GEORG-BÜCHNER-GYMNASIUM AM RHEIN

Schulinterner Lehrplan
Gymnasium - Sekundarstufe II

Informatik

(Fassung vom 28.02.2023)

Inhaltsverzeichnis

1 Rahmenbedingungen

Ziel dieses schulinternen Lehrplanes des Georg-Büchner-Gymnasiums Düsseldorf (GBG) ist es, das überaus komplexe und sich ständig weiter entwickelnde Themengebiet der Informatik für die unterrichtenden Lehrer*innen in zentrale Unterrichtsvorhaben so zu zerlegen, dass die Schüler*innen die im Kernlehrplan geforderten Kompetenzen bis zur Hochschulreife erlangen können. Dazu sind zunächst die Rahmenbedingungen der Fachschaft, der technischen Ausstattung und der Schülerschaft des GBG relevant.

Derzeit besteht die Fachschaft aus drei aktiven, d.h. unterrichtenden Lehrer*innen und zwei weiteren Mitgliedern. Im einzelnen sind dies Frau Demus, Herr Dr. Ledabo, Herr Mallmann, Herr Töns und Herr Ulrich.

Am GBG wird das Fach Informatik in der EF und in der Qualifikationsphase (Q1 und Q2) in Grundkursen angeboten. Ein Leistungskurs in Informatik kann als schulübergreifender Kurs in Düsseldorf belegt werden.

Die Durchführung des Faches Informatik ist durch die Ausstattung gewährleistet. Es wird an Desktop-PCs oder an mobilen Laptops mit der jeweiligen Software gearbeitet. Dabei geht das GBG prinzipiell davon aus, dass keinerlei Vorkenntnisse in der Programmierung eines Rechners vorhanden sind.

2 Stoffauswahl und Obligatorik

Gemäß Kernlehrplan Informatik (Kernlehrplan für die Sekundarstufe II – Gymnasium/Gesamtschule in Nordrhein-Westfalen, Heft 4725, 2014) und gemäß den Vorgaben zum Zentralabitur lässt sich das Schulfach Informatik im wesentlichen in die Inhaltsfelder

(1) Daten und ihre Strukturierung

(2) Algorithmen

(3) Formale Sprachen und Automaten

(4) Informatiksysteme

(5) Informatik, Mensch und Gesellschaft

einteilen. Alle Inhaltsfelder werden dabei in verschiedenen Kursabschnitten immer wieder thematisiert um einem spiralförmig angeordnetem Curriculum Rechnung zu tragen. Daraus sollen sich in diesem Lehrplan konkrete Unterrichtsvorhaben ergeben, die in den folgenden Abschnitten erläutert werden. Für die Auswahl der Inhalte ist ein jeweiliger Anwendungsbezug obligatorisch, der sowohl als Motivation wie auch als exemplarisches Beispiel für das Unterrichtsthema dienen kann.

Schließlich fordert der Kernlehrplan auch eine vielfältige Unterrichtsmethodik, die sowohl selbstständige wie auch projektorientierte Arbeitsformen umfasst. Hierzu werden die Kompetenzbereiche

(A) Argumentieren

(M) Modellieren

(I) Implementieren

(D) Darstellen und Interpretieren

(K) Kommunizieren und Kooperieren

ausgewiesen, die durch die Unterrichtsvorhaben in verschiedenster Art und Weise berührt werden. Dabei deckt kein Unterrichtsvorhaben eine Kompetenz vollständig ab, so wie auch ein Inhaltsfeld nicht vollständig von nur einem Unterrichtsvorhaben abgedeckt werden kann (Spiralcurriculum).

Obige Kompetenzen sollen die SchülerInnen natürlich nicht nur für das Abitur sondern auch für das Leben danach erwerben. Bestenfalls stehen ihnen auch informatische Konzepte weiterhin zur Verfügung, die sich fachübergreifend anwenden lassen und ihnen im Verlauf ihrer weiteren Ausbildung hilfreich sind. Gemeint sind damit z.B. die Modellierung von Sachzusammenhängen im Rahmen der Objektorientierung oder im Rahmen eines Datenbankdesigns, ebenso wie das Zerlegen von Produktionsabläufen für die Erstellung eines Algorithmus. Die hierzu besprochenen Verfahren lassen sich auch allgemeiner in vielen Bereichen der Wirtschaft anwenden.

Die folgenden Unterrichtsvorhaben stellen Minimalanforderungen an die oben genannten Inhaltsfelder und Kompetenzbereiche für das Ende der jeweiligen Jahrgangsstufe dar. Die angegebenen Unterrichtssequenzen sind daher

ungefähre Vorschläge, die die angegebenen Kompetenzstufen gewährleisten, aber durch die Lehrkraft natürlich weiter ausgearbeitet werden müssen um zu einer brauchbaren Reihen- und Stundenplanung zu gelangen.

2.1 Übersicht zu den Unterrichtsvorhaben

Die im Folgenden beschriebenen Unterrichtsvorhaben sind auch als tabellarische Übersicht im Querformat erhältlich.

2.2 Konkretisierte Unterrichtsvorhaben in der Einführungsphase

Die Einführungsphase soll die Schüler*innen an die gymnasiale Arbeit insbesondere in Informatik heranführen. Die Themen sollen daher einen Überblick über das gesamte Fach bieten, so dass die Schüler*innen ihre Fachwahl zum Wechsel in die Qualifikationsphase auf einer fundierten Basis machen können. Des Weiteren dürfen diese Themen auch nicht bis in alle Tiefen ausgelotet werden, weil der Überblick so verloren geht und sich für die Einführungsphase unnötige Schwierigkeiten ergeben. Die "Feinheiten" sind somit eher der Qualifikationsphase vorbehalten (vgl. Spiralcurriculum).

Ein wesentlicher zu erlernender Teil der Informatik am GBG ist die objektorientierte Programmierung in der Programmiersprache Java. Ihre grundlegende Struktur wird in der Einführungsphase unter dem allgemeineren Konzept der Information und des Informationsflusses eingeführt und in der Qualifikationsphase verfeinert. So kommen in der Einführungsphase auch eher "fertige" Lernumgebungen (z.B. JavaKara, Greenfoot, GLOOP, ...) zum Einsatz, die an die zentralen Elemente heranführen ohne durch zu viele Details den Blick darauf zu versperren. Nichtsdestotrotz werden auch entsprechende Entwicklungswerkzeuge (z.B. BlueJ, Eclipse, ...) verwendet, mit denen in der Qualifikationsphase weitergearbeitet wird.

Unterrichtsvorhaben EF-I

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

Leitfragen: Womit beschäftigt sich die Wissenschaft der Informatik? Welche technischen Hilfsmittel (insbesondere die schulische und heimische Ausstattung) werden genutzt? Wie werden sie genutzt?

Vorhabenbezogene Konkretisierung: Das erste Unterrichtsvorhaben stellt eine allgemeine Einführung in das Fach Informatik dar. Zunächst wird auf den Begriff der Information eingegangen und die Zerlegung in Daten und ihre Interpretation mittels eines Formats thematisiert (z.B. formatierter Text, BMP-Dateiformat, Morse-Code, Bild im Morse-Code?).

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. Hier wird insofern nochmal ein besonderer Blick auf den Informationsbegriff geworfen, dass Daten auch als Steuerbefehle eines Programms gesehen werden können (Maschinenbefehlsformat, Befehlstabelle eines Prozessors).

