Schulinterner Lehrplan
des Aldegrever-Gymnasiums
für das Fach Informatik
in der Sekundarstufe II

Stand: August 2020

Inhalt

1 Die Fachgruppe Informatik 1

2 Entscheidungen zum Unterricht 1

2.1 Hinweise und Entscheidungen 1
2.2 Unterrichtsvorhaben 2
2.1.1 Übersichtsraster Unterrichtsvorhaben 2
2.1.2 Konkretisierte Unterrichtsvorhaben 3
2.3 Grundsätze der Leitstungsbewertung und Leistungsrückmeldung 14
2.3.1 Beurteilungsbereich Klausuren 50
2.3.2 Beurteilungsbereich Sonstige Mitarbeit 50

3. Entscheidungen zu fach- und unterrichtsübergreifenden Fragestellungen . 52

4. Qualitätssicherung und Evaluation 52

1 Die Fachgruppe Informatik

Das Städt. Aldegrever-Gymnasium liegt in der Innenstadt von Soest. An ihm werden zurzeit ca. 700 Schülerinnen und Schüler von ca. 60 Kolleginnen und Kollegen unterrichtet.

Das Fach Informatik wird am Aldegrever-Gymnasium im Rahmen des Kurses „Angewandte Informatik“ bereits im Wahlpflichtbereich II angeboten und zweistündig unterrichtet. In der Sekundarstufe II kooperiert die Schule mit den anderen beiden Soester Gymnasien sowie der Gesamtschule, sodass für alle Soester Schülerinnen und Schüler durchgängig Grundkurse angeboten werden. Ein Leistungskurs kam in den letzten Jahren aufgrund der geringen Nachfrage nicht zustande, wird aber grundsätzlich angeboten.

Die Fachschaft Informatik besteht zurzeit aus vier Lehrkräften. Den Fachkonferenzvorsitz hat Frau Rühl, ihre Stellvertreterin ist Frau Streitberg. Weiter Mitglieder der Fachkonferenz sind Herr Düllmann und Herr Reimann, letztgenannter unterrichtet Informatik nur in der Sekundarstufe I.

2 Entscheidungen zum Unterricht

2.1 Hinweise und Entscheidungen

- Die Fachschaft befürwortet bei den unten beschriebenen Unterrichtsvorhaben den Einsatz des Lehrwerks *Informatik* des Schöningh Verlags.
- Als Werkzeuge für die Einführungsphase werden *Greenfoot* oder *GLOOP* empfohlen, für die Qualifikationsphase *BlueJ*.
- Eine Unterrichtsstunde ist 45 min lang. Die im folgenden Unterrichtsraster angegebenen Zeitangaben sind Richtwerte.
- Die Kompetenzbereiche des Kernlehrplans werden wie folgt abgekürzt:
    - Argumentieren (A)
    - Modellieren (M)
    - Implementieren (I)
    - Darstellen und Interpretieren (D)
    - Kommunizieren (K)

2.2 Unterrichtsvorhaben

2.1.1 Übersichtsraster Unterrichtsvorhaben

Einführungsphase	
<u>Unterrichtsvorhaben E-I</u> **Thema:** *Einführung in die Informatik* **Zentrale Kompetenzen:** • Argumentieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Informatiksysteme • Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** • Einzelrechner • Dateisystem • Internet • Digitalisierung • Einsatz von Informatiksystemen • Wirkungen der Automatisierung • Geschichte der automatischen Datenverarbeitung **Zeitbedarf**: ca. 8 Stunden (+optionale Exkursion)	<u>Unterrichtsvorhaben E-II</u> **Thema**: *Einführung in die OOP/Informatisches Problemlösen und Algorithmen als Beschreibung von Abläufen* **Zentrale Kompetenzen:** • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren • Argumentieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Analyse, Entwurf und Implementierung einfacher Algorithmen • Syntax und Semantik einer Programmiersprache **Zeitbedarf**: ca. 30 Stunden
<u>Unterrichtsvorhaben E-III</u> **Thema**: *Suchen und Sortieren* **Zentrale Kompetenzen:** • Argumentieren • Darstellen und Interpretieren • Modellieren • Implementieren **Inhaltsfelder:** • Algorithmen • Formale Sprachen und Automaten	<u>Unterrichtsvorhaben E-IV</u> **Thema**: *Softwareentwicklung* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen

• Daten und ihre Strukturierung **Inhaltliche Schwerpunkte:** • Algorithmen zum Suchen und Sortieren • Syntax und Semantik einer Programmiersprache **Zeitbedarf**: ca. 12 Stunden	• Formale Sprachen und Automaten **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Syntax und Semantik einer Programmiersprache • Analyse, Entwurf und Implementierung einfacher Algorithmen **Zeitbedarf**: ca. 20 Stunden
Summe Einführungsphase: 70	

Qualifikationsphase – Q1 Grundkurs	
Unterrichtsvorhaben Q1-I **Thema**: Wiederholung und Vertiefung der objektorientierten Modellierung **Zentrale Kompetenzen**: - Modellieren - Darstellen und Interpretieren - Implementieren - Kommunizieren und Kooperieren **Inhaltsfelder**: - Daten und ihre Strukturierung - Algorithmen - Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte**: - Objekte und Klassen - Wirkung der Automatisierung **Zeitbedarf**: ca. 14 Stunden	***Unterrichtsvorhaben Q1-II*** **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**: ca. 20 Stunden
Unterrichtsvorhaben Q1-III **Thema**: Algorithmen zum Suchen und Sortieren auf linearen Datenstrukturen **Zentrale Kompetenzen:** - Argumentieren - Darstellen und Interpretieren - Modellieren - Implementieren **Inhaltsfelder:** - Algorithmen - Formale Sprachen und Automaten	***Unterrichtsvorhaben Q1-IV*** **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:**

Inhaltliche Schwerpunkte: - Analyse, Entwurf und Implementierung von Algorithmen - Algorithmen in ausgewählten informatischen Kontexten - Syntax und Semantik einer Programmiersprache **Zeitbedarf**: ca. 15 Stunden	- 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**: ca. 20 Stunden
Summe Qualifikationsphase 1: ca. 69 Stunden	

Qualifikationsphase – Q2 Grundkurs	
<u>Unterrichtsvorhaben Q2-I</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**: ca. 12 Stunden	***<u>Unterrichtsvorhaben Q2-I</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**: ca. 20 Stunden
<u>Unterrichtsvorhaben Q2-III</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	***<u>Unterrichtsvorhaben Q2-IV (optional)</u>*** ***Thema:*** *Wiederholung und Vertiefung zur konkreten Vorbereitung auf die Abiturprüfungen* ***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**: ca. 20 Stunden	*- Informatiksysteme* *- Formale Sprachen und Automaten* *- Informatik, Mensch und Gesellschaft* *Zeitbedarf: nach Bedarf*
Summe Qualifikationsphase 2: ca. 52 Stunden	

Qualifikationsphase – Q1 Leistungskurs	
<u>Unterrichtsvorhaben Q1-I</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**: ca. 20 Stunden	***<u>Unterrichtsvorhaben Q1-II</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**: ca. 30 Stunden
<u>Unterrichtsvorhaben Q1-III</u> **Thema**: Algorithmen zum Suchen und Sortieren auf linearen Datenstrukturen **Zentrale Kompetenzen:** - Argumentieren - Darstellen und Interpretieren - Modellieren - Implementieren **Inhaltsfelder:** - Algorithmen - Formale Sprachen und Automaten - Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:**	***<u>Unterrichtsvorhaben Q1-IV</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

