Schulinternen Lehrplan

zum Kernlehrplan für die gymnasiale Oberstufe

am Gymnasium Delbrück

Informatik

(Stand: Mai 2015)

Inhaltsverzeichnis

1 Die Fachgruppe Informatik am Gymnasium Delbrück

Beim Gymnasium Delbrück handelt es sich um eine vier- bis fünfzügige Schule im Zentrum von Delbrück. Das Einzugsgebiet der Schule sind die zehn Delbrücker Orte, sowie die Stadt Hövelhof. Das Fach Informatik wird ab der Jahrgangsstufe 8 im Wahlpflichtbereich II (WP II) dreistündig unterrichtet und von etwa einem Fünftel der Schülerinnen und Schüler besucht. In der zweijährigen Laufzeit dieser Kurse wird in altersstufengerechter Weise unter anderem auf Grundlagen der Algorithmik am Beispiel einer didaktischen Lernumgebung, auf die technische Informatik am Beispiel von Schaltwerken und Schaltnetzen und auf Robotik eingegangen.

In den Jahrgangsstufen 5 und 6 wird ein Kurs zum Umgang mit informatischen Systemen durchgeführt, ein sogenannter Office-Kurs, der jedoch nicht unmittelbar dem Fach Informatik zuzuordnen ist. In der Sekundarstufe II bietet das Gymnasium Delbrück je nach Anwahlen zwei bis drei Grundkurse in Informatik an. Um insbesondere Schülerinnen und Schülern gerecht zu werden, die in der Sekundarstufe I keinen Informatikunterricht besucht haben, wird in Kursen der Einführungsphase besonderer Wert daraufgelegt, dass keine Vorkenntnisse aus der Sekundarstufe I zum erfolgreichen Durchlaufen des Kurses erforderlich sind.

Der Unterricht der Sekundarstufe II wird mit Hilfe der Programmiersprache Java durchgeführt. Durch projektartiges Vorgehen, offene Aufgaben und Möglichkeiten, Problemlösungen zu verfeinern oder zu optimieren, entspricht der Informatikunterricht der Oberstufe in besonderem Maße den Erziehungszielen, Leistungsbereitschaft zu fördern, ohne zu überfordern.

Zurzeit besteht die Fachschaft Informatik aus sieben Lehrkräften, denen drei Computerräume mit jeweils ca. 16 Computerarbeitsplätzen zur Verfügung stehen. Alle Arbeitsplätze sind an das schulinterne Rechnernetz angeschlossen.

2 Entscheidungen zum Unterricht

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

Während der Fachkonferenzbeschluss zum „Übersichtsraster Unterrichtsvorhaben“ zur Gewährleistung vergleichbarer Standards sowie zur Absicherung von Lerngruppenübertritten und Lehrkraftwechseln für alle Mitglieder der Fachkonferenz Bindekraft entfalten soll, beinhaltet die Ausweisung „konkretisierter Unterrichtsvorhaben“ Beispiele und Materialien, die empfehlenden Charakter haben.
Da in den folgenden Unterrichtsvorhaben Inhalte in der Regel anhand von Problemstellungen in Anwendungskontexten bearbeitet werden, werden in einigen Unterrichtsvorhaben jeweils mehrere Inhaltsfelder angesprochen.

2.1 Übersichtsraster Unterrichtsvorhaben in der Einführungsphase

Unterrichtsvorhaben E-I **Thema**: *Was macht Informatik? - Ein Gang durch die Geschichte der Informatik* **Zentrale Kompetenzen**: • Kommunizieren und Kooperieren • Darstellen und Interpretieren • Argumentieren **Inhaltsfelder**: • Informatiksysteme • Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte**: • Einsatz, Nutzung und Aufbau von Informatiksystemen • Wirkung der Automatisierung **Zeitbedarf**: 4 Stunden	***Unterrichtsvorhaben E-II*** **Thema**: *Grundlagen der objektorientierten Analyse, Modellierung und Implementierung* **Zentrale Kompetenzen:** • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Formale Sprachen und Automaten **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Syntax und Semantik einer Programmiersprache **Zeitbedarf**: 8 Stunden
Unterrichtsvorhaben E-III **Thema**: *Algorithmische Grundstrukturen in Java* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung	***Unterrichtsvorhaben E-IV*** **Thema:** *Modellierung und Implementierung von Klassen- und Objektbeziehungen anhand lebensnaher Anforderungsbeispiele* **Zentrale Kompetenzen**: • Kommunizieren und Kooperieren • Darstellen und Interpretieren • Argumentieren • Modellieren • Implementieren **Inhaltsfelder**: • Daten und ihre Strukturierung

• Algorithmen • Formale Sprachen und Automaten **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Syntax und Semantik einer Programmiersprache • Analyse, Entwurf und Implementierung einfacher Algorithmen **Zeitbedarf**: 18 Stunden	• Algorithmen • Formale Sprachen und Automaten **Inhaltliche Schwerpunkte**: • Objekte und Klassen • Syntax und Semantik einer Programmiersprache • Analyse, Entwurf und Implementierung einfacher Algorithmen **Zeitbedarf**: 20 Stunden
UnterrichtsvorhabenE-V **Thema**: *Such- und Sortieralgorithmen* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Algorithmen • Daten und ihre Strukturierung **Inhaltliche Schwerpunkte:** • Algorithmen zum Suchen und Sortieren • Analyse, Entwurf und Implementierung einfacher Algorithmen • Objekte und Klassen **Zeitbedarf**: 9 Stunden	***Unterrichtsvorhaben E-VI*** **Thema**: *Das ist die digitale Welt! - Einführung in die Grundlagen, Anwendungsgebiete und Verarbeitung binärer Codierung* **Zentrale Kompetenzen**: • Kommunizieren und Kooperieren • Darstellen und Interpretieren • Argumentieren **Inhaltsfelder**: • Informatiksysteme • Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte**: • Binäre Codierung und Verarbeitung • Besondere Eigenschaften der digitalen Speicherung und Verarbeitung von Daten • Das von Neumann-Konzept als Modell für die digitale Informationsverarbeitung von Computern **Zeitbedarf**: 8 Stunden
Unterrichtsvorhaben E-VII **Thema**: *Leben in der digitalen Welt – Immer mehr Möglichkeiten und immer mehr Gefahren!?* **Zentrale Kompetenzen**: • Kommunizieren und Kooperieren • Darstellen und Interpretieren • Argumentieren	

Inhaltsfelder: • Informatiksysteme • Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte**: • Geschichte der automatischen Datenverarbeitung • Wirkungen der Automatisierung • Dateisystem **Zeitbedarf**: 10 Stunden	**Summe Einführungsphase:** **77 Stunden**

2.2 Konkretisierte Unterrichtsvorhaben Einführungsphase

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

Die Schülerinnen und Schüler

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

2.2.1 Unterrichtsvorhaben EF-I

Thema: Was macht Informatik? - Ein Gang durch die Geschichte der Informatik

Leitfragen: Was macht Informatik? Welche fundamentalen Entwicklungen und Erfindungen sowie welche Personen haben die Informatik in ihrem Werdegang beeinflusst?