Es empfiehlt sich ein erster Exkurs in den Variablenbegriff, bei dem der Variableninhalt als das eigentliche Datum und der Variablentyp als sein Format angesehen wird.

Anschließend wird auf die Übertragung von Daten im Sinne des Sender-Empfänger-Modells eingegangen. Dabei wird eine überblickartige Vorstellung der Kommunikation von Rechnern in Netzwerken erarbeitet (z.B. anhand des POP3-Protokolls).

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

Zu entwickelnde Kompetenzen: Die Schüler*innen

beschreiben und erläutern den Aufbau und die Arbeitsweise singulärer Rechner am Beispiel der „Von-Neumann-Architektur“ (A),

nutzen die im Unterricht eingesetzten Informatiksysteme selbstständig, sicher, zielführend und verantwortungsbewusst (D),

nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation (K).

Zeitbedarf: 9 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Beispiele, Medien, Materialien**
1. Information, deren Kodierung und Speicherung Informatik als Wissenschaft der Verarbeitung von Informationen Darstellung von Informationen in Schrift, Bild und Ton Speichern von Daten mit informatischen Systemen am Beispiel der Schulrechner Vereinbarung von Richtlinien zur Datenspeicherung auf den Schulrechnern (z.B. Ordnerstruktur, Dateibezeichner usw.)	*Beispiel*: Textkodierung Darstellung von Texten in verschiedenen Formaten *Material*: Infos zum Morse-Code *Beispiel*: Bildkodierung Kodierung von Bildinformationen in Raster- und Vektorgrafiken *Material*: RGB-Modell
2. Aufbau informatischer Systeme Identifikation typischer Komponenten informatischer Systeme und anschließende Beschränkung auf das Wesentliche, Herleitung der „Von-Neumann-Architektur" Identifikation des EVA-Prinzips (Eingabe-Verarbeitung-Ausgabe) als Prinzip der Verarbeitung von Daten und Grundlage der „Von-Neumann-Architektur"	*Material:* Demonstrationshardware Durch Demontage eines Demonstrationsrechners entdecken die Schüler*innen die verschiedenen Hardwarekomponenten eines Informatiksystems. *Material:* Visualisierungen des von-Neumann-Zyklus im Internet
3. Informations- und Datenübermittlung in Netzen „Sender-Empfänger-Modell" und seine Bedeutung für die Eindeutigkeit von Kommunikation Informatische Kommunikation in Rechnernetzen am Beispiel des Schulnetzwerks (z.B. Benutzeranmeldung, Netzwerkordner, Zugriffsrechte, Client-Server)	*Beispiel:* Rollenspiel zum E-Mail-Versand Schüler*innen übernehmen die Rollen von Client und E-Mail-Server und erfahren die Notwendigkeit eines "geleiteten Gesprächs" zur korrekten Übermittlung.

Unterrichtssequenzen	Beispiele, Medien, Materialien
Grundlagen der technischen Umsetzung von Rechner-kommunikation am Beispiel des Internets (z.B. Netzwerkadresse, Paketvermittlung, Protokoll) Richtlinien zum verantwortungsvollen Umgang mit dem Internet	

Unterrichtsvorhaben EF-II

Thema: Grundlagen der objektorientierten Analyse, Modellierung und Implementierung

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

Vorhabenbezogene Konkretisierung: Zunächst werden konkrete Gegenstandsbereiche aus der Lebenswelt der Schüler*innen analysiert und im Sinne des objektorientierten Paradigmas strukturiert. Dabei werden die grundlegenden Begriffe der Objektorientierung und Modellierungswerkzeuge wie Objektkarten, Klassenkarten oder Beziehungsdiagramme eingeführt.

Im Anschluss wird mit der Realisierung erster Projekte begonnen. Dazu muss der grundlegende Aufbau einer Java-Klasse thematisiert und zwischen Deklaration, Initialisierung und Methodenaufrufen unterschieden werden. Nach der Implementierung oben genannter Gegenstandsbereiche soll an dieser Stelle auch schon mit der Swing-Bibliothek gearbeitet werden. Dabei sollen einfache GUIs erstellt werden, wobei Buttons etc. zwar verwendet werden, aber noch funktionslos bleiben (da dies fortgeschrittene Konzepte erfordert).

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

Zeitbedarf: 10 Stunden

Zu entwickelnde Kompetenzen: Die Schüler*innen

ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M),

modellieren Klassen mit ihren Attributen, ihren Methoden und Assoziationsbeziehungen (M),

stellen die Kommunikation zwischen Objekten grafisch dar (M),

implementieren einfache Algorithmen unter Beachtung der Syntax und Semantik einer Programmiersprache (I),

stellen den Zustand eines Objekts dar (D).

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Beispiele, Medien, Materialen**
1. Identifikation von Objekten Anhand eines lebensweltnahen Beispiels werden Objekte im Sinne der objektorientierten Modellierung eingeführt. Objekte werden mit Objektkarten visualisiert und mit sinnvollen Attributen und „Fähigkeiten", d.h. Methoden versehen. Manche Objekte sind prinzipiell typgleich und werden so zu einer Objektsorte bzw. Objektklasse zusammengefasst. Vertiefung: Modellierung weiterer Beispiele ähnlichen Musters	*Beispiel:* Vogelschwarm Schüler*innen betrachten einen Vogelschwarm als Menge gleichartiger Objekte, die in einer Klasse mit Attributen und Methoden zusammengefasst werden können.
2. Analyse von Klassen didaktischer Lernumgebungen Objektorientierte Programmierung als modularisiertes Vorgehen (Entwicklung von Problemlösungen auf	*Material:* Java-API

Unterrichtssequenzen	Beispiele, Medien, Materialen
Grundlage vorhandener Klassen) Teilanalyse der Klassen der Swing-Bibliothek	
3. Implementierung einer einfachen GUI Grundaufbau einer Java-Klasse Konzeption einer GUI mit verschiedenen Objekten Deklaration und Initialisierung von Objekten Methodenaufrufe mit Parameter-übergabe zur Manipulation von Objekteigenschaften (z.B. Farbe, Position)	*Beispiel:* GUI eines Chat-Programms *Materialien:* Java-API

Unterrichtsvorhaben EF-III

Thema: Grundlagen der objektorientierten Programmierung und algorithmischer Grundstrukturen in Java

Leitfragen: Wie lassen sich Abläufe (z.B. Animationen) und Simulationen (ggf. unter Berücksichtigung von Tastatureingaben) realisieren?

Vorhabenbezogene Konkretisierung: Der Schwerpunkt dieses Unterrichtsvorhabens liegt auf der Entwicklung mehrerer kleiner Projekte, bei denen Animationen in Panels dargestellt werden sollen. Für die Umsetzung dieses Projekts werden Kontrollstrukturen in Form von Schleifen und Verzweigungen benötigt und eingeführt. Damit eine Animation ohne Interaktion in einer Schleife, welche in festgelegten Intervallen das Panel neu zeichnet, ablaufen kann, muss entweder auf Thread.sleep() zurückgegriffen werden (was einen Vorgriff auf Exceptions erfordert) oder mit einem Timer-Objekt gearbeitet werden (was etwas komplizierter ist und Interfaces und ggf. Extraklassen erfordert – allerdings ist diese Methode eleganter, da für den eigentlichen Animationsloop keine Schleife mehr benötigt wird).