- Analyse, Entwurf und Implementierung von Algorithmen - Algorithmen in ausgewählten informatischen Kontexten - Syntax und Semantik einer Programmiersprache - Iteration und Rekursion - Wirkungen der Automatisierung **Zeitbedarf**: ca. 30 Stunden	**Inhaltliche Schwerpunkte:** - Objekte und Klassen - Analyse, Entwurf und Implementierung von Algorithmen - Algorithmen in ausgewählten informatischen Kontexten - Syntax und Semantik einer Programmiersprache **Zeitbedarf**: ca. 40 Stunden
Summe Qualifikationsphase 1: ca. 120 Stunden	

Qualifikationsphase – Q2 Leistungskurs	
Unterrichtsvorhaben Q2-I **Thema**: Aufbau von und Kommunikation in Netzwerken **Zentrale Kompetenzen:** - Argumentieren - Darstellen und Interpretieren - Modellieren - Implementieren - Kommunizieren und Kooperieren **Inhaltsfelder:** - Informatiksysteme - Algorithmen - Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** - Einzelrechner und Rechnernetzwerke - Sicherheit - Nutzung von Informatiksystemen, Wirkungen der Automatisierung - Analyse, Entwurf und Implementierung von Algorithmen - Algorithmen in ausgewählten informatischen Kontexten **Zeitbedarf**: ca. 25 Stunden	***Unterrichtsvorhaben Q2-II*** **Thema**: Automaten und formale Sprachen **Zentrale Kompetenzen:** - Argumentieren - Darstellen und Interpretieren - Modellieren - Implementieren - Kommunizieren und Kooperieren **Inhaltsfelder:** - Formale Sprachen und Automaten - Informatiksysteme - Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** - Syntax und Semantik einer Programmiersprache - 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 - Grenzen der Automatisierung **Zeitbedarf**: ca. 30 Stunden
Unterrichtsvorhaben Q2-III **Thema**: Nutzung und Modellierung von relationalen Datenbanken in Anwendungskontexten **Zentrale Kompetenzen:** - Argumentieren - Modellieren - Implementieren - Darstellen und Interpretieren - Kommunizieren und Kooperieren **Inhaltsfelder:**	***Unterrichtsvorhaben Q2-IV*** **Thema:** Wiederholung und Vertiefung zur konkreten Vorbereitung auf die Abiturprüfungen **Zentrale Kompetenzen:** - Argumentieren - Modellieren - Implementieren - Darstellen und Interpretieren - Kommunizieren und Kooperieren **Inhaltsfelder:** - Daten und ihre Strukturierung

- 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**: ca. 25 Stunden	- Algorithmen - Informatiksysteme - Formale Sprachen und Automaten - Informatik, Mensch und Gesellschaft **Zeitbedarf: ca.** 10 Stunden
Summe Qualifikationsphase 2: ca. 90 Stunden	

2.1.2 Konkretisierte Unterrichtsvorhaben

Unterrichtsvorhaben EF-I: Einführung in die Informatik

Leitfragen: Womit beschäftigt sich das Fach Informatik in der Sek. II? Was sind die Fragestellungen und Meilensteine der Informatik?

Vorhabenbezogene Konkretisierung: Das erste Unterrichtsvorhaben stellt eine allgemeine Einführung in das Fach Informatik dar. Dabei sollen die Meilensteine in der Geschichte der Informatik und grundlegende Prinzipien wie das EVA-Prinzip und die Von-Neumann-Architektur erarbeitet werden.

Mit Hilfe eines geschichtlichen Abrisses und populärer informatischer Fallbeispiele sollen die Wechselbeziehung zwischen der Informatik und der Gesellschaft thematisiert und die Auswirkungen des Einsatzes von Informatiksystemen auf die Gesellschaft deutlich werden.

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

Das gesamte Vorhaben kann durch eine eintägige Exkursion in das Heinz-Nixdorf-Forum nach Paderborn ergänzt werden.

Zeitbedarf: ca. 8 Stunden (+ optionale Exkursion)

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Mögliche Beispiele, Medien oder Materialien
1. Geschichte und „Geschichtchen“ der Informatik und ihre Auswirkungen auf die heutige Gesellschaft	Die Schülerinnen und Schüler (SuS)… • …erläutern wesentliche Grundlagen der Geschichte der digitalen Datenverarbeitung (A) • …bewerten anhand von Fallbeispielen die Auswirkungen	• Fallbeispiele (berühmte Softwarefehler: z.B. Scud-

	des Einsatzes von Informatiksystemen (A)	Rakete im Irak-Krieg, Softwarefehler bei EC-Karten, Apollo-Mondlandung, Ariane 5)
2. Aufbau informatischer Systeme • Identifikation typischer Arbeitsweisen und Komponenten informatischer Systeme. • Herleitung der VN-Architektur • Identifikation des EVA-Prinzips als Prinzip der Verarbeitung von Daten und Grundlage der VN-Architektur.	Die SuS… • …beschreiben und erläutern den strukturellen Aufbau und die Arbeitsweise singulärer Rechner am Beispiel der VN-Architektur (A) • …nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation (K)	• Aus den vorherigen Beispielen und Diskussionen werden das EVA-Prinzip die in der VN-Architektur vorkommenden Komponenten herausgearbeitet. • Hier ist die Vergabe von Referaten oder eine arbeitsteilige Erarbeitung möglich.
3. Der Binärcode als Grundlage der Datenverarbeitung	Die SuS… • …stellen ganze Zahlen und Zeichen in Binärcodes dar (D) • …interpretieren Binärcodes als Zahlen und Zeichen (D)	• z.B. „Abenteuer Informatik“, Kapitel 5/10
4. Datenschutz	Die SuS… • … nutzen im die im Unterricht eingesetzten Informatiksysteme selbstständig, sicher , zielführend und Verantwortungsbewusst (D)	• z.B. Facebook • www.klicksafe.de

<u>Unterrichtsvorhaben EF-II: Einführung in die OOP/Informatisches Problemlösen und Algorithmen als Beschreibung von Abläufen</u>

Leitfragen: Was sind Klassen, Objekte, Methoden, Attribute? Wie interagieren Objekte unterschiedlicher und gleicher Klassen miteinander? Wie beschreibt man in der Informatik Abläufe computernah? Was zeichnet Algorithmen aus und wie kann man sie vergleichen? Wie werden Algorithmen entwickelt, dargestellt und in der Programmiersprache Java umgesetzt?

Vorhabenbezogene Konkretisierung: Dieses Unterrichtsvorhaben beinhaltet eine Einführung in die Objektorientierte Modellierung (OOM). In Kurzprojekten werden Objekte, Klassen und ihre Beziehungen in Form von Objekt-, Entwurfs- und Implementationsklassendiagrammen dargestellt. In einem weiteren Schritt sollen Objekte miteinander interagieren können. Hierbei wird der Unterschied zwischen beiden Beziehungsarten (gerichtete Assoziation, Ist-Beziehung) erarbeitet und visualisiert.

Anhand von alltäglichen Beispielen werden der Algorithmus-Begriff und seine Eigenschaften herausgearbeitet. Dabei werden Algorithmen erstellt, implementiert und beispielsweise in Programmab-

laufplänen (PAP) dokumentiert. Bedeutsam sind dabei die Herausarbeitung der grundlegenden algorithmischen Strukturen (Sequenz, bedingte Verzweigung und kopfgesteuerte Schleife) und die Überprüfung der Eigenschaften von Algorithmen.

Begonnen wird mit einem einfachen Szenario. Nach der sukzessiven Einführung der zu behandelnden Kontrollstrukturen Verzweigung, Schleifen werden zunehmend komplexere Problemstellungen von den SuS erarbeitet und gelöst, wobei die Modellierung als Vorstufe der Implementation eingefordert wird.

Im Anschluss werden anhand von konkreten Algorithmen die erarbeiteten Techniken geübt, Fragestellung zu Grenzen von Algorithmen thematisiert und die Idee der rekursiv formulierten Algorithmen erarbeitet.