Vorhabenbezogene Konkretisierung:

Im ersten Unterrichtsvorhaben werden zentrale Entwicklungen und Erfindungen erkundet, die die elektronische Datenverarbeitung maßgeblich beeinflusst haben. Die zentralen Elemente der Datenspeicherung, Datenverarbeitung und Computervernetzung werden mithilfe geeigneter Materialien erarbeitet und in einer Ausstellung in der Schule – wenn möglich kursübergreifend - präsentiert.

Während dieser Erarbeitung werden das Schulnetzwerk und die Einzelplatzrechner so erkundet, dass ein kompetenter Umgang mit diesem ermöglicht wird.

Zeitbedarf: 4 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Informatiksysteme und ihr genereller Aufbau a) Zentrale Komponenten der Datenspeicherung b) Zentrale Komponenten der Datenverarbeitung c) Zentrale Komponenten der Vernetzung	Die Schülerinnen und Schüler • bewerten anhand von Fallbeispielen die Auswirkungen des Einsatzes • von Informatiksystemen (A) • erläutern wesentliche Grundlagen der Geschichte der digitalen Datenverarbeitung (A), • nutzen die im Unterricht eingesetzten Informatiksysteme selbstständig, sicher, zielführend und verantwortungsbewusst (D)	
2. Der kompetente Umgang mit dem Schulnetzwerk a) Erstellen und Anlegen von Ordnerstrukturen b) Sortieren von Dateien und Ordnern c) Eingabe von Befehlen über Eingabeaufforderung d) Einzelrechner und Netzwerk e) Sicherheit und Datenschutz		Informationen zum Schulnetzwerk des GyD

2.2.2 Unterrichtsvorhaben EF-II

Thema: Grundlagen der objektorientierten Analyse, Modellierung und Implementierung

Leitfragen: Wie lassen sich Gegenstandsbereiche informatisch modellieren und in einem Green-foot-Szenario informatisch realisieren?

Vorhabenbezogene Konkretisierung:

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

Dazu werden zunächst konkrete Gegenstandsbereiche aus der Lebenswelt der Schülerinnen und Schüler analysiert und im Sinne des objektorientierten Paradigmas strukturiert. Dabei werden die grundlegenden Begriffe der Objektorientierung und Modellierungswerkzeuge wie Objektdiagramme und Klassendiagramme eingeführt.

Im Anschluss wird die objektorientierte Analyse für ein Greenfoot-Szenario durchgeführt. Die vom Szenario vorgegebenen Klassen werden von Schülerinnen und Schülern in Teilen analysiert und entsprechende Objekte anhand einfacher Problemstellungen erprobt. Die Lernenden implementieren und testen einfache Programme. Die Greenfoot-Umgebung ermöglicht es, Beziehungen zwischen Klassen zu einem späteren Zeitpunkt zu thematisieren.

So kann der Fokus hier auf Grundlagen wie der Unterscheidung zwischen Klasse und Objekt, Attribute, Methoden, Objektidentität und Objektzustand gelegt werden.

Da auf die Verwendung von Kontrollstrukturen verzichtet wird und der Quellcode aus einer rein linearen Sequenz besteht, ist auf diese Weise eine Fokussierung auf die Grundlagen der Objektorientierung möglich, ohne dass algorithmische Probleme ablenken.

Zeitbedarf: 8 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Identifikation von Objekten und Klassen a) An einem lebensweltnahen Beispiel werden Objekte und Klassen im Sinne der objektorientierten Modellierung eingeführt. b) Objekte werden durch Objektdiagramme, Klassen durch Klassendiagramme dargestellt. c) Die Modellierungen werden einem konkreten Anwendungsfall entsprechend angepasst.	Die Schülerinnen und Schüler • ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften und ihre Operationen (M), • stellen den Zustand eines Objekts dar (D), • modellieren Klassen mit ihren Attributen und ihren Methoden (M), • implementieren einfache Algorithmen unter Beachtung der Syntax und Semantik einer Programmiersprache (I), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I).	Anhand von alltäglichen Beispielen wird das Prinzip von Objekten, Klassen Eigenschaften und Fähigkeit „informatisiert“
2. Analyse von Objekten und Klassen im Greenfoot-Szenario a) Schritte der objektorientierten Analyse, Modellierung und Implementation b) Analyse und Erprobung der Objekte im Greenfoot-Szenario		Am Greenfoot-Szenario „Roboter“ oder „Planetenerkundung“ werden Objekte, Klassen, Klassendokumentation Objektinspektion, Methodenaufrufe, Objektidentität und Objektzustand erkundet
3. Implementierung einfacher Algorithmen a) Quelltext einer Java-Klasse b) Implementation eigener Methoden, Dokumentation mit Java-Doc c) Programme übersetzen (Aufgabe des Compilers) und testen		Am Greenfoot-Szenario „Roboter“ oder „Planetenerkundung“ werden erste Algorithmen mit Kontrollschleifen entwickelt

2.2.3 Unterrichtsvorhaben EF-III

Thema: Algorithmische Grundstrukturen in Java

Leitfragen: Wie lassen sich Aktionen von Objekten flexibel realisieren?

Vorhabenbezogene Konkretisierung:

Das Ziel dieses Unterrichtsvorhabens besteht darin, das Verhalten von Objekten flexibel zu programmieren. Ein erster Schwerpunkt liegt dabei auf der Erarbeitung von Kontrollstrukturen. Die Strukturen Wiederholung und bedingte Anweisung werden an einfachen Beispielen eingeführt und anschließend anhand komplexerer Problemstellungen erprobt. Da die zu entwickelnden Algorithmen zunehmend umfangreicher werden, werden systematische Vorgehensweisen zur Entwicklung von Algorithmen thematisiert.

Ein zweiter Schwerpunkt des Unterrichtsvorhabens liegt auf dem Einsatz von Variablen. Beginnend mit lokalen Variablen, die in Methoden und Zählschleifen zum Einsatz kommen über Variablen in Form von Parametern und Rückgabewerten von Methoden bis hin zu Variablen, die die Attribute einer Klasse realisieren, lernen die Schülerinnen und Schüler die unterschiedlichen Einsatzmöglichkeiten des Variablenkonzepts anzuwenden. Des Weiteren werden Methoden ausgehend von ihrer Signatur her entwickelt und dokumentiert.