Sind an einem solchen Beispiel im Schwerpunkt Schleifen und Verzweigungen eingeführt worden, sollen diese Konzepte an weiteren Beispielprojekten eingeübt werden. Auch die Erzeugung größerer Mengen von Objekten und deren Verwaltung in einem Feld kann ein Anlass zur Verwendung von Kontrollstrukturen sein.

Die Programmierung von Benutzerinteraktionen ist an dieser Stelle sehr spannend, aber auch knifflig, da im Prinzip mit ActionListenern gearbeitet werden müsste. Es ist aber vorstellbar, dass man diese Komplexität noch vor den Schüler*innen verbirgt, indem man einfach eine anonyme innere Klasse für den ActionListener verwendet, so dass den Schüler*innen eine sehr kompakte Lösung zur Verfügung steht (wenngleich die Schüler*innen diese Lösung noch nicht vollständig verstehen können).

Das Unterrichtsvorhaben schließt mit einem Projekt, in welchem die Schüler*innen mehr als nur die Klasse erstellen müssen. Elemente des Modells müssen zu sinnhaften eigenen Klassen zusammengefasst werden, die dann ihre eigenen Attribute und Dienste besitzen.

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

Zeitbedarf: 18 Stunden

Zu entwickelnde Kompetenzen: Die Schüler*innen

analysieren und erläutern einfache Algorithmen und Programme (A),

entwerfen einfache Algorithmen und stellen sie umgangssprachlich und grafisch dar (M),

ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M),

modellieren Klassen mit ihren Attributen, ihren Methoden und Assoziationsbeziehungen (M),

ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare Datensammlungen zu (M),

ordnen Klassen, Attributen und Methoden ihren Sichtbarkeitsbereich zu (M),

modifizieren einfache Algorithmen und Programme (I),

implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I),

implementieren Algorithmen unter Verwendung von Variablen und Wertzuweisungen, Kontrollstrukturen sowie Methodenaufrufen (I),

implementieren einfache Algorithmen unter Beachtung der Syntax und Semantik einer Programmiersprache (I),

testen Programme schrittweise anhand von Beispielen (I),

interpretieren Fehlermeldungen und korrigieren den Quellcode (I).

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Beispiele, Medien, Materialien**
1. Bewegungsanimationen einfacher geometrischer Primitive in JPanels Kontinuierliche Verschiebung eines Primitivs mit Hilfe einer Schleife (While-Schleife) Mehrstufige Animationen mit mehreren sequenziellen Schleifen Berechnung von Abständen zwischen Primitiven mit Hilfsvariablen	*Beispiel:* Ballsimulation Die Schüler*innen realisieren eine Simulation eines Balls (Kreises), welcher entsprechend physikalischer Regeln von den Fensterrändern abprallt. *Materialien:* Java-API
2. Erstellen und Verwalten größerer Mengen geometrischer Primitive Erzeugung von Objekten mit Hilfe von Zählschleifen (FOR-Schleife) Verwaltung von Objekten in eindimensionalen Feldern (Arrays) Animation von Objekten, die in eindimensionalen Feldern (Arrays) verwaltet werden Vertiefung: Verschiedene Feldbeispiele	*Beispiel:* Wurm Die Schüler*innen realisieren eine Simulation eines „Wurms“, welcher aus mehreren Gliedern (Kreisen) besteht und nach bestimmten Regeln im JPanel „umherkriecht“. *Materialien:* Java-API
3. Modellierung und Animation unter Berücksichtigung von Benutzer-	*Beispiel:* Die Schüler*innen modellieren ein

Unterrichtssequenzen	Beispiele, Medien, Materialien
interaktionen Erarbeitung der Herausforderungen bei Benutzerinteraktion Realisierung von Zustandsvariablen, Änderung dieser Zustandsvariablen Thematisierung des Geheimnisprinzips und des Autonomitätsprinzips von Objekten	grafisches Objekt, welches sich entsprechend einer Zustandsvariable bewegt. Die Zustandsvariable kann durch Benutzerinteraktion geändert werden. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator – Java-API

Unterrichtsvorhaben EF-IV

Thema: Modellierung und Implementierung von komplexeren Klassen- und Objektbeziehungen

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

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

Dazu bedarf es zunächst einer präzisen Unterscheidung zwischen Objektreferenzen und Objekten, so dass klar wird, dass Dienste eines Objektes von unterschiedlichen Objekten über unterschiedliche Referenzen in Anspruch genommen werden können. Auch der Aufbau solcher Objektbeziehungen muss thematisiert werden. Des Weiteren wird das Prinzip der Vererbung im objektorientierten Sinne spätestens hier angesprochen. Dazu werden die wichtigsten Varianten der Vererbung anhand von verschiedenen Projekten vorgestellt. Zunächst wird die Vererbung als Spezialisierung im Sinne einer einfachen Erweiterung einer Oberklasse vorgestellt. Schließlich werden Interfaces eingeführt, um damit ereignisgesteuerte Benutzeroberflächen programmieren und vollständig verstehen zu können.

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

Zeitbedarf: 18 Stunden

Zu entwickelnde Kompetenzen: Die Schüler*innen

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 Datensammlungen zu (M),

ordnen Klassen, Attributen und Methoden ihren Sichtbarkeitsbereich zu (M),

modellieren Klassen unter Verwendung von Vererbung (M),

implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I),

testen Programme schrittweise anhand von Beispielen (I),

interpretieren Fehlermeldungen und korrigieren den Quellcode (I),

modifizieren einfache Algorithmen und Programme (I),

stellen Klassen, Assoziations- und Vererbungsbeziehungen in Diagrammen grafisch dar (D),

dokumentieren Klassen durch Beschreibung der Funktionalität der Methoden (D).

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Beispiele, Medien, Materialien
1. Vertiefung des Referenzbegriffs und Einführung des Prinzips der dynamischen Referenzierung	*Beispiel:* Herzschlag Mehrere Kreise befinden sich im Grafikbereich, aber nur einer pulsiert

Unterrichtssequenzen	Beispiele, Medien, Materialien
Steuerung einfacher grafischer Objekte über eine Referenz *aktuell*, die jeweils durch eine Selektion über GUI-Elemente auf ein neues Objekt gesetzt werden kann.	(wechselt seine Größe) in periodischen Abständen. Der Benutzer kann den pulsierenden Kreis z.B. durch einen Buttonclick auswählen
2. Entwicklung einer GUI welche mit unterschiedlichen Event-Listenern arbeitet Thematisierung unterschiedlicher Listener-Typen Behandlung des Observer-Entwurfsmusters (als Basis aller Event-Listener) Entwicklung einer GUI mit Buttons (ActionListener), Mausverfolgung (MouseListener), Fensterkontrolle (WindowListener) usw. Klärung des Unterschieds zwischen den beiden Ansätzen "ist ein JFrame" und "hat einen JFrame"	*Beispiel:* Mauszeigerhilfe Die Schüler*innen entwickeln ein Grafikpanel, in welchem an die Position des Mauszeigers ein Fadenkreuz gezeichnet wird.
3. Programmierung eines Spiels oder Simulation, in welchem eine Figur durch GUI-Elemente gesteuert werden kann. Analyse der benötigten Klassen und Elemente Implementierung Erweiterung auf mehrere Spieler / Akteure, wobei die Schüler*innen Vererbungsbeziehungen zwischen den Akteuren herstellen sollen	*Beispiel:* Schwarm In einem Grafikbereich befinden sich mehrere Objekte, welche sich stets in Richtung des Mauszeigers bewegen. *Beispiel:* Computerspiele Pong, Arkanoid, Wordle