Zeitbedarf: ca. 30 Stunden

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Mögliche Beispiele, Medien oder Materialien**
a) Grundlagen der OOP • Erarbeitung der Bedeutung von Klassen, Objekten und Methoden • Darstellung von Klassen und Objekten in UML-Notation • Benutzung von Objekten	Die SuS… • …ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M). • …stellen den Zustand eines Objekts dar (D).	• Greenfoot/GLOOP • PAP-Designer, yEd, OpenOffice
b) Algorithmik • Alltagsbeispiele zu Algorithmen • Erlernen grundlegender Algorithmen mit Kontrollstrukturen • Erarbeitung des Unterschieds zwischen Aufträgen und Anfragen anhand von Anwendungsbeispielen	Die SuS… • …kennen den Begriff „Algorithmus“ und seine Eigenschaften • …analysieren und erläutern einfache Algorithmen und Programme (A). • …modifizieren einfache Algorithmen und Programme (I). • …entwerfen einfache Algorithmen und stellen sie umgangssprachlich und grafisch dar (M). • …testen Programme schrittweise anhand von Beispielen (I) • …dokumentieren Klassen	• Ausgehend von den berühmten Softwarefehlern und mit Hilfe alltäglicher Beispiele wird der Algorithmus-Begriff entwickelt und an Beispielen überprüft, geschärft und eingeübt. • Mögliche Beispiele: Papierflieger (Sequenz), Kochrezept (Sequenz und bedingte Verzweigung) … • Material: PAP-

c) Objektbeziehungen modellieren und umsetzen • Analyse von Problemstellungen, bei denen die Interaktion von Objekten notwendig ist • Darstellung der Beziehungen in Form von Klassendiagrammen • Erarbeitung, Programmierung und Bewertung verschiedener Umsetzungsmöglichkeiten der Beziehungen d) Einführung des Variablenkonzepts/Datentypen • Problemorientierte Erarbeitung der Benutzung von Variablen als universellem Speicher. • Benutzung der Datentypen Integer, String, Char, Boolean. e) Vertiefung der Inhalte anhand eines weiteren Anwendungsbeispiels	durch Beschreibung der Funktionalität der Methoden (D). Die SuS… • …analysieren und erläutern eine objektorientierte Modellierung (A) • …modellieren Klassen mit ihren Attributen, ihren Methoden und Assoziationsbeziehungen (M). • …modellieren Klassen unter Verwendung von Vererbung (M). • …stellen die Kommunikation zwischen Objekten grafisch dar (M). • …stellen Klassen, Assoziations- und Vererbungsbeziehungen in Diagrammen grafisch dar (D). • …implementieren Algorithmen unter Verwendung von Variablen und Wertzuweisungen, Kontrollstrukturen sowie Methodenaufrufen (I). Die SuS… • …testen Programme schrittweise anhand von Beispielen (I). • …nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung und gemeinsamen Verwendung von Daten unter Berücksichtigung der Rechteverwaltung (K).	Designer • Euklidischer Algorithmus: Darstellung eines iterativen mathematischen Algorithmus mit Hilfe eines PAPs (Bedingte Verzweigung, kopfgesteuerte Schleife) und ugs. Beschreibung • Labyrinth-Problem: Die Rechte-Hand-Regel Darstellung in Umgangssprache und im PAP / Testen von Algorithmen / Übertragung auf vergleichbare Labyrinthe / Grenze des Algorithmus (Problemklasse) • Türme von Hanoi: Herausarbeiten von Regeln und Fragestellung aus der Geschichte, Reduzierung, praktisches Ausprobieren, Darstellung als Algorithmus, Zeitproblematik (Grenze der Umsetzung des Algo-

		rithmus) Rekursive Darstellung; Eigenschaften von Rekursionen (Selbstaufruf mit Laufbedingung bei Verkleinerung des Problems)

<u>Unterrichtsvorhaben EF-III: Suchen und Sortieren</u>

Leitfragen: Wie funktioniert die effektive Suche nach Objekten? Wie funktioniert das Sortieren von Objekten? Welche Algorithmen gibt es und wie effizient sind diese?

Vorhabenbezogene Konkretisierung: Anhand von alltäglichen Beispielen wird das Suchen nach Objekten motiviert. Dabei wird zunächst die iterative Suche betrachtet. Es soll deutlich werden, dass ein Objekt wesentlich schneller gefunden werden kann, wenn eine Vorsortierung der zu durchsuchenden Objekte vorliegt. Daraus soll der Algorithmus der binären Suche entwickelt werden. Die benötigte Vorsortierung motiviert die Betrachtung von Sortieralgorithmen. Als einfache Algorithmen wird hier beispielhaft der Selection- oder Bubblesort betrachtet. Dazu gehört die Modellierung, Implementierung (in einem vorgefertigtem Szenario) und Untersuchung auf Effizienz, die sich auf die Anzahl der Vergleiche und Vertauschungen beschränkt. Die Algorithmen in diesem Unterrichtsvorhaben werden in Programmablaufplänen modelliert.

Zeitbedarf: ca. 12 Stunden

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien oder Materialien**
a) Suchalgorithmen (iterative und binäre Suche) b) Einfache Sortieralgorithmen (Selection- / Bubblesort) c) (eindimensionales) Array, for-Schleife	Die Schülerinnen und Schüler • analysieren Such- und Sortieralgorithmen und wenden sie auf Beispiele an (D) • entwerfen einen weiteren Algorithmus zum Sortieren (M) • beurteilen die Effizienz von Algorithmen am Beispiel von Sortierverfahren hinsichtlich Zeitaufwand und Speicherplatzbedarf. (A) Die Schülerinnen und Schüler • analysieren und erläutern Algorithmen und Programme (A), • analysieren und erläutern	• Suchen in Alltagsbeispielen (z.B.: CD-Regal, Suche der größten/kleinsten Zahl) • Betrachtung der Effizienz der iterativen Suche • Optimierung der iterativen Suche zur binären Suche. (Die Vorsortierung als Bedingung) • Beschreibung der Algorithmen in PAPs • Umsetzung des Bubblesorts oder Selectionsorts • Zählen von Zugriffen / Vergleichen / (Dreiecks-tausch)

	einfache Algorithmen und Programme (A), • entwerfen einfache Algorithmen und stellen sie umgangssprachlich und grafisch dar (M), • implementieren Algorithmen unter Verwendung von Variablen und Wertzuweisungen, Kontrollstrukturen sowie Methodenaufrufen (I) • Stellen lineare und nicht lineare Strukturen grafisch dar und erläutern ihren Aufbau (D)	

<u>Unterrichtsvorhaben EF-IV: Softwareentwicklung</u>

Thema: Softwareentwicklung

Leitfragen: Wie wird Software entwickelt? Welche Phasen durchläuft Software während der Entwicklung? Welche Bereiche werden strikt voneinander getrennt (MVC-Konzept)?

Vorhabenbezogene Konkretisierung: Schülerinnen uns Schüler durchlaufen in diesem UV einen vollständigen Softwareentwicklungsprozess (z.B. Spiel „Verflixte 7“).

Im Rahmen der Unterrichtsreihe erlernen sie die einzelnen Phasen Objektorientierte Analyse (OOA), Objektorientiertes Design (OOD) bis hin zur Objektorientierten Programmierung (OOP). In der OOA-Phase werden beteiligte Objekte mit ihren Fähigkeiten und Eigenschaften beispielsweise mit Hilfe der Substantiv-Verb Methode, Use-Case Diagramme oder der CRC-Methode identifiziert. Das OOD dient der Konkretisierung der OOA und Implementationsdiagramme werden erstellt.

Der letzte Schritt ist die Umsetzung des Projekts/Modells mit Java einschließlich der Gestaltung einer grafischen Oberfläche. Das System wird getestet und bei Bedarf verbessert, optional unter der Verwendung weiterer Diagramme.