Zeitbedarf: 18 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Algorithmen a) Wiederholungen (WHILE - Schleife) b) bedingte Anweisungen c) Verknüpfung von Bedingungen durch die logischen Funktionen UND, ODER und NICHT d) Systematisierung des Vorgehens zur Entwicklung von Algorithmen zur Lösung komplexerer Probleme	Die Schülerinnen und Schüler • analysieren und erläutern einfache Algorithmen und Programme (A), • entwerfen einfache Algorithmen und stellen sie umgangssprachlich und grafisch dar (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen zu (M), • modifizieren einfache Algorithmen und Programme (I), • implementieren Algorithmen unter Verwendung von Variablen und Wertzuweisungen, Kontrollstrukturen sowie Methodenaufrufen (I), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), • implementieren einfache Algorithmen unter Beachtung der Syntax und	Am Greenfoot-Szenario „Roboter“ oder „Planetenerkundung“ werden erste Algorithmen mit Kontrollschleifen entwickelt
2. **Variablen und Methoden** a) Implementierung eigener Methoden mit lokalen Variablen, auch zur Realisierung einer Zählschleife b) Implementierung eigener Methoden mit Parameterübergabe und/oder Rückgabewert c) Implementierung von		In einer neuen Entwicklungsumgebung (z.B. JavaEditor) werden Klassen um Attribute - unter Verständnis des Variablenkonzepts – und Methoden erweitert bzw. eigene Klassen entwickelt und implementiert.

Konstruktoren d) Realisierung von Attributen	Semantik einer Programmiersprache (I), • testen Programme schrittweise anhand von Beispielen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I).	

2.2.4 Unterrichtsvorhaben EF-IV

Thema: Modellierung und Implementierung von Klassen- und Objektbeziehungen anhand lebensnaher Anforderungsbeispiele

Leitfragen: Wie werden realistische Systeme anforderungsspezifisch reduziert, als Entwurf modelliert und implementiert? Wie kommunizieren Objekte und wie wird dieses dargestellt und realisiert?

Vorhabenbezogene Konkretisierung:

Das Unterrichtsvorhaben hat die Entwicklung von Objekt - und Klassenbeziehungen zum Schwerpunkt. Dazu wird von der Realität, Objektidentifizierung über den Entwurf bis zur Implementation kleine Softwareprodukte in Teilen oder ganzheitlich erstellt.

Zuerst identifizieren die Schülerinnen und Schüler Objekte anhand eines einfachen Beispiels (einfacher Bruchrechner) und stellen diese dar. Aus diesen Objekten werden Klassen und ihre Beziehun-gen in Entwurfsdiagrammen erstellt. Die Schülerinnen und Schüler sind in der Lage, diesen einfachen Bruchrechner zu implementieren. Daher erfolgt in der frühen Phase dieses Unterrichtsvorhabens bereits die Implementation eines einfachen Bruchrechners.

Die Grenzen des einfachen Bruchrechners werden im folgenden Schritt herausgearbeitet, die dann auf natürliche Weise zu einer erweiterten Anforderung an einen Bruchrechner führt.

Nach dem ersten Modellierungs- und Implementationsschritt werden über Klassendokumentationen und der Darstellung von Objektkommunikationen an Implementationsdiagramme entwickelt. Ebenso wird das Konzept der Vererbung sowie seiner Vorteile erarbeitet.

In der vierten Sequenz werden die Implementationsdiagramme unter Berücksichtigung der Klassen-dokumentationen in JAVA-Klassen programmiert.

Schließlich sind die Schülerinnen und Schüler in der Lage, eigene kleine Softwareprojekte zu entwi-ckeln. Ausgehend von der Dekonstruktion und Erweiterung eines Spiels (Tetramino) wird ein weiteres Projekt (z.B. Pentomino) von Grund auf modelliert und implementiert. Dabei können arbeitsteilige Vorgehensweisen zum Einsatz kommen. In diesem Zusammenhang wird auch das Erstellen von graphischen Benutzeroberflächen eingeführt.

Zeitbedarf: 20 Stunden

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Eine einfache Klasse a) Formulierung der Funktionalitäten einer einfachen Klasse b) Implementation einer einfachen Klasse **2. Grenzen einfacher Klassen** a) Formulierung der Anforderungen und Identifikation von Objekten des Bruchrechners b) Zusammenfassung der Objekte zu Klassen im Entwurfsdiagramm und Erweiterung um Datentypen/Methoden **3. Entwurf eines „komplexen" Modells** a) Vom Entwurfs- zum Implementationsdiagramm b) Erweiterung von getter-/setter-Methoden und Konstruktor c) Das Grundprinzip und die Vorteile der Vererbungsbeziehungen d) Entwicklung einer geeigneten Klassendokumentation e) Formulierung der Beziehungen zwischen den Klassen f) Kommunikation zwischen verschiedenen Objekten g) Dokumentation der Klassen	Die Schülerinnen und Schüler • analysieren und erläutern eine objektorientierte Modellierung (A), • stellen die Kommunikation zwischen Objekten grafisch dar (M), • ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modellieren Klassen mit ihren Attributen, ihren Methoden und Assoziationsbeziehungen (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare Da-tensammlungen zu (M), • ordnen Klassen, Attributen und Methoden ihren Sichtbarkeitsbereich zu (M), • modellieren Klassen unter Verwendung von Vererbung (M), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), • testen Programme schrittweise anhand von Beispielen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • 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). • stellen Klassen, Assoziations- und Vererbungsbeziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen durch Beschreibung der Funktionalität der Methoden (D)	*Beispiel: Bruchrechner* Die Schüler entwickeln die Funktionalität eines Bruchrechners Die Schülerinnen und Schüler entwickeln und implementieren eine einfache Klasse eines Bruchrechners, der über die einfachen Grundfunktionalitäten (Addition und Multiplikation) *Informationsmaterial: Entwurfsdiagramm* Die Schülerinnen und Schüler erkennen die Grenzen des „einfachen" Bruchrechners und „fordern" eine Erweiterung. *Beispiel: Bruchrechnerprojekt* Die Schülerinnen und Schüler dekonstruieren den einfachen Bruchrechner im Sinne des MVC – Konzeptes *Informationsmaterial: MVC-Konzept* Die Schüler begeben sich auf „die Ebene" des Computers, erweitern diesen und gestalten Beziehungen der Klassen *Informationsmaterial: Assoziationen* *Informationsmaterial: Objektdiagramm* *Informationsmaterial: Implementationsdiagramm* Interaktion- und Kommunikation der Bruchobjekte im Sequenzdiagramm
4. Implementation des komplexen Modells a) Realisierung der Klassen des Modells in einer Programmiersprache b) Das Geheimnisprinzip wird umgesetzt c) Einzelne Klassen und das Geamtsystem werden anhand der Anforderungen und Dokumentationen auf ihre Korrektheit überprüft.		