Unterrichtsvorhaben EF-V

Thema: Such- und Sortieralgorithmen anhand kontextbezogener Beispiele

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

Vorhabenbezogene Konkretisierung: Dieses Unterrichtsvorhaben beschäftigt sich mit der Erarbeitung von Such- und Sortieralgorithmen. Der Schwerpunkt des Vorhabens liegt dabei auf den Algorithmen selbst und nicht auf deren Implementierung in einer Programmiersprache, auf die in diesem Vorhaben vollständig verzichtet werden soll. Evtl. sind in den vergangenen Unterrichtsvorhaben Datenfelder angesprochen worden, so dass ein Ankerpunkt für eine Programmierung vorhanden ist.

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

Daran anschließend werden die erarbeiteten Strategien systematisiert und im Pseudocode notiert. Die SchülerInnen sollen auf diese Weise das Sortieren durch das Vertauschen von Elementen (Dreieckstausch) am Beispiel des Sortierens durch direktes Auswählen und mindestens einem weiteren Sortieralgorithmus kennen lernen.

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

Zeitbedarf: 7 Stunden

Zu entwickelnde Kompetenzen: Die Schüler*innen

beurteilen die Effizienz von Algorithmen am Beispiel von Sortierverfahren hinsichtlich Zeit und Speicherplatzbedarf (A),

entwerfen einen weiteren Algorithmus zum Sortieren (M),

analysieren Such- und Sortieralgorithmen und wenden sie auf Beispiele an (D).

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Beispiele, Medien, Materialien
1. Explorative Erarbeitung eines	*Beispiel:* Sortieren mit Karten

Unterrichtssequenzen	Beispiele, Medien, Materialien
Sortierverfahrens Sortierprobleme im Kontext informatischer Systeme und im Alltag (z.B. Dateisortierung, Tabellenkalkulation, Telefonbuch, Bundesligatabelle, usw.) Vergleich und Tausch zweier Elemente als Grundlage eines Sortieralgorithmus Erarbeitung eines Sortieralgorithmus durch die SchülerInnen	Die Schüler*innen bekommen die Aufgabe, Karten mit Zahlen bedruckt aufsteigend zu sortieren. *Materialien:* Computer science unplugged – Sorting Algorithms, www.csunplugged.org/sorting-algorithms
2. Systematisierung von Algorithmen und Effizienzbetrachtungen Formulierung (falls selbst gefunden) oder Erläuterung von mehreren Algorithmen im Pseudocode (auf jeden Fall: Sortieren durch Vertauschen, Sortieren durch Auswählen) Anwendung von Sortieralgorithmen auf verschiedene Beispiele Bewertung von Algorithmen anhand der Anzahl der nötigen Vergleiche	*Beispiele:* Sortieren durch direkte Auswahl, Bubblesort, Quicksort, ein Negativbeispiel (Slowsort?) *Materialien:* Computer science unplugged – Sorting Algorithms, www.csunplugged.org/sorting-algorithms
3. Binäre Suche auf sortierten Daten Suchaufgaben im Alltag und im Kontext informatischer Systeme Effizienzbetrachtungen zur binären Suche	*Beispiel:* Simulationsspiel zur binären Suche in sortierten Feldern *Materialien:* Computer science unplugged – Searching Algorithms, www.csunplugged.org/searching-algorithms

Unterrichtsvorhaben EF-VI

Thema: Geschichte der digitalen Datenverarbeitung und Grundlagen des Datenschutzes

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

Vorhabenbezogene Konkretisierung: Das folgende Unterrichtsvorhaben stellt den Abschluss der Einführungsphase dar. Schüler*innen sollen selbstständig informatische Themenbereiche aus dem Kontext der Geschichte der Datenverarbeitung und insbesondere den daraus sich ergebenen Fragen des Datenschutzes bearbeiten. Diese Themenbereiche können in Kleingruppen bearbeitet und in Form von Plakatpräsentationen vorgestellt werden. Die Schüler*innen sollen dabei mit Unterstützung des Lehrenden selbstständige Recherchen zu ihren Themen anstellen und auch eine sinnvolle Eingrenzung ihres Themas vornehmen. Insbesondere sollen die Vorträge dazu genutzt werden, bereits bearbeitete Inhalte (Kodierung, ASCII, RGB, vgl. vergangene Unterrichtsvorhaben) zu vertiefen.

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

Zeitbedarf: 12 Stunden

Zu entwickelnde Kompetenzen: Die Schüler*innen

bewerten anhand von Fallbeispielen die Auswirkungen des Einsatzes von Informatiksystemen (A),

erläutern wesentliche Grundlagen der Geschichte der digitalen Datenverarbeitung (A),

stellen ganze Zahlen und Zeichen in Binärcodes dar (D),

interpretieren Binärcodes als Zahlen und Zeichen (D),

nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation. (K).

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Beispiele, Medien, Materialien
1. Selbstständige Erarbeitung von Themen durch die Schüler*innen Mögliche Themen: „Digitalisierung: vom Morsen zum modernen Digitalcomputer" „Kryptographie: von Caesar zur Enigma" „Stellenwertsysteme und wie man mit ihnen rechnet" „Kodieren von Texten und Bildern: ASCII, RGB und mehr" „Auswirkungen der Digitalisierung: Veränderungen der Arbeitswelt und Datenschutz"	*Beispiel:* Ausstellung zu informatischen Themen Die Schüler*innen bereiten eine Ausstellung zu informatischen Themen vor. Dazu werden Stellwände und Plakate vorbereitet, die ggf. auch außerhalb des Informatikunterrichts in der Schule ausgestellt werden können. *Materialien:* Internet, Schulbibliothek, öffentliche Bibliotheken, usw.
2. Vertiefung des Themas Datenschutz Erarbeitung grundlegender Begriffe des Datenschutzen Problematisierung und Anknüpfung an die Lebenswelt der Schüler*innen Diskussion und Bewertung von Fallbeispielen aus dem Themenbereich „Datenschutz"	*Beispiel:* Fallbeispiele Die Schüler*innen bearbeiten Fallbeispiele aus ihrer eigenen Erfahrungswelt oder der aktuellen Medienberichterstattung. *Materialien:* Materialblatt zum BDSG

2.3 Konkretisierte Unterrichtsvorhaben für das erste Jahr der Qualifikationsphase

Zu Beginn der Qualifikationsphase werden die Programmierkenntnisse in Java weiter vertieft und auf dynamische lineare Datenstrukturen ausgeweitet. Das Suchen und Sortieren solcher Datenmengen wird nun vollständig implementiert und um rekursive Verfahren ergänzt. Dabei findet auch ein detaillierter Efiizienzvergleich statt. Die in den Programmierprojekten verwendeten GUIs

können nach einer ersten Wiederholung hier weiter in den Hintergrund gedrängt und z.B. vorgegeben oder mit entsprechenden Werkzeugen erzeugt werden.