Zeitbedarf: 20 Stunden

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien oder Materialien**
a) OOA Die OOA dient der Untersuchung der Aufgabenstellung und führt zu einem ersten UML-Modell. Die SuS benutzen dazu die Fachmethoden: USE-Case Diagramme, Substantiv Verb	Die Schülerinnen und Schüler … • ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M),	Use-Case-Diagramme, Substantiv-Verb-Methode, CRC-Karten, Sequenzdiagramme, BlueJ (GUI Editor)

Methode, CRC Karten b) OOD Die OOD dient der Konkretisierung und führt zu einem Implementationsdiagramm c) OOP Mit BlueJ wird das Modell implementiert und unter anderem mit Hilfe des Debuggers und von Sequenzdiagrammen auf seine Lauffähigkeit getestet. d) MVC Konzept Das Modell wird mit Hilfe des Java-Editors um eine einfache grafische Oberfläche erweitert.	• modellieren Klassen mit ihren Attributen, ihren Methoden und Assoziationsbeziehungen (M), • modellieren Klassen unter Verwendung von Vererbung (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare Datensammlungen zu (M), • ordnen Klassen, Attributen und Methoden ihren Sichtbarkeitsbereich zu (M), • stellen den Zustand eines Objekts dar (D), • stellen die Kommunikation zwischen Objekten grafisch dar (M), • stellen Klassen, Assoziations- und Vererbungsbeziehungen in Diagrammen grafisch dar (D), • implementieren Klassen in einer Programmiersprache (I). • analysieren und erläutern einfache Algorithmen und Programme (A), • entwerfen einfache Algorithmen und stellen sie umgangssprachlich und grafisch dar (M), • implementieren Algorithmen unter Verwendung von Variablen und Wertzuweisungen, Kontrollstrukturen sowie Methodenaufrufen (I),	

	• nutzen die im Unterricht eingesetzten Informatiksysteme selbstständig, sicher, zielführend und verantwortungsbewusst (D). • implementieren einfache Algorithmen unter Beachtung der Syntax und Semantik einer Programmiersprache (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I). • dokumentieren Klassen durch Beschreibung der Funktionalität der Methoden (D). • analysieren und erläutern eine objektorientierte Modellierung (A)	

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

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? Welche Auswirkungen hat die informatisch-technische Entwicklung auf das Leben der Menschen?

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 sowie um das Subtyping erweitert.

Zeitbedarf: ca. 14 Stunden

<table>
<tr><th>Unterrichtssequenzen</th><th>Zu entwickelnde Kompetenzen</th><th>Mögliche Beispiele, Medien oder Materialien</th></tr>
<tr><td>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</td><td rowspan="4">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),
- 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).</td><td>2.1 Die Welt ist voller Objekte
Projekteinstieg: Klassenentwurf – step by step</td></tr>
<tr><td>2. Erweiterung der objektorientierten Programmierung
a) Umsetzung einer Anforderung in Entwurfs- und Klassendiagramm
b) Objektkommunikation im Sequenzdiagramm
c) Klassendokumentation
d) Umsetzung von Teilen der Modellierung</td><td>2.2 Gut geplant – Klassenentwurf
2.3 Vererbungshierarchien nutzen</td></tr>
<tr><td>3. Mensch und Technik
a) Verantwortung von Informatikern
b) Automatisierung des Alltags durch Informatik</td><td>Die digitale Welt 001 - Mensch und Technik</td></tr>
<tr><td>4. Übung und Vertiefung der OOM / OOP</td><td>Aufgaben zur Prüfungsvorbereitung</td></tr>
</table>

Unterrichtsvorhaben Q1-2-GK

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?

Vorhabenbezogene Konkretisierung:
Ausgehend von 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. Die Klasse *Queue* selbst wird gemeinsam modelliert und implementiert. Darauf folgt die Erarbeitung der Struktur *Stack*, die mithilfe eines einfachen Anwendungsszenarios eingeführt wird. Hier wird die Klasse Stack von den Schüler/innen weitestgehend selbstständig modelliert und implementiert.
Weitere Aufgaben dienen der Vertiefung und Sicherung sowie der Erarbeitung des FIFO- und LIFO-Prinzips.

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 erarbeitet und in einem weiteren Beispiel vertieft wurden, werden abschließend in einem Anwendungskontext verschiedene lineare Datenstrukturen angewendet. Die Modellierung erfolgt in Entwurfs- und Implementationsdiagrammen.

Zeitbedarf: ca. 20 Stunden

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Mögliche Beispiele, Medien oder Materialien**
1. Die Datenstruktur *Feld* a) Erarbeitung der Anforderungen an eine Datenstruktur b) Wiederholung der Datenstruktur Array, Eigenschaften der Datenstruktur, Standardoperationen für ein und zweidimensionale Arrays c) Modellierung und Implementierung von Anwendungen	Die Schülerinnen und Schüler… - erläutern und begründen methodische Vorgehensweisen, Entwurfs- und Implementationsentscheidungen sowie Aussagen über Informatiksysteme (A) - konstruieren zu kontextbezogenen Problemstellungen informatische Modelle (M) - 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	**Kapitel:** - Anforderungen an eine Datenstruktur - Speichern mit Struktur *neue Wdh. Aufgabe entwickeln, z. B. eine Chart-Top-10,* *eine Aufgabe mit zweidimensionalem Array (vgl. Anforderungen KLP)*
2. Die Datenstruktur *Schlange*		**Kapitel:**

a) Modellierung und Implementierung der 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*	Datensammlungen zu (M), - implementieren auf der Grundlage von Modellen oder Modellausschnitten Computerprogramme (I) - testen und korrigieren Computerprogramme (I) - interpretieren Fehlermeldungen und korrigieren den Quellcode (I), - überführen gegebene textuelle und grafische Darstellungen informatischer Zusammenhänge in die jeweils andere Darstellungsform (D) - stellen informatische Modelle und Abläufe in Texten, Tabellen, Diagrammen und Grafiken dar (D) - stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D) - 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)	- Wer zuerst kommt - Objekte miteinander verketten - Verwaltung und Inhalt - Funktionen der Queue **Aufgaben:** - Warteschlange Büro (Standardoperationen/Basiskompetenz) Kunden warten auf einem Flur, um in ein Büro vorgelassen zu werden. Sie können sich am Ende der Warteschlange anstellen, vorgelassen werden oder müssen alle gehen, wenn die Sprechzeit vorüber ist. - Erweiterte Queue Verkehrskontrolle (Vertiefung) Die Polizei kontrolliert die Fahrzeuge im Hinblick auf ihre Verkehrstauglichkeit. Für die Kontrolle werden die Fahrzeuge aus dem Verkehr gewunken. Es werden so lange Fahrzeuge kontrolliert, bis eine gewissen Menge an Verstößen vorliegt oder Autos kontrolliert wurden.
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		**Kapitel:** - Daten gut abgelegt – Stapel - Funktionen der Datenstruktur Stapel **Aufgaben:** - Standardoperationen/Basiskompetenz (Stapel Münzen/ CDs) zur

Implementierung einer Anwendung auf Basis einer Anforderungsbeschreibung mit Objekten der Klasse *Stack* d) Modellierung und Implementierung einer Anwendung unter Verwendung verschiedener Datenstrukturen (Objekte der Klassen *Queue*, *Stack* und *Array (Palindrom)*)		Umsetzung der gegebenen Funktionen der Klasse *Stack* - Biber und Teller Es gibt große und kleine Biber sowie grüne und braune Teller. Es muss überprüft werden, ob die gestapelten Teller zur Schlange der Biber passen, da die großen Biber nur von den braunen Tellern essen und die kleinen von den grünen. Hierbei müssen sowohl Queue als auch Stack verwendet werden. - Palindrom Es wird überprüft, ob ein beliebiges Wort ein Palindrom ist.
4. Die Datenstruktur *Liste* a) Analyse der Möglichkeiten bisheriger Datenstrukturen zwecks Bestimmung notwendiger Funktionalitäten für komplexere Anwendungen (Abgrenzung zu *Stack*/*Queue*, zusätzliche 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)		**Kapitel:** - Flexibel für alle Fälle – (die) lineare Liste - Funktionen der Datenstruktur Liste **Aufgaben:** - LEDs - Textzeilen verarbeiten