5. Softwareprojekt a) Analyse und Dekonstruktion eines Spiels (Modelle, Quelltexte) b) Erweiterung des Spiels um weitere Funktionalitäten c) Modellierung eines Spiels aufgrund einer Anforderungsbeschreibung, inklusive einer graphischen Benutzeroberfläche (arbeitsteilige) Implementation des Spiels		*Beispiel: Bruchrechner* Die Schülerinnen und Schüler implementieren das Modell gem. den Vorgaben in JAVA *Informationsmaterial: JAVA-Grundstrukturen* Die Schülerinnen und Schüler entwickeln Testszenarien zum Beweis der Korrektheit des Modells. *Beispiel: Pentomino* Die Schülerinnen und Schüler erarbeiten den grundsätzlichen Aufbau (und das Modell) von Tetraminos am Beispiel eines vorgegebenen Quelltextes. *Material: Tetramino.java* Die Schülerinnen und Schüler erweitern das Modell zu den Tetraminos zur Darstellung von Pentominos Die Schülerinnen und Schüler implementieren das Spiel *Material: sum.java und spiel.java*

2.2.5 Unterrichtsvorhaben EF-V

Thema: Such- und Sortieralgorithmen

Leitfragen: *Wie können Objekte bzw. Daten effizient gesucht und sortiert werden?*

Vorhabenbezogene Konkretisierung:

Dieses Unterrichtsvorhaben beschäftigt sich mit der Erarbeitung von Such- und Sortieralgorithmen. Der Schwerpunkt des Vorhabens liegt dabei auf den Algorithmen selbst und nicht auf deren Implementierung in einer Programmiersprache.

Zunächst lernen die Schülerinnen und Schüler das Feld als eine erste Datensammlung kennen. Optional können nun zunächst die wesentlichen Eigenschaften von Algorithmen wie z.B. Korrektheit, Terminiertheit, Effizienz und Verständlichkeit sowie die Schritte einer Algorithmenentwicklung erarbeitet werden (Klärung der Anforderung, Visualisierung, Zerlegung in Teilprobleme).

Daran anschließend lernen die Schülerinnen und Schüler zunächst Strategien des Suchens (lineare Suche, binäre Suche, Hashing) und dann des Sortierens (Selection Sort, Insertion Sort, Bubble Sort) kennen. Die Projekteinstiege dienen dazu, die jeweiligen Strategien handlungsorientiert zu erkunden und intuitive Effizienzbetrachtungen der Suchalgorithmen vorzunehmen.

Schließlich wird die Effizienz unterschiedlicher Sortierverfahren beurteilt.

Zeitbedarf: 9 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Explorative Erarbeitung von Suchverfahren a) Erkundung von Strategien für das Sortieren von Daten b) Betrachtung der Vorgehensweise von Computern beim Sortieren c) Vergleich von Verfahren durch intuitive Effizienzbetrachtungen.	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 Zeit und Speicherplatzbedarf (A), • ordnen Attributen lineare Datenansammlungen zu (M).	Anhand eines Kartenspiels werden die Unterschiede zwischen Computeralgorithmen zur Sortierung und menschlichen Strategien erarbeitet. Vorgegebene Sortieralgorithmen werden anhand von Vergleichs- und Vertauschungszählungen verglichen
2. Modellierung und Implementation von Datenansammlungen a) Modellierung von Attributen als Felder b) Deklaration, Instanziierung und Zugriffe auf ein Feld		Ausgehend von 1. wird die Notwendigkeit einer Datenstruktur ermittelt und anhand einfacher Beispiele modelliert und implementiert.
3. Systematisierung von Algorithmen und Effizienzbetrachtungen a) Formulierung (falls selbst gefunden) oder Erläuterung von mehreren Algorithmen im Pseudocode b) Anwendung von Sortieralgorithmen auf verschiedene Beispiele c) Bewertung von Algorithmen anhand der Anzahl der nötigen Vergleiche d) Effizienzbetrachtungen an einem konkreten Beispiel bezüglich der Rechenzeit und des Speicherplatzbedarfs e) Analyse eines weiteren Sortieralgorithmus (sofern nicht in		Dieses wird durch die Einfache Sortierverfahren werden anhand ihres Quellcodes analysiert und bewertet.

a) bereits geschehen)		

2.2.6 Unterrichtsvorhaben EF-VI

Thema: Das ist die digitale Welt! - Einführung in die Grundlagen, Anwendungsgebiete und Verarbeitung binärer Codierung

Leitfragen: *Wie werden binäre Informationen gespeichert und wie können sie davon ausgehend weiter verarbeitet werden? Wie unterscheiden sich analoge Medien und Geräte von digitalen Medien und Geräten? Wie ist der Grundaufbau einer digitalen Rechenmaschine?*

Vorhabenbezogene Konkretisierung:
Das Unterrichtsvorhaben hat die binäre Speicherung und Verarbeitung sowie deren Besonderheiten zum Inhalt.
Im ersten Schritt erarbeiten die Schülerinnen und Schüler anhand ihnen bekannter technischer Gegenstände die Gemeinsamkeiten, Unterschiede und Besonderheiten der jeweiligen analogen und digitalen Version. Nach dieser ersten grundlegenden Einordnung des digitalen Prinzips wenden die Schülerinnen und Schüler das Binäre als Zahlensystem mit arithmetischen und logischen Operationen an und codieren Zeichen binär. Zum Abschluss soll der grundlegende Aufbau eines Rechnersystems im Sinne der von-Neumann-Architektur erarbeitet werden, wobei der Schwerpunkt auf dem Verstehen des Rechenzyklus eines von-Neumann-Rechners liegt.

Zeitbedarf: 8 Stunden

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Unterscheidung zwischen Information und Daten a) Analoge vs. digitale Daten b) Digitale Informationsverarbeitung	Die Schülerinnen und Schüler • wissen, dass Information immer durch Daten dargestellt werden müssen, bevor sie von Informatiksystemen verarbeitet oder transportiert werden können (K) • können die Bedeutung binärer Daten für die technische Verarbeitung erklären (K) • stellen Informationen/Daten in unterschiedlicher Form dar (D) • können einfache Umwandlungsalgorithmen in einer Programmiersprache implementieren (I) • kennen Standardcodes (z.B. ASCII, Unicode, UTF-8) (D) • können einfache Rechenoperationen mit binärcodierten Zahlen ausführen (D) • können die Komponenten eines von-Neumann-Rechners differenziert beschreiben und ihr Zusammenwirken in den einzelnen Phasen erläutern (A) • nutzen das Internet zur Informationsbeschaffung (K)	*<u>Beispiel: Handout</u>* Die Schülerinnen und Schüler erstellen ein Handout zum Verhältnis von Daten und Informationen Schülerinnen und Schüler recherchieren selbstständig im Internet oder in öffentlichen Bibliotheken
2. Strukturierte Darstellung von Informationen a) Bits und Bytes b) Binär- und Hexadezimaldarstellung von Zahlen und Zeichen c) Euklidischer Algorithmus zur Umwandlung in versch. Stellenwertsysteme Rechnen im Binärsystem		*<u>Beispiel: Das Verfahren von Euklid</u>* Die Schüler stellen verschiedene Informationen (Daten) in binärer Codierung dar. *Informationsmaterial: Codierung* *Informationsmaterial: Binärsystem*

		Die Schülerinnen und Schüler entwickeln aus dem euklidischen Algorithmus ein Verfahren zur Umrechnung in verschiedene Stellenwertsysteme und implementieren dies in einer Programmiersprache *Informationsmaterial: JAVA-Grundstrukturen* *Material: Euklid.java*
3. Das von-Neumann-Konzept a) Komponenten eines von-Neumann-Rechners und ihr Zusammenwirken im Modell		*Beispiel: von-Neumann-Simulator* Die Schüler erarbeiten sich anhand eines **einfachen** Simulators des von-Neumann-Rechners dessen Aufbau und vollziehen die 5 Phasen des von-Neumann-Zyklus nach Material: Know-How-Computer

2.2.7 Unterrichtsvorhaben EF-VII

Thema: Leben in der digitalen Welt – Immer mehr Möglichkeiten und immer mehr Gefahren!?

