informatik Lehrbuch II*, Duden Paetec Schulbuchverlag Mögliche Software: - mySQL-Datenbank - HeidiSQL Datenbank-Ideen: - Fahrradhandlung - Schulverwaltung - Drogerielager
2. Modellierung von relationalen Datenbanken a) Datenbankentwurf durch ER-Diagramme - Ermittlung von Entitäten, zugehörigen Attributen, Beziehungen und Kardinalitäten in Anwendungssituationen und Modellierung eines Datenbankentwurfs in Form eines Entity-Relationship-Diagramms - Erläuterung und Erweiterung einer Datenbankmodellierung b) Entwicklung eines relationalen Modells aus einem Datenbankentwurf - Überführung eines Entity-Relationship-Diagramms in ein relationales Datenbankschema		

inklusive der Bestimmung von Primär- und Fremdschlüsseln - implementieren das Datenbankschema als Datenbank c) Normalformen - Überprüfung von Datenbank-schemata hinsichtlich der 1. bis 3. Normalform und Normali-sierung (um Redundanzen zu vermeiden und Konsistenz zu gewährleisten)		
3. Wirkungen der Automatisierung: Datenschutzaspekte a) Sicherheit von Daten in Datenbanken b) Datenschutzbestimmungen in Deutschland c) Fallbeispiel: Facebook		Aktuelle Vorfälle: Datenklau bei großen Firmen (Sony, GMX, …). Wie sollte man (als Schüler) mit seinen Daten umgehen? Was ist erlaubt, was nicht? (Bilder hochladen, etc…) Wie wird das bei Facebook gehandhabt?
4. Übung und Vertiefung der Nutzung und Modellierung von relationalen Datenbanken		Prüfungsvorbereitung

Unterrichtsvorhaben Q2L-IV:

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. Mithilfe theoretischer Überlegungen zur Mächtigkeit von Methoden wird bewiesen, dass es unmöglich ist, ein Informatiksystem zu entwickeln, dass für jedes beliebige Computerprogramm und jede beliebige Eingabe entscheidet ob das Programm mit der Eingabe terminiert oder nicht (Halteproblem). Anschließend werden Vor- und Nachteile der Grenzen der Automatisierbarkeit angesprochen und der Einsatz von Informatiksystemen hinsichtlich prinzipieller Möglichkeiten und prinzipieller Grenzen beurteilt.

Zeitbedarf: 15 Stunden

Sequenzierung des Unterrichtsvorhabens: siehe nächste Seite

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Medien oder Materialien
1. Von-Neumann-Architektur und die Ausführung maschinennaher Programme a) prinzipieller Aufbau einer von Neumann-Architektur mit CPU, Rechenwerk, Steuerwerk, Register und Hauptspeicher b) Analyse und Erläuterung der Funktionsweise eines einfachen maschinennahen Programms c) Schreiben eigener einfacher Programme	Die Schülerinnen und Schüler - erläutern die Ausführung eines einfachen maschinennahen Programms sowie die Datenspeicherung auf einer „Von-Neumann-Architektur“ (A), - untersuchen und beurteilen Grenzen des Problemlösens mit Informatiksystemen (A).	Bspw. "Know-How-Rechner" zur Simulation maschinennahen Programmierens Bspw.: - Addieren - Multiplizieren - Sortieren
2. Grenzen der Automatisierbarkeit a) Vorstellung des Halteproblems b) Unlösbarkeit des Halteproblems c) Beurteilung des Einsatzes von Informatiksystemen hinsichtlich prinzipieller Möglichkeiten und prinzipieller Grenzen		Beispiel: Halteproblem inkl. Beweis der Unlösbarkeit Praktische Berechenbarkeit für wichtige Problemstellungen (NP-schwere Probleme): - Travelling-Salesman - Rucksackproblem

Zusätzliches individuelles Unterrichtsvorhaben: Wiederholung der Themen der Qualifikationsphase im Rahmen eines umfangreicheren Softwareprojekts (Bspw. Spielesammlung mit Benutzeroberfläche)