5. Übungen und Vertiefungen zur Verwendung linearer und dynamischer Daten-strukturen anhand weiterer Problemstellungen		Aufgaben zur Prüfungsvorbereitung

Unterrichtsvorhaben Q1-III-GK

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: ca. 15 Stunden

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Mögliche Beispiele, Medien oder Materialien**
1. Eigenschaften von Algorithmen a) Qualitätseigenschaften von Algorithmen b) Strukturierung von Algorithmen mit Hilfe der Strategien „Modularisierung“ und „Teile und Herrsche“ 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	4.1 Ohne Algorithmen läuft nichts 4.2 Teile die Arbeit – rekursive Algorithmen

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	„Modularisierung“ und „Teile und Herrsche“ (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), - testen Programme systematisch anhand von Beispielen (I),	4.3 Suchen – iterativ und rekursiv
3. Sortieren auf Listen und Arrays a) Entwicklung und Implementierung eines iterativen Sortierverfahrens für eine Liste b) Entwicklung und Implementierung eines rekursiven Sortierverfahrens für ein Array c) Untersuchung der beiden Verfahren bzgl. Laufzeit und Speicherplatzbedarf		4.4 Sortieren – iterativ und rekursiv

Unterrichtsvorhaben Q1-IV-GK

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 eines Anwendungskontextes 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).
Ein Anwendungskontext 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 und anhand eines Kartenspiels mit den europäischen Ländern (Beitrag zum Europaschul-Curriculum) der Bezug zur binären Suche hergestellt (https://verlage.westermanngruppe.de/schoeningh/backend/anlage/herunterladen/4576778). 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 (z.B. europäische Länder) mithilfe der Klassen BinarySearchTree und Item modelliert und (ggf. in Teilen) implementiert.

Zeitbedarf: ca. 20 Stunden

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Mögliche Beispiele, Medien oder Materialien**
1. Aufbau von Baumstrukturen und Grundbegriffe a) Erarbeitung der Begriffe Wurzel, Knoten, Blatt, Kante, Grad eines Knotens und	Die Schülerinnen und Schüler - stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D), - erläutern Operationen dynamischer (linea-	6.1 Spielen mit Struktur – Baumstrukturen Projekteinstieg 1: Spielbäume

eines Baumes, Pfad, Tiefe, Ebene, Teilbaum b) Aufbau und Darstellung von Baumstrukturen in verschiedenen Anwendungskontexten	rer oder nicht-linearer) Datenstrukturen (A), - analysieren und erläutern Algorithmen und Programme (A), - stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D).	
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)	- 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),	6.2 Zwei Nachfolger sind genug! - Binäre Bäume Implementation des Projekts Ahnenbaum
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)	- 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),	6.3 Wer Ordnung hält, spart Zeit beim Suchen – Binäre Suchbäume Projekteinstieg 2: Europäische Länder - Binäre Suchbäume Implementation des Projekts Benutzerverwaltung

4. Übung und Vertiefungen der Verwendung von Binärbäumen oder binären Suchbäumen anhand weiterer Problemstellungen		Aufgaben zur Prüfungsvorbereitung

Unterrichtsvorhaben Q2-I-GK

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 und Reichweiten kategorisiert. Ausgehend davon wird der Protokollbegriff entwickelt und anhand des TCP/IP-Schichtenmodells analysiert. Anschließend wird das Client-Server-Prinzip vorgestellt und angewandt.
Sichere Kommunikation in Netzen ist nur dank kryptografischer Verfahren möglich. Stellvertretend werden zwei symmetrische und ein asymmetrisches Verfahren erläutert, angewandt und bewertet.

Zeitbedarf: ca. 12 Stunden

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Mögliche Beispiele, Medien oder 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 Netzwerk-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)	7.1 Menschen kommunizieren – ohne und mit Technik Projekteinstieg: Kommunikation im Wilden Westen
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		7.2 Ohne Protokoll läuft nichts - Netzwerke

3. Aufgabenteilung in Netzwerken durch Server und Client a) Aufbau und Aufgaben der Client-Server-Struktur b) Protokolle zwischen Client und Server		7.3 Einer für alle – Client-Server-Struktur
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		Die digitale Welt 100 – Kryptologie
6. Übung und Vertiefung des Aufbaus von und der Kommunikation in Netzwerken		Aufgaben zur Prüfungsvorbereitung

Unterrichtsvorhaben Q2-II-GK

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: ca. 20 Std.

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Mögliche Beispiele, Medien oder 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) sowie dessen Darstellungsformen; Erschließung der Fachbegriffe Alphabet, Wort, (akzeptierte) Sprache, Determinismus	Die Schülerinnen und Schüler - stellen informatische Modelle und Abläufe in Texten, Tabellen, Diagrammen und Grafiken dar (D) - 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 endliche Automaten (M)	Vom realen Automaten zum Modell Projekteinstieg: Schatzsuche Der Mealy-Automat Der erkennende endliche Automat Wort und Sprache

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	- 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)	
2. Grammatiken regulärer Sprachen a) Erarbeitung der formalen Beschreibung einer regulären Grammatik (Sprache, Terminal und Nicht-Terminal, Produktionen und Produktionsvorschriften) b) Analyse der Eigenschaften einer regulären Grammatik durch deren Entwicklung und Modellierung zu einer gegebenen Problemstellung.		Grammatiken regulärer Sprachen
3. Übungen und Vertiefungen Verwendung endlicher Automaten und Grammatiken regulärer Sprachen		Aufgaben zur Prüfungsvorbereitung

Unterrichtsvorhaben Q2-III-GK

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:

An einem Beispiel 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. Mithilfe des Projekteinstiegs „Tabellen“ können bereits zu einem frühen Zeitpunkt des Unterrichtsvorhabens Redundanzen, Inkonsistenzen und Anomalien problematisiert werden.

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: ca. 20 Stunden

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Mögliche Beispiele, Medien oder 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	Die Schülerinnen und Schüler - erläutern die Eigenschaften und den Aufbau von Datenbanksystemen unter dem Aspekt der sicheren Nutzung (A), - analysieren und erläutern die Syntax und Semantik einer Datenbankabfrage (A), - analysieren und erläutern eine Datenbankmodellierung (A), - erläutern die Eigenschaften normalisierter Datenbankschemata (A), - bestimmen Primär- und Sekundärschlüssel (M), - ermitteln für anwendungsbezogene Problemstellungen Entitäten, zugehörige Attri-	**Kapitel 8 Datenbanken** 8.1 Wissen speichern und verwalten – Datenbanksysteme 8.2 Daten anordnen mit Tabellen Beispiel: Buchhandlung Redundanzen, Anomalien und Inkonsistenzen Projekteinstieg: Tabellen 8.3 Daten filtern mit SQL 8.4 Komplexe Filter

- 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	bute, Relationen und Kardinalitäten (M), - modifizieren eine Datenbankmodellierung (M), - modellieren zu einem Entity-Relationship-Diagramm ein relationales Datenbankschema (M), - überführen Datenbankschemata in vorgegebene Normalformen (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).	
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		8.5 Datenbankentwurf Beispiel: Online-Buchhandel Datenanalyse und Entwurf 8.6 Umsetzung des ER-Modells Entitätsmengen m:n-Beziehungen 1:n-Beziehungen 1:1-Beziehungen Wiederaufgriff des Projekteinstiegs 8.7 Datenbanken verbessern durch Normalformen

c) Normalformen - Überprüfung von Datenbankschemata hinsichtlich der 1. bis 3. Normalform und Normalisierung (um Redundanzen zu vermeiden und Konsistenz zu gewährleisten)		
3. Übung und Vertiefung der Nutzung und Modellierung von relationalen Datenbanken		Aufgaben zur Prüfungsvorbereitung

Unterrichtsvorhaben Q1-I-LK

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? Welche Auswirkungen hat die informatisch-technische Entwicklung auf das Leben der Menschen?