Leitfragen: *Welche Entwicklungen, Ideen und Erfindungen haben zur heutigen Informatik geführt? Welche Auswirkungen hat die Informatik für das Leben des modernen Menschen?*

Vorhabenbezogene Konkretisierung:
Das Unterrichtsvorhaben stellt aktuelle Entwicklungsstränge der Informatik auf unterschiedlichen Wirkungsebenen in den Fokus. Dabei wird zudem beispielhaft analysiert und bewertet, welche Möglichkeiten und Gefahren die moderne Informationsverarbeitung mit sich bringt.
Ausgehend von dieser Betrachtung kann die aktuelle Informatik hinsichtlich ihrer Leistungsfähigkeit analysiert werden. Dabei soll herausgestellt werden, welche positiven und negativen Folgen Informatiksysteme jetzt und in Zukunft mit sich bringen können.

Zeitbedarf: 10 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Aktuelles aus der Informatik a) Aktuelle Nachrichten aus der „Informatik“ zu Datenspeicherung, Maschinen, Vernetzung werden ausgewertet. b) Anhand der unterschiedlichen Schwerpunkte sollen universelle Tendenzen der Entwicklung der Informationsverarbeitung erarbeitet werden.	Die Schülerinnen und Schüler • bewerten anhand von Fallbeispielen die Auswirkungen des Einsatzes von Informatiksystemen (A), • erläutern wesentliche Grundlagen der Geschichte der digitalen Datenverarbeitung (A), • nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation. (K).	Zeitungstexte oder Dokumentation bieten Anlass zur Auseinandersetzung mit informatischen Themen
2. Die Informationsverarbeitung und ihre Möglichkeiten und Gefahren a) Ausgehend von 1. werden Tendenzen der Entwicklung der Informatik erarbeitet b) Informatik wird als Hilfswissenschaft klassifiziert, die weit über ihren originären Bereich Effizienz- und Leistungssteigerungen erzeugt c) Anhand von Fallbeispielen werden technische und organisatorische Vorteile, sowie deren datenschutzrechtlichen Nachteile betrachtet.		Ausgesuchte Fallbeispiele werden hinsichtlich ihrer Entwicklungstendenzen wie Auswirkungen auf den Arbeitsplatz, Datenschutz usw. klassifiziert und ausgewertet.

2.3 Übersichtsraster Unterrichtsvorhaben in der Einführungsphase

<table>
<tr><th colspan="2">Qualifikationsphase – Q1 – GK/LK</th></tr>
<tr><td><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
- Informatik, Mensch und Gesellschaft

Inhaltliche Schwerpunkte:
- Objekte und Klassen
- Wirkung der Automatisierung

Zeitbedarf: 14/25 Stunden</td><td><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: 20/30 Stunden</td></tr>
<tr><td><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

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

Zeitbedarf: 20/25 Stunden</td><td><u>Unterrichtsvorhaben Q1-IV</u>

Thema: Automaten und formale Sprachen

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

Inhaltsfelder:
- Daten und ihre Strukturierung
- Algorithmen
- 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
- Scanner, Parser und Interpreter für reg. Sprachen
- Kellerautomaten
- Kontextfreie Sprachen, Grammatiken

Zeitbedarf: 20/35 Stunden</td></tr>
<tr><td colspan="2">Summe Qualifikationsphase 1: 74/110 Stunden</td></tr>
</table>

<table>
<tr><th colspan="2">Qualifikationsphase – Q2 – GK/LK</th></tr>
<tr><td><u>Unterrichtsvorhaben Q2-I</u>

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

Zentrale Kompetenzen:
- Argumentieren
- Darstellen und Interpretieren
- Modellieren
- Implementieren
- Kommunizieren und Kooperieren
-
Inhaltsfelder:
- Daten und ihre Strukturierung
- Algorithmen
- Formale Sprachen und Automaten
-
Inhaltliche Schwerpunkte:
- Objekte und Klassen
- Analyse, Entwurf und Implementierung von Algorithmen
- Algorithmen in ausgewählten informatischen Kontexten
- Syntax und Semantik einer Programmiersprache

Zeitbedarf: 20/25 Stunden</td><td><u>Unterrichtsvorhaben Q2-II</u>

Thema: Modellierung und Implementierung dynamischer nichtlinearer Datenstrukturen am Beispiel der Graphen

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

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

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

Zeitbedarf: 25 Stunden</td></tr>
<tr><td><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
- Formale Sprachen und Automaten
- Informatik, Mensch und Gesellschaft

Inhaltliche Schwerpunkte:
- Datenbanken
- Algorithmen in ausgewählten informatischen Kontexten
- Syntax und Semantik einer Programmiersprache
- Sicherheit

Zeitbedarf: 20/25 Stunden</td><td><u>Unterrichtsvorhaben Q2-IV</u>

Thema: Aufbau von und Kommunikation in Netzwerken

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

Inhaltsfelder:
- Informatiksysteme
- Informatik, Mensch und Gesellschaft

Inhaltliche Schwerpunkte:
- Einzelrechner und Rechnernetzwerke
- Sicherheit
- Nutzung von Informatiksystemen, Wirkungen der Automatisierung

Zeitbedarf: 16 Stunden</td></tr>
<tr><td colspan="2">Summe Qualifikationsphase 1: 56/91 Stunden</td></tr>
</table>

2.3.1 Unterrichtsvorhaben Q1-I

Thema: Wiederholung und Vertiefung der objektorientierten Modellierung

Leitfragen:

Wie wird aus einem anwendungsbezogenen Sachkontext ein informatisches Klassenmodell entwickelt? Wie werden Attribute, Methoden und Beziehungen identifiziert, den Klassen zugeordnet und dargestellt? 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: 14/**25** Stunden