Zeitlich erstrecken sich diese Inhalte ungefähr bis zu den Osterferien gegen Ende des dritten Quartals. Den Abschluss des ersten Jahres der Qualifikationsphase bilden dann die Datenbanken sowie ein Exkurs zur Netzwerktechnik (Client-Server-Modell) und Netzwerksicherheit.

Unterrichtsvorhaben Q1-I

Thema: Objektorientierte Modellierung und Programmierung am Beispiel von Feldern und ihrer Sortierung

Leitfragen: Wie modelliert und implementiert man zu einer Problemstellung, die größere Mengen gleichartiger Daten beinhaltet, Java-Klassen inklusive ihrer Attribute, Methoden und Beziehungen? Wie kann man die Modellierung und die Funktionsweise der Anwendung grafisch darstellen?

Vorhabenbezogene Konkretisierung: Zu einer Problemstellung in einem Anwendungskontext soll eine Java-Anwendung entwickelt werden. Die Problemstellung soll so gewählt werden, dass für diese Anwendung die Verwendung einer abstrakten Oberklasse als Generalisierung verschiedener Unterklassen sinnvoll erscheint und eine Klasse durch eine Unterklasse spezialisiert werden kann.

Die Anwendung soll zudem ein Feld benutzen, dass für eine Suche bestimmter Objekte und eine Sortierung angesprochen werden kann. Dies kann ein einfaches Zahlenfeld sein oder ein Säulendiagramm modelliert durch ein Feld von spezialisierten Panelobjekten. Auch zweidimensionale Felder sollen Thema sein.

Die Schüler*innen erläutern und modifizieren sowie implementieren weitere Klassen und Methoden für eine entsprechende Anwendung (Steuerung, GUI, Datenmodell). Klassen und Beziehungen werden in einem Implementationsdiagramm dargestellt. Dabei werden Sichtbarkeitsbereiche zugeordnet und mindestens eine Klasse wird vollständig dokumentiert. Der Nachrichtenaustausch zwischen verschiedenen Objekten wird verdeutlicht, indem die Kommunikation zwischen zwei ausgewählten Objekten grafisch dargestellt wird (z.B. in einem Sequenzdiagramm). In diesem Zusammenhang wird das Nachrichtenkonzept der objektorientierten Programmierung vertieft.

In den implementierten Anwendungen werden Verfahren zur Suche von Informationen in einem Feld entwickelt und programmiert bzw. Analysiert und erläutert. Dabei wird neben iterativen auch ein rekursives Verfahren thematisiert und mindestens ein Verfahren selbst entwickelt und implementiert. Die verschiedenen Verfahren werden hinsichtlich Speicherbedarf und Zahl der Vergleichsoperationen miteinander verglichen.

Anschließend werden Sortierverfahren entwickelt und implementiert, hierbei soll auch ein rekursives Verfahren entwickelt werden. Die Implementationen von Quick- oder Mergesort sowie dem Sortieren durch Einfügen werden analysiert und erläutert. Falls diese Verfahren vorher schon entdeckt wurden, sollen sie hier wieder erkannt werden. Die rekursive Abarbeitung eines Methodenaufrufs von Quick- oder Mergesort wird grafisch dargestellt. Abschließend werden verschiedene Sortierverfahren hinsichtlich der Anzahl der benötigten Vergleichsoperationen und des Speicherbedarfs beurteilt.

Zeitbedarf: 20 Stunden

Zu entwickelnde Kompetenzen: Die Schüler*innen

analysieren und erläutern objektorientierte Modellierungen, Algorithmen und Programme (A),

beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A),

beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A)

modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M),

ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M),

entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien "Modularisierung" und "Teilen und Herrschen" (M),

implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I),

modifizieren Algorithmen und Programme (I),

implementieren und erläutern iterative und rekursive Such- und Sortierverfahren (I),

nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I),

wenden eine didaktisch orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I)

interpretieren Fehlermeldungen und korrigieren den Quellcode (I),

testen Programme systematisch anhand von Beispielen (I),

stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D),

dokumentieren Klassen (D),

stellen die Kommunikation zwischen Objekten grafisch dar (D),

stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D).

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Beispiele, Medien, Materialien**
1. Wiederholung und Erweiterung der objektorientierten Modellierung und Programmierung durch Analyse und Erweiterung eines kontextbezogenen Beispiels Analyse der Problemstellung und der Modellierung (Implementations-diagramm) Erweiterung der Modellierung im Implementationsdiagramm (Vererbung, abstrakte Klasse) Kommunikation zwischen mindestens zwei Objekten (Sequenzdiagramm) Dokumentation einer Klasse Implementierung der Anwendung oder Teilen davon	*Beispiel:* Wetthüpfen Mehrere Tiere oder Formen hüpfen in zufälliger Weite in dieselbe Richtung. *Beispiel:* Karteiverwaltung mit Adressdaten *Beispiel:* Bundesjugendspiele Mehrere Schüler*innen sollen Punktwerte in drei Disziplinen bekommen
2. Suchen von Daten in Arrays	*Beispiele:* Suchen von Adressen oder Namen (Karteiverwaltung, ...)

Unterrichtssequenzen	Beispiele, Medien, Materialien
Lineare Suche in Arrays (auch zweidimensional) Binäre Suche in Arrays als Beispiel für rekursives Problemlösen Untersuchung der beiden Suchverfahren hinsichtlich ihrer Effizienz (Laufzeitverhalten, Speicherbedarf)	
3. Sortieren in Arrays – Entwicklung und Implementierung von iterativen und rekursiven Sortierverfahren Implementierung eines einfachen Sortierverfahrens für ein Feld (z.B. Insertion-, Selectionsort) Entwicklung eines rekursiven Sortierverfahrens für ein Feld (z.B. Mergesort, Quicksort) Grafische Veranschaulichung der Sortierverfahren Untersuchung der Anzahl der Vergleichsoperationen und des Speicherbedarfs bei iterativen und rekursiven Sortierverfahren Beurteilung der Effizienz	*Beispiel:* Sortieren von Punktwerten, Namen (Bundesjugendspiele, ...)

Unterrichtsvorhaben Q1-II

Thema: Modellierung und Implementierung von Anwendungen mit dynamischen, linearen Listen und ihrer Sortierung

Leitfragen: Wie können beliebig viele linear angeordnete Daten in einem Anwendungskontext verwaltet, gesucht und sortiert werden?

Vorhabenbezogene Konkretisierung: Nach Analyse einer Problemstellung in einem geeigneten Anwendungskontext wird eine Klasse zur Verwaltung

von Daten in einer linearen, dynamischen Liste eingeführt und ausgebaut. Modellierungen werden dabei in Entwurfs- und Implementations-diagrammen dargestellt.

Die im Vorhaben Q1-I entwickelten Such- und Sortierverfahren werden hier auf dynamische Datenstrukturen übertragen. Dabei wird der Schwerpunkt auf die iterativen Verfahren gelegt und beim Vergleich verschiedener Verfahren eher der Blick darauf gelenkt, was in dynamischen Strukturen gut zu realisieren ist und was eher gar nicht.