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 sowie um das Subtyping erweitert.

Zeitbedarf: 20 Stunden

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Mögliche Beispiele, Medien oder Materialien**
1. Wiederholung der grundlegenden Konzepte der objektorientierten Programmierung a) Sichtweise der objektorientierten Informatik auf die Welt	Die Schülerinnen und Schüler … - ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M),	**Kapitel 1 Konzepte des objektorientierten Modellierens** 1.1 Modellierung der Realität

b) OOP als informatikspezifische Modellierung der Realität c) Schritte der Softwareentwicklung	- 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), - verwenden bei der Modellierung geeigneter Problemstellungen Möglichkeiten der Polymorphie (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), - nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Daten, zur Organisation von Arbeitsabläufen sowie zur Verteilung und Zusammenführung von Arbeitsanteilen (K) - stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D), - dokumentieren Klassen (D), - stellen die Kommunikation zwischen Objekten grafisch dar (D).	1.2 Die Welt ist voller Objekte Projekteinstieg: Klassenentwurf – step by step
2. Erweiterung der objektorientierten Programmierung a) Umsetzung einer Anforderung in Entwurfs- und Klassendiagramm b) Objektkommunikation im Sequenzdiagramm c) Klassendokumentation d) Umsetzung von Teilen der Modellierung		1.3 Gut geplant – Klassenentwurf 1.4 Hierarchien machen´s einfacher – Vererbung

Unterrichtsvorhaben Q1-II-LK

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?

Vorhabenbezogene Konkretisierung:

Ausgehend von 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 (Biber/Palindrom) wird. Die Operationen push() und pop() werden grafisch dargestellt und implementiert.

Anschließend werden die beiden Prinzipien FIFO und LIFO gegenübergestellt und verglichen.

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 (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. (Dies kann auch im Zusammenhang mit Such- und Sortieralgorithmen geschehen, vgl. Q1-III-LK.)

Zeitbedarf: 30 Stunden

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Mögliche Beispiele, Medien oder Materialien**
1. Die Datenstruktur Feld a) Erarbeitung der Anforderungen an eine Datenstruktur b) Wiederholung der Datenstruktur Array, Eigenschaften der Datenstruktur, Standardoperationen für ein und zweidimensionale Arrays c) Modellierung und	Die Schülerinnen und Schüler … - erläutern Operationen dynamischer (linearer) Datenstrukturen (A), - implementieren Operationen dynamischer (linearer oder nichtlinearer) Datenstrukturen (I), - ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M)	**Kapitel 2 Lineare Datenstrukturen** 2.1 Anforderungen an eine Datenstruktur 2.2 Datenansammlungen fester Größe – Arrays

Implementierung von Anwendungen	- ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare Datensammlungen zu (M), - interpretieren Fehlermeldungen und korrigieren den Quellcode (I), - stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D), - 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), - verwenden bei der Modellierung geeigneter Problemstellungen Möglichkeiten der Polymorphie (M),	
2. Die Datenstruktur Schlange a) Modellierung und Implementierung der 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*		2.3 Wer zuerst kommt... – Schlangen
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,* Erschließen der Standardoperationen c) Modellierung und Implementierung einer Anwendung auf Basis einer Anforderungsbeschreibung mit Objekten der Klasse *Stack* d) Modellierung und Implementierung einer Anwendung unter Verwendung verschiedener		2.4 Daten gut abgelegt – Stapel

Datenstrukturen (Objekte der Klassen *Queue*, *Stack* und *Array (Palindrom)*)		
4. Die Datenstruktur Liste a) Analyse der Möglichkeiten bisheriger Datenstrukturen zwecks Bestimmung notwendiger Funktionalitäten für komplexere Anwendungen (Abgrenzung zu *Stack*/*Queue*, zusätzliche 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)		2.5 Flexibel für alle Fälle – lineare Listen

Unterrichtsvorhaben Q1-III-LK

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. Hier bietet sich die Implementation einer Testanwendung an, die die Laufzeiten unterschiedlicher Sortieralgorithmen misst. Daran anschließend folgen theoretische Analysen der Laufzeit.

Zeitbedarf: 30 Stunden

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Mögliche Beispiele, Medien oder Materialien**
1. Eigenschaften von Algorithmen a) Qualitätseigenschaften von Algorithmen b) Strukturierung von Algorithmen mit Hilfe der Strategien „Modularisierung“ und „Teile und Herrsche“ 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	**Kapitel 3 Algorithmen** 3.1 Ohne Algorithmen läuft nichts 3.2 Teile die Arbeit – rekursive Algorithmen

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	„Modularisierung“ und „Teilen und Herrschen“ (M), - implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), - testen Programme systematisch anhand von Beispielen und mithilfe 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), - 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).	3.3 Suchen – iterativ und rekursiv Projekteinstieg: Laufzeitanalyse experimentell
3. Sortieren auf Listen und Arrays a) Entwicklung und Implementierung eines iterativen Sortierverfahrens für eine Liste (Sortieren durch Einfügen) b) Entwicklung und Implementierung eines rekursiven Sortierverfahrens für eine Liste (Quicksort) c) Untersuchung der beiden Verfahren bzgl. Laufzeit und Speicherplatzbedarf (praktische Tests und theoretische Analyse) d) Weitere Sortierverfahren auf Listen und Arrays (Sortieren durch Auswählen, Mergesort) **4. Verantwortung der Informatik** a) Der Unterschied zwischen Anwender und Produzent von Informatiksystemen b) Informatik und Ethik		3.4 Sortieren – iterativ und rekursiv Implementation einer Testanwendung Die digitale Welt 011 – Verantwortung der Informatik

Unterrichtsvorhaben Q1-IV-LK

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 und anhand eines Kartenspiels mit den europäischen Ländern (Beitrag zum Europaschul-Curriculum) der Bezug zur binären Suche hergestellt
(https://verlage.westermanngruppe.de/schoeningh/backend/anlage/herunterladen/4576778). Die
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 *ComparableContent* 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 *ComparableContent* modelliert und (ggf. in Teilen) implementiert.

Zeitbedarf: 40 Stunden

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Mögliche Beispiele, Medien oder 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	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), - implementieren Operationen dynamischer	**Kapitel 5 Nicht-lineare Datenstrukturen** 5.1 Spielen mit Struktur – Baumstrukturen Projekteinstieg 1: Spielbäume

Baumstrukturen in verschiedenen Anwendungskontexten	(linearer oder nicht-linearer) Datenstrukturen (I), - analysieren und erläutern Algorithmen und Programme (A),	
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)	- 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),	5.2 Zwei Nachfolger sind genug! - Binäre Bäume Implementation des Projekts Ahnenbaum
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)	- 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 (I).	5.3 Wer Ordnung hält, spart Zeit beim Suchen – Binäre Suchbäume Projekteinstieg 2: Europäische Länder - Binäre Suchbäume Implementation des Projekts Benutzerverwaltung
4. Graphen a) Anwendungskontexte für		Projekteinstieg 3: Navigationssysteme 5.4 Navigieren mit Struktur

Graphen, Grundbegriffe b) Modellierung von Graphen mit Adjazenzlisten mit Hilfe der Klassen Graph und GraphNode und Adjazenzmatrizen. c) Algorithmus von Dijkstra d) Tiefen- und Breitensuche		– Graphen 5.5 Modellierung von Graphen 5.6 Kürzester Weg – der Algorithmus von Dijkstra 5.7 Graphen durchsuchen

Unterrichtsvorhaben Q2-I-LK

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? Wie werden Protokolle zur zuverlässigen Kommunikation eingesetzt und in Software umgesetzt?