Sequenzierung des Unterrichtsvorhabens: s. nächste Seite

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Wiederholung der grundlegenden Konzepte der objektorientierten Programmierung a) Sichtweise der objektorientierten Informatik auf die Welt b) OOP als informatikspezifische Modellierung der Realität c) Schritte der Softwareentwicklung	Die Schülerinnen und Schüler … • analysieren und erläutern objektorientierte Modellierungen (**A**), • modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (**M**), • ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (**M**), • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (**M**), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (**I**), • wenden eine didaktisch orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (**I**), • stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (**D**), • dokumentieren Klassen (**D**), • stellen die Kommunikation zwischen Objekten grafisch dar (**D**), • 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**).	2.1 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		2.2 Gut geplant – Klassenentwurf 2.3 Vererbungshierarchien nutzen
3. Mensch und Technik a) Verantwortung von Informatikern b) Automatisierung des Alltags durch Informatik		Die digitale Welt 001 - Mensch und Technik
4. Übung und Vertiefung der OOM / OOP		Prüfungsvorbereitung

2.3.2 Unterrichtsvorhaben Q1-2

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

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

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

Zeitbedarf: 20/30 Stunden

Sequenzierung des Unterrichtsvorhabens: s. nächste Seite

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, 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 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	**Aufgaben** Aufgabe entwickeln, z. B. eine Chart-Top-10, eine Aufgabe mit zweidimensionalem Array (vgl. Anforderungen KLP)
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		**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 gewisse 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 Implementierung einer Anwendung auf Basis einer Anforderungsbeschreibung mit Objekten der Klasse Queue d) Modellierung und Implementierung einer Anwendung unter Verwendung verschiedener Datenstrukturen (Objekte der Klassen Queue, Stack und Array (Palindrom))	Nutzung dokumentierter Klassenbibliotheken (I)	• Standardoperationen/Basiskompetenz (Stapel Münzen/ CDs) zur 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. • PalindromEs 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)		**Aufgaben** • LEDs • Textzeilen verarbeiten

5. Übungen und Vertiefungen zur Verwendung linearer und dynamischer Datenstrukturen anhand weiterer Problemstellungen		Prüfungsvorbereitung Projekteinstieg Heldenspiel Mit dem Heldenspiel können alle im Kapitel behandelten Datenstrukturen erarbeitet werden. Das Spiel kann bis zu einem beliebigen Grad realisiert werden, sodass es sowohl als Einstieg als auch als ein umfassendes Projekt für lineare Datenstrukturen genutzt werden kann.

2.3.3 Unterrichtsvorhaben Q1-III

Thema:

Algorithmen zum Suchen und Sortieren auf linearen Datenstrukturen

Leitfragen:

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

Vorhabenbezogene Konkretisierung:

Zunächst werden anhand eines Anwendungsbeispiels übergreifende Algorithmuseigenschaften (wie Korrektheit, Effizienz und Verständlichkeit) erarbeitet und Schritte der Algorithmusentwicklung wiederholt. Dabei kommen Struktogramme zur Darstellung von Algorithmen zum Einsatz.

Als besondere Struktur von Algorithmen wird die Rekursion an Beispielen veranschaulicht und gegenüber der Iteration abgegrenzt. Rekursive Algorithmen werden von den Schülerinnen und Schülern analysiert und selbst entwickelt.

In der zweiten Unterrichtssequenz geht es um die Frage, wie Daten in linearen Strukturen (lineare Liste und Array) (wieder-)gefunden werden können. Die lineare Suche als iteratives und die binäre Suche als rekursives Verfahren werden veranschaulicht und implementiert. Die Bewertung der Algorithmen erfolgt, indem jeweils die Anzahl der Vergleichsoperationen und der Speicherbedarf ermittelt wird.

Möchte man Daten effizient in einer linearen Struktur wiederfinden, so rückt zwangsläufig die Frage nach einer Sortierstrategie in den Fokus. Es wird mindestens ein iteratives und ein rekursives Sortierverfahren erarbeitet und implementiert sowie ihre Effizienz bewertet.

Zeitbedarf: 20/25 Stunden

Sequenzierung des Unterrichtsvorhabens: s. nächste Seite

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, 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 „Modularisierung“ und „Teilen und Herrschen“ (**M**), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (**I**), • testen Programme systematisch anhand von Beispielen (**I**). • implementieren und erläutern iterative und rekursive Such- und Sortierverfahren (**I**), • beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (**A**), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (**A**), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (**I**), • interpretieren Fehlermeldungen und korrigieren den Quellcode (**I**), • - testen Programme systematisch anhand von Beispielen (**I**),	
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		
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		

2.3.4 Unterrichtsvorhaben Q-I IV

Thema:

Automaten und formale Sprachen

Leitfragen:

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

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.

Scanner, Parser und Interpreter für kontextfreie Sprachen werden implementiert bzw. analysiert.

Aufbauend auf die Erkenntnis der Begrenztheit des Modells der regulären Sprachen wird die nächste Ebene in der Chomsky-Hierarchie betrachtet: kontextfreie Sprachen bzw. Grammatiken und die mit dieser Sprachebene korrespondierenden Kellerautomaten.

Zeitbedarf: 20/35 Stunden

Sequenzierung des Unterrichtsvorhabens: s. nächste Seite

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, 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) (Moore-Automat) sowie dessen Darstellungsformen; Erschließung der Fachbegriffe Alphabet, Wort, (akzeptierte) Sprache, Determinismus c) Analyse der Eigenschaften von DEA durch die Modellierung eines Automaten zu einer gegebenen Problemstellung, der Modifikation eines Automaten sowie die Überführung der gegebenen Darstellungsform in eine andere	Die Schülerinnen und Schüler • stellen informatische Modelle und Abläufe in Texten, Tabellen, Diagrammen und Grafiken dar (D) • analysieren und erläutern die Eigenschaften endlicher Automaten bzw. Kellerautomaten einschließlich ihres Verhaltens bei bestimmten Eingaben (A) • ermitteln die Sprache, die ein endlicher Automat akzeptiert, bzw. die ein Kellerautomat akzeptiert(D) • entwickeln und modifizieren zu einer Problemstellung endlicher Automaten (M) • stellen endliche Automaten in Tabellen und Graphen dar und überführen sie in die jeweils andere Darstellungsform (D) • entwickeln zur Grammatik einer regulären Sprache einen zugehörigen endlichen Automaten (M) • entwickeln zur Grammatik einer kontextfreien Sprache einen zugehörigen Kellerautomaten (M) • analysieren und erläutern Grammatiken regulärer Sprachen bzw. Kontextfreier Sprachen (A) • modifizieren Grammatiken regulärer Sprachen bzw. kontextfreier Sprachen (M) • entwickeln zu einer regulären Sprache eine Grammatik, die die Sprache erzeugt (M) • entwickeln zu einer 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) • zeigen die Grenzen endlicher Automaten und regulärer	
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. c) Scanner, Parser, Interpreter für reguläre Sprachen d) Erweiterung des Automatenmodells hin zu Kellerautomaten und des Modells der regulären Grammatik hin zu kontextfreien Grammatik		

3. Übungen und Vertiefungen Verwendung endlicher Automaten und Grammatiken regulärer Sprachen Projekteinstieg Erarbeitung der formalen Beschreibung und Überprüfung des Verhaltens eines erkennenden Automaten auf bestimmte Eingaben	Grammatiken auf • analysieren und entwickeln Scanner, Parser und Interpreter für reguläre Beispielsprachen (A, I) • erkennen, dass sich bestimmte Sprachen nicht mehr mit dem Modell der regulären Sprache erfassen lassen (A) • analysieren und erläutern Grammatiken kontextfreier Sprachen (A)	

2.3.5 Unterrichtsvorhaben Q2-I

Thema:

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

Leitfragen:

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

Vorhabenbezogene Konkretisierung:

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

Anschließend rückt der Fokus auf die binären Bäume, deren rekursiver Aufbau für die Traversierung der Datenstruktur genutzt wird. Die Preorder-Traversierung wird verwendet, um einen gespeicherten Inhalt in einem Binärbaum zu finden (Tiefensuche).