Zeitbedarf: 20 Stunden

Zu entwickelnde Kompetenzen: Die Schüler*innen

erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A),

analysieren und erläutern Algorithmen und Programme (A),

beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A),

beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A)

ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nicht-lineare Datensammlungen zu (M),

ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M),

entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien "Modularisierung" und "Teilen und Herrschen" (M),

entwickeln iterative und rekursive Such- und Sortierverfahren (M),

modifizieren Algorithmen und Programme (I),

implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I),

nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I),

wenden eine didaktisch orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I)

interpretieren Fehlermeldungen und korrigieren den Quellcode (I),

testen Programme systematisch anhand von Beispielen (I),

stellen lineare und nicht-lineare Strukturen grafisch dar (D),

stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D).

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Beispiele, Medien, Materialien**
1. Die Datenstruktur lineare Liste im Anwendungskontext unter Nutzung der Klasse "List" Erarbeitung der Vorteile der Klasse "List" im Gegensatz zu den bereits bekannten linearen Strukturen Modellierung und Implementierung einer kontextbezogenen Anwendung unter Verwendung der Klasse "List"	*Beispiel:* Abfahrtslauf Skifahrer kommen nacheinander an und werden gemäß ihrer Zeit in eine Rangliste eingeordnet *Beispiel:* Skispringen Jeder Springer wird gemäß der Punktzahl eines ersten Sprungs in einer Rangliste einsortiert, die dann in umgekehrter Reihenfolge nochmals bepunktet wird. *Beispiel:* Karteiverwaltung komplexer *Beispiel:* Rangierbahnhof Mehrere Waggons mit Nummern müssen über drei Gleise in die richtige Reihenfolge rangiert werden.
2. Suchen von Daten in linearen Listen Lineare Suche Möglichkeiten einer binären Suche	
3. Sortieren in linearen Listen – Entwicklung und Implementierung von iterativen Sortierverfahren Implementierung eines einfachen Sortierverfahrens für eine lineare Liste (z.B. Insertion-, Selectionsort) Möglichkeiten eines rekursiven Sortierverfahrens für eine lineare Liste (z.B. Mergesort, Quicksort)	

Unterrichtsvorhaben Q1-III

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

Leitfragen: Wie können beliebig viele linear angeordnete Daten in einem Anwendungskontext verwaltet werden?

Vorhabenbezogene Konkretisierung: Nach Analyse einer Problemstellung in einem geeigneten Anwendungskontext, in dem Daten nach dem First-In-First-Out-Prinzip verwaltet werden, werden der Aufbau von Schlangen am Beispiel dargestellt und die Operationen der Klasse "Queue" erläutert. Anschließend werden für die Anwendung notwendige Klassen modelliert und implementiert. Eine Klasse für eine den Anforderungen der Anwendung entsprechende Oberfläche sowie die Klasse "Queue" wird dabei von der Lehrkraft vorgegeben.

Anschließend wird die Anwendung modifiziert, um den Umgang mit der Datenstruktur zu üben. Anhand einer Anwendung, in der Daten nach dem Last-In-First-Out-Prinzip verwaltet werden, werden Unterschiede zwischen den Datenstrukturen Schlange und Stapel erarbeitet. Modellierungen werden dabei in Entwurfs- und Implementationsdiagrammen dargestellt.

Zeitbedarf: 16 Stunden

Zu entwickelnde Kompetenzen: Die Schüler*innen

erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A),

analysieren und erläutern Algorithmen und Programme (A),

beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A),

ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M),

ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M),

modifizieren Algorithmen und Programme (I),

implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I),

nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I),

interpretieren Fehlermeldungen und korrigieren den Quellcode (I),

testen Programme systematisch anhand von Beispielen (I),

stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D).

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Beispiele, Medien, Materialien
1. Die Datenstruktur Schlange im Anwendungskontext unter Nutzung der Klasse "Queue" Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen Erarbeitung der Funktionalität der Klasse "Queue" Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse "Queue"	*Beispiel:* Patientenwarteschlange (jeder kennt seinen Nachfolger bzw. alternativ: seinen Vorgänger) Sobald ein Patient in einer Arztpraxis eintrifft, werden sein Name und seine Krankenkasse erfasst. Die Verwaltung der Patientenwarteschlange geschieht über eine Klasse, die hier als Wartezimmer bezeichnet wird. Wesentliche Operationen sind das „Hinzufügen" eines Patienten und das „Entfernen" eines Patienten, wenn er zur Behandlung gerufen wird. Die Simulationsanwendung stellt eine GUI zur Verfügung, legt ein Wartezimmer an und steuert die Abläufe. Wesentlicher Aspekt des Projektes ist die Modellierung des Wartezimmers mit Hilfe der Klasse "Queue". Anschließend wird der Funktionsumfang der Anwendung erweitert: Patienten können sich zusätzlich in die Warteschlange zum Blutdruckmessen einreihen. Objekte werden von zwei Schlangen verwaltet.

Unterrichtssequenzen	Beispiele, Medien, Materialien
2. Die Datenstruktur Stapel im Anwendungskontext unter Nutzung der Klasse "Stack" Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen Erarbeitung der Funktionalität der Klasse "Stack" Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse "Stack"	*Beispiel:* Heftstapel In einem Heftstapel soll das Heft einer Schülerin/eines Schülers gefunden werden. *Beispiel:* Kisten stapeln In einem Stapel nummerierter Kisten soll eine bestimmte Kiste gefunden und an einen Kunden geliefert werden. Dazu müssen Kisten auf verschiedene Stapel gestapelt und wieder zurückgestellt werden.

Unterrichtsvorhaben Q1-IV

Thema: Modellierung und Nutzung von relationalen Datenbanken in Anwendungskontexten

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

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

In anderen Anwendungskontexten müssen Datenbanken erst noch entwickelt werden, um Daten zu speichern und Informationen für die Beantwortung von möglicherweise auftretenden Fragen zur Verfügung zu stellen. Dafür ermitteln die SchülerInnen in den Anwendungssituationen Entitäten, zugehörige Attribute, Relationen und Kardinalitäten und stellen diese in Entity-Relationship-Modellen dar. Entity-Relationship-Modelle werden interpretiert und erläutert, modifiziert und in Datenbankschemata

überführt. Mit Hilfe von SQL-Anweisungen können anschließend im Kontext relevante Informationen aus der Datenbank extrahiert werden.

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

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

Schließlich soll auch eine Implementation in Java erfolgen, die die Abiturklassen *DatabaseConnector* und *QueryResult* verwendet. Hier empfiehlt sich eine SQLite-Datenbank, die keinen eigenen Server benötigt. Aber auch SQL-Server mittels XAMPP oder tatsächlich im Netz betriebene Server sind möglich.

Zeitbedarf: 20 Stunden

Zu entwickelnde Kompetenzen: Die Schüler*innen

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),

implementieren Datenbankabfragen innerhalb eines Java-Programms mit Hilfe geeigneter Konnektor-Software (M, I).