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 und Reichweiten kategorisiert. Ausgehend davon wird der Protokollbegriff entwickelt und anhand des TCP/IP-Schichtenmodells analysiert.

Anschließend wird das Client-Server-Prinzip vorgestellt und in einer Projektarbeit angewandt. Es werden eigenständig Protokolle entwickelt und in einer Client-Server-Anwendung umgesetzt.

Sichere Kommunikation in Netzen ist nur dank kryptografischer Verfahren möglich. Stellvertretend werden zwei symmetrische und ein asymmetrisches Verfahren erläutert, angewandt und bewertet.

Zeitbedarf: 40 Stunden

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Mögliche Beispiele, Medien oder 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 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), - 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-	**Kapitel 6 Kommunikation in Netzwerken** 6.1 Menschen kommunizieren – ohne und mit Technik Projekteinstieg: Kommunikation im Wilden Westen
2. Aufbau von Netzwerken und Kommunikationsregeln a) Das Netzwerk als Organisationsprinzip der Kommunikation und Möglichkeiten der Ausformung b) Geregelte technische Kommunikation durch		6.2 Ohne Protokoll läuft nichts – Netzwerke

Protokolle in Schichtenmodellen	Kommunikation (I), - analysieren und erläutern Eigenschaften, Funktionsweisen und Einsatzbereiche symmetrischer und asymmetrischer Verschlüsselungsverfahren (A), - nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zur Erschließung, Aufbereitung und Präsentation fachlicher Inhalte (D), - untersuchen und bewerten anhand von Fallbeispielen Auswirkungen des Einsatzes von Informatiksystemen sowie Aspekte der Sicherheit von Informatiksystemen, des Datenschutzes und des Urheberrechts (A).	
3. Aufgabenteilung in Netzwerken durch Server und Client a) Aufbau und Aufgaben der Client-Server-Struktur b) Protokolle zwischen Client und Server c) Implementation von Anwendungen unter Berücksichtigung der Client-Server-Struktur und gegebener Protokolle		6.3 Einer für alle – Client-Server-Struktur Projektarbeit (z.B. Chatanwendung, Spiele wie Schiffe versenken oder Stein, Schere, Papier, Fussballmanager)
4. Kryptologie a) Veranschaulichen und Anwenden von symmetrischen und asymmetrischen kryptographischen Verfahren (Caesar, Vigenère, RSA, Diffie-Hellman Schlüsselaustausch) b) Bewertung der Verfahren hinsichtlich ihrer Sicherheit und ihrem Aufwand		Die digitale Welt 111 – Kryptologie

Unterrichtsvorhaben Q2-II-LK

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 oder durch Kellerautomaten und kontextfreie Grammatiken beschrieben werden und welche nicht? 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? In welchen Anwendungen finden Automaten Einsatz?

Vorhabenbezogene 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. Anhand einer Problemstellung wird die Grenze der endlichen Automaten aufgezeigt und in das Konzept des Kellerautomaten bzw. der kontextfreien Sprachen überführt.

Der Anwendungsbezug zu den theoretischen Grundlagen der Sprachen- und Automatentheorie wird durch das Modellieren und Implementieren von Scannern, Parsern und Interpretern vertieft.

Zeitbedarf: 40 Std.

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Mögliche Beispiele, Medien oder Materialien**
1. Endliche Automaten a) Erarbeitung der formalen Beschreibung eines Mealy-Automaten und der Darstellungsformen	Die Schülerinnen und Schüler - analysieren und erläutern die Eigenschaften endlicher Automaten und Kellerautomaten einschließlich ihres Verhaltens bei bestimmten Eingaben (A),	**Kapitel 4 Endliche Automaten und formale Sprachen** 4.1. Endliche Automaten Projekteinstieg:

b) Erarbeitung der formalen Beschreibung eines deterministischen endlichen Automaten (DEA) sowie dessen Darstellungsformen; Erschließung der Fachbegriffe Alphabet, Wort, (akzeptierte) Sprache, Determinismus c) Analyse der Eigenschaften von DEAs durch die Modellierung eines Automaten zu einer gegebenen Problemstellung, der Modifikation eines Automaten sowie die Überführung der gegebenen Darstellungsform in eine andere	- 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), - 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), - 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 Sprachen im Anwendungszusammenhang (A), - modellieren und implementieren Scanner, Parser und Interpreter zu einer gegebenen regulären Sprache (I), - erläutern die Ausführung eines einfachen maschinennahen Programms sowie die Datenspeicherung auf einer „Von-Neumann-Architektur“ (A), - untersuchen und beurteilen Grenzen des Problemlösens mit Informatiksystemen	Schatzsuche
2. Grammatiken regulärer Sprachen a) Erarbeitung der formalen Beschreibung einer regulären Grammatik (Sprache, Terminal und Nicht-Terminal, Produktionen und Produktionsvorschriften) b) Analyse der Eigenschaften einer regulären Grammatik durch deren Entwicklung und Modellierung zu einer gegebenen Problemstellung.		4.2 Formale Sprachen
3. Scanner, Parser, Interpreter a) Erarbeitung der Konzepte von Scanner, Parser und Interpreter b) Modellierung und Implementation von Scanner, Parser und Interpreter		4.3 Vom Text zum Programm 4.4 Scanner und Parser
4. Übungen und Vertiefungen Verwendung endlicher Automaten und Grammatiken regulärer Sprachen		4.5 Prüfungsvorbereitung

5. Grundsätzliche Arbeitsweise eines Computers und Grenzen der Berechenbarkeit a) Von-Neumann-Architektur und die Ausführung maschinennaher Programme b) Grenzen der Berechenbarkeit anhand des Halteproblems, nicht effizient berechenbare Probleme	(A).	Die digitale Welt 101 – Maschinennahe Programmierung Die digitale Welt 100 – Berechenbarkeit

Unterrichtsvorhaben Q2-III-LK

Thema: Nutzung und Modellierung und Implementation 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 ein, die eine bestehende Datenbank beschreiben und analysieren und mithilfe von SQL-Abfragen Daten gezielt herausfiltern.
Mithilfe des Projekteinstiegs „Tabellen“ können bereits zu einem frühen Zeitpunkt des Unterrichtsvorhabens Redundanzen, Inkonsistenzen und Anomalien problematisiert werden.

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.

Den Abschluss des Unterrichtsvorhaben bildet eine Projektarbeit, bei der eine Datenbankanwendung von den Schülerinnen und Schülern modelliert, implementiert, dokumentiert und getestet wird. Die Implementation der Java-Anwendung wird dabei mit den Klassen *DatabaseConnector* und *QueryResult* (der Materialien für das Zentralabitur NRW) realisiert. Die Projektarbeit vertieft insbesondere die Kompetenzen im Bereich Kommunizieren und Kooperieren, da die Lernenden eigenverantwortlich und kooperativ arbeiten, ihre Arbeitsabläufe bewerten und Ergebnisse vorstellen. Darüber hinaus werden Datenschutzaspekte thematisiert.