Der Anwendungskontext Ahnenbaum wird mithilfe der Klasse BinaryTree (der Materialien für das Zentralabitur in NRW) modelliert und (ggf. in Teilen) implementiert. Dabei wird u. a. die Erzeugung eines Binärbaums mithilfe der beiden Konstruktoren der Klasse BinaryTree thematisiert.

Möchte man Daten geordnet speichern, so bietet sich die Struktur des binären Suchbaums an. An Beispielen wird zunächst das Prinzip des binären Suchbaums erarbeitet. Die Operationen des Suchens, Einfügens, Löschens und der sortierten Ausgabe werden thematisiert.

Um Daten in einem Anwendungskontext mithilfe eines binären Suchbaums verwalten zu können, müssen sie in eine Ordnung gebracht werden können, d. h. sie müssen vergleichbar sein. Diese Vorgabe wird mithilfe des Interfaces Item realisiert, das alle Klassen, dessen Objekte in einem Suchbaum verwaltet werden sollen, implementieren müssen. Auf diese Weise wird ein Anwendungskontext (Benutzerverwaltung) mithilfe der Klassen BinarySearchTree und Item modelliert und (ggf. in Teilen) implementiert.

Zeitbedarf: 20/25 Stunden

Sequenzierung des Unterrichtsvorhabens: s. nächste Seite

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Aufbau von Baumstrukturen und Grundbegriffe a) Erarbeitung der Begriffe Wurzel, Knoten, Blatt, Kante, Grad eines Knotens und eines Baumes, Pfad, Tiefe, Ebene, Teilbaum b) Aufbau und Darstellung von Baumstrukturen in verschiedenen Anwendungskontexten	Die Schülerinnen und Schüler • stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D), • erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), • analysieren und erläutern Algorithmen und Programme (A), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D). • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M), • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M), • verwenden bei der Modellierung geeigneter Problemstellungen die Möglichkeiten der Polymorphie (M), • entwickeln iterative und rekursive Algorithmen unter Nutzung der Konstruktionsstrategien „Modularisierung“ und „Teilen und Herrschen“ (M), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • modifizieren Algorithmen und Programme (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I),	
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)		
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)		

4. Übung und Vertiefungen der Verwendung von Binärbäumen oder binären Suchbäumen anhand weiterer Problemstellungen	• testen Programme systematisch anhand von Beispielen (I),	

2.3.6 Unterrichtsvorhaben Q2-II

Thema:

Modellierung und Implementierung dynamischer nichtlinearer Datenstrukturen am Beispiel der Graphen

Leitfragen:

Bei welchen Problemstellungen reichen die bekannten Datenstrukturen nicht aus? Welche Möglichkeiten gibt es, flexibel miteinander verknüpfte Informationen zu verwalten? Wie hängen die Datenstrukturen Graph, Baum und und lineare Strukturen (Liste, Queue, Stack) zusammen?

Vorhabenbezogene Konkretisierung:

Nach Analyse einer Problemstellung in einem geeigneten Anwendungskontext (z. B. Adventure-Game, Wolf, dem Schaf und dem Kohlkopf-Problem, das Eulerkreisproblem, Nikolausproblem), in dem Daten in Form eines Graphen verwaltet werden, werden der Aufbau (Knoten, Kanten, gerichtete und ungerichtete Graphen, Teilgraphen, Knotengrad, Wege und Kreise) behandelt. Anhand exemplarischer Problemstellungen wird die Repräsentation von Graphen (Adjazenzliste/-matrix) analysiert. Die Operationen der Klasse Graph werden erläutert und im Anwendungszusammenhang bei der Lösung grundlegender Probleme (z.B. Zyklensuche, topologisches Sortieren) genutzt. Dabei werden Standardtraversierungsverfahren (Tiefen- und Breitensuche) exemplarisch herausgearbeitet. In einem geeigneten Anwendungskontextes (z.B. Navigationssystem) wird die kürzeste Wege-Problematik mihilfe geeigneter Algorithmen (Dijkstra, Kruskal, Prim) unter Nutzung der Klasse Graph behandelt.

Zeitbedarf: 30 Stunden

Sequenzierung des Unterrichtsvorhabens: s. nächste Seite

Unterrichtssequenz	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Analyse von Graphenstrukturen in verschiedenen Kontexten a) Erarbeitung der Begriffe Knoten, Kanten, gerichtete und ungerichtete Graphen, Teilgraphen, Knotengrad, Wege und Kreise im Anwendungskontext. b) Modellierung eines Graphen als Entwurfs- und Implementationsdiagramm c) Implementation der Graphklassen	Die Schülerinnen und Schüler • analysieren einen gegebenen Anwendungskontext (**A**) • stellen das hinter dem Anwendungskontext liegende Problem unter Zuhilfenahme der ihnen bekannten nichtlinearen Struktur Baum dar (**D**) • modellieren Klassen (Graph, Node, Edge) mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (**M**) • dokumentieren Klassen durch Beschreibung der Funktonalität der Methoden (**D**) • ermitteln bei der Analyse von Graphen deren Eigenschaften (**M**) • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (**I**).	Beispiele: - Eulerkreisproblem - Das Haus vom Nikolaus - Adventure-Game Medien: Materialien: - ZA Graphklassen
2. Repräsentation von Graphen und Suchstrategien auf Graphen a) Adjazenzliste und –matrix als Repräsentationen eines Graphen b) Anpassung der Graphklassen c) Suchstrategien auf Graphen d) **Exkurs**: Zyklensuche und topologisches Sortieren: Anwendungen von	Die Schülerinnen und Schüler • ordnen Daten linearen und nichtlinearen Datensammlungen zu (**M**), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (**D**) • analysieren und erläutern Algorithmen und Programme (**A**), • entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien	Arbeitsblätter zu … - Adjazenzmatrix/-liste - Tiefen-/Breitensuche - Topologisches Sortieren - Zyklensuche Materialien … - JAVA-Quellcode für Zyklensuche