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Beispiele, Medien, Materialien**
1. Nutzung von relationalen Datenbanken Entwicklung von Fragestellungen zu einer vorhandenen Datenbank Analyse der Struktur der vorgegebenen Datenbank und Erarbeitung der Begriffe Tabelle, Attribut, Datensatz, Datentyp, Primärschlüssel, Fremdschlüssel, Datenbankschema Analyse vorgegebener SQL-Abfragen und Erarbeitung der Sprachelemente von SQL (SELECT (DISTINCT) … FROM ... WHERE (NOT) ... AND/OR ...) auf einer Tabelle Bedingungen mit Vergleichsoperatoren: =, <>, >, <, >=, <=, LIKE, BETWEEN, IN, IS NULL Analyse und Erarbeitung von SQL-Abfragen auf einer und mehrerer Tabelle zur Beantwortung der Fragestellungen (Kreuzprodukt, Einschränkung auf sinnvolle	*Beispiel:* Videocenter VideoCenter ist die Simulation einer Online-Videothek für den Informatik-Unterricht mit Webfrontends zur Verwaltung der Kunden, der Videos und der Ausleihe. Außerdem ist es möglich direkt SQL-Abfragen einzugeben. Es ist auch möglich, die Datenbank herunter zu laden und lokal zu installieren. Unter http://dokumentation.videocenter.schule.de/old/video/index.html (abgerufen: 30. 03. 2014) findet man den Link zu dem VideoCenter-System sowie nähere Informationen. *Beispiel:* Schulbuchausleihe Unter www.brd.nrw.de/lerntreffs/informatik/structure/material/sek2/

Unterrichtssequenzen	Beispiele, Medien, Materialien
Datensätze, Sprachelemente JOIN und UNION sind nicht notwendig) Vertiefung an einem weiteren Datenbankbeispiel (auch mit Funktionen COUNT, MAX, MIN, SUM und arithmetische Operatoren +, -, *, / möglich)	datenbanken.php (abgerufen: 30. 03. 2014) wird eine Datenbank zur Verfügung gestellt, die Daten einer Schulbuch-Ausleihe enthält (über 1000 Entleiher, 200 Bücher mit mehreren tausend Exemplaren und vielen Ausleihvorgänge). Die Datenbank kann in OpenOffice eingebunden werden.
2. Modellierung von relationalen Datenbanken Ermittlung von Entitäten, zugehörigen Attributen, Relationen und Kardinalitäten in Anwendungssituationen und Modellierung eines Datenbankentwurfs in Form eines Entity-Relationship-Diagramms Erläuterung und Modifizierung einer Datenbankmodellierung Modellierung eines relationalen Datenbankschematas zu einem Entity-Relationship-Diagramm inklusive der Bestimmung von Primär- und Sekundärschlüsseln Untersuchung einer Datenbank hinsichtlich Konsistenz und Redundanz in einer Anwendungssituation Überprüfung von Datenbankschemata hinsichtlich der 1. bis 3. Normalform und Normalisierung (um Redundanzen zu vermeiden und Konsistenz zu gewährleisten)	*Beispiel:* Fahrradverleih Der Fahrradverleih BTR (BikesToRent) verleiht unterschiedliche Typen von Fahrrädern diverser Firmen an seine Kunden. Die Kunden sind bei BTR registriert (Name, Adresse, Telefon). BTR kennt von den Fahrradfirmen den Namen und die Telefonnummer. Kunden von BTR können CityBikes, Treckingräder und Mountainbikes ausleihen. *Beispiel:* Reederei Die Datenverwaltung einer Reederei soll in einem Datenbanksystem umgesetzt werden. Ausgehend von der Modellierung soll mit Hilfe eines ER-Modells und eines Datenbankschemas dieser erste Entwurf normalisiert und in einem Datenbanksystem umgesetzt werden. Es schließen sich diverse SQL-Abfragen an, wobei auf die Relationenalgebra eingegangen wird. *Beispiel:* Buchungssystem

Unterrichtssequenzen	Beispiele, Medien, Materialien
	In dem Online-Buchungssystem einer Schule können die Lehrer Medienräume, Beamer, Laptops, Kameras, usw. für einen bestimmten Zeitpunkt buchen, der durch Datum und die Schulstunde festgelegt ist. Dazu ist die Datenbank zu modellieren, ggf. zu normalisieren und im Datenbanksystem umzusetzen. Weiter sollen sinnvolle Abfragen entwickelt werden. Unter http://mrbs.sourceforge.net (abgerufen: 30.03. 2014) findet man ein freies Online-Buchungssystem inklusive Demo, an Hand derer man erläutern kann, worum es in dem Projekt geht. *Beispiel:* Schulverwaltung In einer Software werden die Schulhalbjahre, Jahrgangsstufen, Kurse, Klassen, Schüler, Lehrer und Noten einer Schule verwaltet. Man kann dann ablesen, dass z.B. Schüler X von Lehrer Y im 2. Halbjahr des Schuljahrs 2011/2012 in der Jahrgangsstufe Q1 im Fach Informatik die Note „sehr gut" erhalten hat. Dazu ist die Datenbank zu modellieren, ggf. zu normalisieren und im Datenbanksystem umzusetzen. Weiter sollen sinnvolle Abfragen entwickelt werden und das Thema Datenschutz besprochen werden.
3. Implementierung einer Datenbank-anwendung in Java mit Hilfe der Abiturklassen *DatabaseConnector* und *QueryResult*	SQLite-Treiber und -Bibliotheken finden sich im Netz

Unterrichtsvorhaben Q1-V

Thema: Sicherheit und Datenschutz in Netzstrukturen

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

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

Zeitbedarf: 10 Stunden

Zu entwickelnde Kompetenzen: Die Schüler*innen

beschreiben und erläutern Topologien, die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken (A),

analysieren und erläutern Eigenschaften und Einsatzbereiche symmetrischer und asymmetrischer Verschlüsselungsverfahren (A),

untersuchen und bewerten anhand von Fallbeispielen die Auswirkungen des Einsatzes von Informatiksystemen, die Sicherheit von Informatiksystemen sowie die Einhaltung der Datenschutzbestimmungen und des Urheberrechts (A),

untersuchen und bewerten Problemlagen, die sich aus dem Einsatz von Informatiksystemen ergeben, hinsichtlich rechtlicher Vorgaben, ethischer Aspekte und gesellschaftlicher Werte unter Berücksichtigung unterschiedlicher Interessenlagen (A),

nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zum Erschließen, zur Aufbereitung und Präsentation fachlicher Inhalte (D).

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Beispiele, Medien, Materialien
1. Daten in Netzwerken und Sicherheitsaspekte in Netzen sowie beim Zugriff auf Datenbanken Beschreibung eines Datenbankzugriffs im Netz anhand eines Anwendungskontextes und einer Client-Server-Struktur zur Klärung der Funktionsweise eines Datenbankzugriffs Netztopologien als Grundlage von Client-Server-Strukturen und OSI/ISO-Schichtenmodell als Beispiel für eine Paketübermittlung in einem Netz Vertraulichkeit, Integrität, Authentizität in Netzwerken sowie symmetrische und asymmetrische kryptografische Verfahren (Cäsar-, Vigenère-, RSA-Verfahren) als Methoden Daten im Netz verschlüsselt zu übertragen	*Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.5 – Zugriff auf Daten in Netzwerken
2. Fallbeispiele zur Datenschutzproblematik und zum Urheberrecht, Grundprinzipien des Datenschutzes	*Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.5 – Datenschutz beim Videocenter, Materialblatt Datenschutzgesetz

2.4 Konkretisierte Unterrichtsvorhaben für das zweite Jahr der Qualifikationsphase

Eine Wiederholungsphase wird zu Beginn des ersten Unterrichtsvorhabens eingefügt als Vertiefung des Vorhabens Q1-III (lineare dynamische Datenstrukturen).