Zeitbedarf: 30 Stunden

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Mögliche Beispiele, Medien oder Materialien**
1. Nutzung von relationalen Datenbanken a) Aufbau von Datenbanksystemen und Grundbegriffe - Aufgaben und Eigenschaften eines Datenbanksystems - Erarbeitung der Begriffe	Die Schülerinnen und Schüler … - erläutern die Eigenschaften und den Aufbau von Datenbanksystemen unter dem Aspekt der sicheren Nutzung (A), - analysieren und erläutern die Syntax und Semantik einer Datenbankabfrage (A), - verwenden die Syntax und Semantik einer	**Kapitel 8 Datenbanken** 7.1 Wissen speichern und verwalten – Datenbanksysteme 7.2 Daten anordnen mit Tabellen

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	Datenbankabfragesprache, um Informationen aus einen Datenbanksystem zu extrahieren (I), - ermitteln Ergebnisse von Datenbankabfragen über mehrere verknüpfte Tabellen (D), - ermitteln für anwendungsbezogene Problemstellungen Entitäten, zugehörige Attribute, Relationen und Kardinalitäten (M), - stellen Entitäten mit ihren Attributen und die Beziehungen zwischen Entitäten 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 eine relationales Datenbankschema als Datenbank (I), - analysieren und erläutern eine Datenbankmodellierung (A), - erläutern die Eigenschaften normalisierter Datenbankschemata (A),	Projekteinstieg: Tabellen 7.3 Daten filtern mit SQL 7.4 Komplexe Filter
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	- überprüfen Datenbankschemata auf vorgegebene Normalisierungseigenschaften (D), - überführen Datenbankschemata in die 1. bis 3. Normalform (M), - 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	7.5 Datenbankentwurf 7.6 Umsetzung des ER-Modells Wiederaufgriff des Projekteinstiegs 7.7 Datenbanken verbessern durch Normalformen

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)	einem Informatiksystem (M). - 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).	
3. Implementation von relationalen Datenbanken Modellierung, Implementation, Dokumentation und Test einer Datenbankanwendung in einem Anwendungskontext (Projektarbeit)		7.3 Daten filtern mit SQL 7.4 Komplexe Filter 7.6 Umsetzung des ER-Modells Projektarbeit (z.B. Kassensystem, Cocktailverzeichnis, Umfragetool für die Abizeitung), ggf. auch mit Netzwerkfunktionalität

Unterrichtsvorhaben Q2-IV-LK

Thema: Wiederholung und Vertiefung zur konkreten Vorbereitung auf die Abiturprüfungen

Leitfragen: Welche abiturrelevanten Inhalte wurden im Unterricht thematisiert, welche Kompetenzen wurden erworben und wie sehen konkrete Aufgaben dazu aus?

Vorhabenbezogene Konkretisierung:

Am Beispiel von alten Abituraufgaben werden die Unterrichtsinhalte wiederholt und die erworbenen Kompetenzen angewandt.

Quellen:

- Beispiel für ein an das Lehrwerk Informatik 2 angepasstes Unterrichtsvorhaben als zentraler Bestandteil eines schulinternen Lehrplans zum Kernlehrplan für die gymnasiale Oberstufe, Grund- und Leistungskurs, Schöningh Verlag

- *https://verlage.westermanngruppe.de/schoeningh/backend/anlage/herunterladen/4576776*
- *https://verlage.westermanngruppe.de/schoeningh/backend/anlage/herunterladen/4576777*

2.3 Grundsätze der Leistungsbewertung und Leistungsrückmeldung

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

2.3.1 Beurteilungsbereich Klausuren

Verbindliche Absprachen:

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

Instrumente:

Einführungsphase:	1 Klausur je Halbjahr, Dauer: 2
Qualifikationsphase 1:	2 Klausuren je Halbjahr, Dauer: 2 U-Std. (GK), 3 U-Std. (LK)
Qualifikationsphase 2.1:	2 Klausuren, Dauer: 3 U-Std. (GK), 4 U-Std. (LK)
Qualifikationsphase 2.2:	1 Klausur unter Abiturbeding., Dauer: 225 min (GK), 270 min (LK)

Schriftliche Leistungsüberprüfungen finden in der Regel im Präsenzunterricht statt.
Die Aufgabentypen sowie die Anforderungsbereiche I-III sind entsprechend den Vorgaben in Kapitel 3 des Kernlehrplans zu beachten.

Die dritte Klausur in der Qualifikationsphase 1 kann durch eine Facharbeit ersetzt werden.

Kriterien

Die Bewertung der schriftlichen Leistungen in Klausuren erfolgt über ein Raster mit Hilfspunkten. Die Note ausreichend soll bei Erreichen von 45 % der Hilfspunkte erteilt werden.

2.3.2 Beurteilungsbereich Sonstige Mitarbeit

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

Leistungsaspekte

<u>Mündliche Leistungen</u>

- Beteiligung am Unterrichtsgespräch
- Zusammenfassungen zur Vor- und Nachbereitung des Unterrichts
- Präsentation von Arbeitsergebnissen
- Referate
- Mitarbeit in Partner-/Gruppenarbeitsphasen

<u>Praktische Leistungen am Computer</u>

- Implementierung, Test und Anwendung von Informatiksystemen

Sonstige schriftliche Leistungen

- Lernerfolgsüberprüfung durch kurze schriftliche Übungen
- Bearbeitung von schriftlichen Aufgaben im Unterricht

Kriterien

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

Die Bewertungskriterien stützen sich auf

- die Qualität der Beiträge,
- die Quantität der Beiträge und
- die Kontinuität der Beiträge.

Besonderes Augenmerk ist dabei auf

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

Bei Gruppenarbeiten auch auf

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

Bei Projektarbeit darüber hinaus auf

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

Grundsätze der Leistungsrückmeldung und Beratung

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

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

Die Leistungsrückmeldung kann

- durch ein Gespräch mit der Schülerin oder dem Schüler,
- durch einen Feedbackbogen,

- durch die schriftliche Begründung einer Note oder
- durch eine individuelle Lern-/Förderempfehlung

erfolgen.

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

Die genannten Grundsätze der Leistungsbewertung und Leistungsrückmeldung gelten sowohl für den Präsenz- als auch für den Distanzunterricht. Im Distanzunterricht werden ergänzende Möglichkeiten der Kommunikation genutzt, wie z.B. Chats, Foren oder Videokonferenzen. Die Teilnahme am Präsenz- und Distanzunterricht ist für die Schülerinnen und Schüler verpflichtend und dient der Erfüllung der Schulpflicht.

3. Entscheidungen zu fach- und unterrichtsübergreifenden Fragestellungen

Zusammenarbeit mit anderen Fächern

Im Informatikunterricht werden Kompetenzen anhand informatischer Inhalte in verschiedenen Anwendungskontexten erworben, in denen Schülerinnen und Schülern aus anderen Fächern Kenntnisse mitbringen können. Diese können insbesondere bei der Auswahl und Bearbeitung von Softwareprojekten berücksichtigt werden und in einem hinsichtlich der informatischen Problemstellung angemessenem Maß in den Unterricht Eingang finden. Zudem arbeitet die Fachschaft Informatik im Rahmen es Projektkurses mit der Fachschaft Mathematik eng zusammen.

Vorbereitung auf die Erstellung der Facharbeit

Möglichst schon im zweiten Halbjahr der Einführungsphase, spätestens jedoch im ersten Halbjahr des ersten Jahres der Qualifikationsphase werden im Unterricht an geeigneten Stellen Hinweise zur Erstellung von Facharbeiten gegeben. Das betrifft u. a. Themenvorschläge, Hinweise zu den Anforderungen und zur Bewertung.

Exkursionen

In der Einführungsphase wird im Rahmen des Unterrichtsvorhabens I angestrebt, eine Exkursion zum Heinz-Nixdorf-Museums-Forum durchzuführen. Die außerunterrichtliche Veranstaltung wird im Unterricht vor- und nachbereitet. Weitere Exkursionen sind möglich, z.B. zu Unternehmen aus der IT-Branche.

4. Qualitätssicherung und Evaluation

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

Erstmalig nach Ende der Einführungsphase im Sommer 2015 wurden in einer Sitzung der Fachkonferenz Erfahrungen mit dem neuen schulinternen Curriculum ausgetauscht.

Nach Abschluss eines jeden Schuljahres nimmt die Fachkonferenz Informatik auf der Grundlage ihrer Unterrichtserfahrungen eine Gesamtsicht des schulinternen Curriculums vor und erstellt in Absprache mit den anderen beiden Soester Gymnasien sowie der Gesamtschule ggf. eine Beschlussvorlage für die erste Fachkonferenz des folgenden Schuljahres.