Suchstrategien auf Graphen	„Modularisierung“ und „Teile und Herrsche“ und „Backtracking“ (**M**), • implementieren und erläutern rekursive Suchverfahren auf Graphen unter Berücksichtigung dynamischer Datenstrukturen (**I**), • dokumentieren Klassen (**D**), • beurteilen den Einsatz von Suchstrategien auf Graphen im Anwendungskontext (**A**) • interpretieren Fehlermeldungen und korrigieren den Quellcode (**I**), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (**I**),	- JAVA-Quellcode für das topologische Sortieren
3. Die Datenstruktur des Graphen im Anwendungskontext unter Nutzung der Klasse Graph a) Ermittlung minimaler Verbindungskosten als Standardanwendung für Graphen b) Die Datenstruktur PriorityQueue im Anwendungskontext c) Modellierung und Implementierung verschiedener Problemstellungen unter Verwendung der Graphklassen	Die Schülerinnen und Schüler • modellieren Anwendungssituationen als Graphen (**M**) • modellieren die PriorityQueue unter Verwendung von Vererbung (**M**) • analysieren und erläutern Algorithmen zur Bestimmung kürzester Wege (**A**), • implementieren den Dijsktraalgorithmus zur Bestimmung des kürzesten Weges unter Nutzung der Graphklassen (**I**), • testen Programme systematisch anhand von Beispielen (**I**), • dokumentieren Klassen (**D**),	Projektarbeit in einem größeren Kontext: Arbeitsblätter - Der chinesische Kaiser Priori-Tii Kiu und seine Tochter Dijsktra - Das Wanderwegproblem - Länge von Telefonleitungen, Wasserleitungen, … - Minimale Spannbäume Materialien - JAVA-Quellcode für PriorityQueue - JAVA-Quellcode für DIjkstraTupel

	• modellieren Klassen (PriorityQueue, Dijkstratupel) mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • strukturieren den Arbeitsprozess, vereinbaren Schnittstellen und führen Ergebnisse zusammen (**K**),	

2.3.7 Unterrichtsvorhaben Q2-III

Thema: Nutzung und Modellierung von relationalen Datenbanken in Anwendungskontexten

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

Vorhabenbezogene Konkretisierung:

Am Beispiel eines Online-Buchhandels werden 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: 20/25 Stunden

Unterrichtssequenz	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Nutzung von relationalen Datenbanken a. Aufbau von Datenbanksystemen und Grundbegriffe • Aufgaben und Eigenschaften eines Datenbanksystems • Erarbeitung der Begriffe Tabelle, Attribut, Attributwert, Datensatz, Datentyp, Primärschlüssel, Datenbankschema • Problematisierung von Redundanzen, Anomalien und Inkonsistenzen b. SQL-Abfragen • Erarbeitung der grundlegenden Sprachelemente von SQL (SELECT(DISTINCT), FROM, WHERE, JOIN) • Analyse und Erarbeitung von SQL-Abfragen (AND, OR, NOT, UNION, AS, GROUP BY,ORDER BY, ASC, DESC, COUNT, MAX, MIN, SUM, Arithmetische Operatoren: +, -, *, /, (…), Vergleichsoperatoren: =, <>, >, <, >=, <=, LIKE, BETWEEN, IN, IS NULL, geschachtelte Select-Ausdrücke) c. Vertiefung an einem weiteren Datenbankbeispiel	Die Schülerinnen und Schüler • erläutern die Eigenschaften und den Aufbau von Datenbanksystemen unter dem Aspekt der sicheren Nutzung (**A**), • analysieren und erläutern die Syntax und Semantik einer Datenbankabfrage (**A**), • analysieren und erläutern eine Datenbankmodellierung (**A**), • erläutern die Eigenschaften normalisierter Datenbankschemata (**A**), • bestimmen Primär- und Sekundärschlüssel (**M**), • ermitteln für anwendungsbezogene Problemstellungen Entitäten, zugehörige Attribute, Relationen und Kardinalitäten (**M**), • modifizieren eine Datenbankmodellierung (**M**), • modellieren zu einem Entity-Relationship-Diagramm ein relationales Datenbankschema (**M**), • 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**).	Aufgaben

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 • Normalformen c. 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		Prüfungsvorbereitung

2.3.8 Unterrichtsvorhaben Q2-IV

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: 16 Stunden

Sequenzierung des Unterrichtsvorhabens: s. nächste Seite

Unterrichtssequenz	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Technische Kommunikation als Fortführung natürlicher Kommunikation a. Kommunikation im Shannon-Weaver-Modell b. Kriterien von technischen Kommunikationsarten c. Die Geschichte der technischen Kommunikation	Die Schülerinnen und Schüler • beschreiben und erläutern Topologien, die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken (**A**), • nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zum Erschließen, zur Aufbereitung und Präsentation fachlicher Inhalte (**D**), • analysieren und erläutern Eigenschaften und Einsatzbereiche symmetrischer und asymmetrischer Verschlüsselungsverfahren (**A**).	
2. Aufbau von Netzwerken und Kommunikationsregeln a. Das Netzwerk als Organisationsprinzip der Kommunikation und Möglichkeiten der Ausformung b. Geregelte technische Kommunikation durch Protokolle in Schichtenmodellen		
3. Aufgabenteilung in Netzwerken durch Server und Client a. Aufbau und Aufgaben der Client-Server-Struktur b. Protokolle zwischen Client und Server		
4. Kryptologie a. Veranschaulichen und Anwenden von symmetrischen und asymmetrischen kryptographischen Verfahren (Caesar, Vigenère, RSA) b. b) Bewertung der Verfahren hinsichtlich ihrer Sicherheit und ihrem Aufwand		
5. Übung und Vertiefung des Aufbaus von und der Kommunikation in Netzwerken		

2.4 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 des Gymnasiums Delbrück 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.

2.4.1 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.

Klausuren
Einführungsphase: 1 Klausur im 1. Halbjahr, 2 Klausuren im 2. Halbjahr
Dauer der Klausur: 2 Unterrichtsstunden

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

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

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

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

2.4.2 Beurteilungsbereich Sonstige Mitarbeit

siehe: http://www.gymnasium-delbrueck.de/unterricht/faecher/informatik/leistungskonzept-if

3 Qualitätssicherung und Evaluation

Das schulinterne Curriculum ist zunächst bis 2017 für den ersten Durchgang durch die gymnasiale Oberstufe nach Erlass des Kernlehrplanes verbindlich. Erstmalig nach Ende der Einführungsphase im Sommer 2015, werden in einer Sitzung der Fachkonferenz Erfahrungen ausgetauscht und ggf. Änderungen für den nächsten Durchgang der Einführungsphase beschlossen, um erkannten ungünstigen Entscheidungen schnellstmöglich entgegenwirken zu können. Nach Abschluss des Abiturs 2017 wir die Fachkonferenz Informatik auf der Grundlage ihrer Unterrichtserfahrungen eine Gesamtsicht des schulinternen Curriculums vornehmen und ggf. eine Beschlussvorlage für die erste Fachkonferenz des folgenden Schuljahres erstellen.