Unterrichtsvorhaben Q2-I

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

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

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

Nicht zuletzt bei der Entscheidung, wie ein neues Element vernünftig eingefügt werden soll, wird der allgemeine Binärbaum zugunsten des binären Suchbaumes nicht weiter verfolgt. Zu weiteren Problemstellungen in entsprechenden Anwendungskontexten werden die Operationen der Datenstruktur Suchbaum thematisiert und die Klasse `BinarySearchTree` (der Materialien für das Zentralabitur in NRW) modelliert und implementiert. Auch in diesem Kontext werden grafische Darstellungen der Bäume verwendet. Die Verwendung von binären Bäumen und Suchbäumen wird anhand weiterer Problemstellungen oder anderen Kontexten weiter geübt.

Zeitbedarf: 24 Stunden

Zu entwickelnde Kompetenzen: Die Schüler*innen

erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A),

analysieren und erläutern Algorithmen und Programme (A),

beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A),

ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M),

ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M),

modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M),

verwenden bei der Modellierung geeigneter Problemstellungen die Möglichkeiten der Polymorphie (M),

entwickeln iterative und rekursive Algorithmen unter Nutzung der Konstruktionsstrategien „Modularisierung" und „Teilen und Herrschen" (M),

implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I),

modifizieren Algorithmen und Programme (I),

nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I),

interpretieren Fehlermeldungen und korrigieren den Quellcode (I),

testen Programme systematisch anhand von Beispielen (I),

stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D),

stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D).

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Beispiele, Medien, Materialien**
1. Wiederholung linearer Datenstrukturen	*Beispiele:* Ähnlicher Kontext zu den Anwendungen aus Q1, Operationen Einfügen und Löschen in linearer Liste, evtl. haben die Einträge in der Liste (Items) einen Verweis, z.B. auf eine weitere Liste (Dictionary)

Unterrichtssequenzen	**Beispiele, Medien, Materialien**
2. Die Datenstruktur Binärbaum im Anwendungskontext Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen im Anwendungskontext Modellierung eines Entwurfs-diagramms und Entwicklung eines Implementationsdiagramms Erarbeitung der Klasse `BinaryTree` und beispielhafte Anwendung der Operationen Implementierung der Anwendung oder von Teilen der Anwendung Funktionsweise der Traversierung eines Binärbaums im Pre-, In- und Postorderdurchlauf	*Beispiele:* Termbaum: Der Aufbau von Termen wird mit Hilfe von binären Baumstrukturen verdeutlicht. Ahnenbaum: Die binäre Baumstruktur ergibt sich daraus, dass jede Person genau einen Vater und eine Mutter hat. Entscheidungsbäume: Um eine Entscheidung zu treffen, werden mehrere Fragen mit ja oder nein beantwortet. Die Fragen, die möglich sind, wenn die Antwort auf eine Frage mit „ja" beantwortet wird, befinden sich im linken Teilbaum, die Fragen, die möglich sind, wenn die Antwort „nein" lautet, stehen im rechten Teilbaum. Codierungsbäume für das Morse-Alphabet: Die Codierungen von Buchstaben als Folge von Punkten und Strichen können in einem Binärbaum dargestellt werden, so dass ein Übergang zum linken Teilbaum einem Punkt und ein Übergang zum rechten Teilbaum einem Strich entspricht. Wenn man im Gesamtbaum startet und durch Übergänge zu linken oder rechten Teilbäumen einen Pfad zum gewünschten Buchstaben sucht, erhält man die Morsecodierung des Buchstabens.
3. Die Datenstruktur binärer Suchbaum im Anwendungskontext Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen	*Beispiel*: Informatikerbaum als Suchbaum: In einem binären Suchbaum werden die Namen und die Geburtsdaten von Informatikern lexikographisch geordnet abgespeichert. Alle Namen,

Unterrichtssequenzen	Beispiele, Medien, Materialien
Modellierung eines Entwurfs-diagramms und Entwicklung eines Implementationsdiagramm, Grafische Darstellung eines binären Suchbaums und Erarbeitung der Struktureigenschaften Erarbeitung der Klasse `BinarySearchTree` und Einführung des Interface `Item` zur Realisierung einer geeigneten Ordnungsrelation Implementierung der Anwendung oder von Teilen der Anwendung inklusive einer sortierten Ausgabe des Baums	die nach dieser Ordnung vor dem Namen im aktuellen Teilbaum stehen, sind in dessen linkemTeilbaum, alle die nach dem Namen im aktuellen Teilbaum stehen, sind in dessen rechtem Teilbaum. (Dies gilt für alle Teilbäume.) Folgende Funktionalitäten werden benötigt: Einfügen der Informatiker-Daten in den Baum, Suchen nach einem Informatiker über den Schlüssel Name, Ausgabe des kompletten Datenbestands in nach Namen sortierter Reihenfolge

Unterrichtsvorhaben Q2-II

Thema: Endliche Automaten und formale Sprachen

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

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

Anhand kontextbezogener Beispiele werden Grammatiken regulärer Sprachen entwickelt, untersucht und modifiziert. Der Zusammenhang zwischen regulären Grammatiken und endlichen Automaten wird verdeutlicht durch die Entwicklung von allgemeinen Verfahren zur Erstellung

einer regulären Grammatik für die Sprache eines gegebenen endlichen Automaten bzw. zur Entwicklung eines endlichen Automaten, der genau die Sprache einer gegebenen regulären Grammatik akzeptiert.

Auch andere Grammatiken werden untersucht, entwickelt oder modifiziert. An einem Beispiel werden die Grenzen endlicher Automaten ausgelotet. Als Ziel kann der deterministische Kellerautomat bzw. die deterministisch kontextfreien Sprachen zur Beschreibung der Klasse der Programmiersprachen stehen.

Zeitbedarf: 20 Stunden

Zu entwickelnde Kompetenzen: Die Schüler*innen

analysieren und erläutern die Eigenschaften endlicher Automaten einschließlich ihres Verhaltens auf bestimmte Eingaben (A),

analysieren und erläutern Grammatiken regulärer Sprachen (A),

zeigen die Grenzen endlicher Automaten und regulärer Grammatiken im Anwendungszusammenhang auf (A),

ermitteln die formale Sprache, die durch eine Grammatik erzeugt wird (A),

entwickeln und modifizieren zu einer Problemstellung endliche Automaten (M),

entwickeln und modifizieren zu einer Problemstellung endliche Automaten (M),

entwickeln zur akzeptierten Sprache eines Automaten die zugehörige Grammatik (M),

entwickeln zur Grammatik einer regulären Sprache einen zugehörigen endlichen Automaten (M),

modifizieren Grammatiken regulärer Sprachen (M),

entwickeln zu einer regulären Sprache eine Grammatik, die die Sprache erzeugt (M),

stellen endliche Automaten in Tabellen oder Graphen dar und überführen sie in die jeweilsandere Darstellungsform (D),

ermitteln die Sprache, die ein endlicher Automat akzeptiert (D).

beschreiben an Beispielen den Zusammenhang zwischen Automaten und Grammatiken (D).

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Beispiele, Medien, Materialien
1. Endliche Automaten Vom Automaten in den SchülerInnen bekannten Kontexten zur formalen Beschreibung eines endlichen Automaten Untersuchung, Darstellung und Entwicklung endlicher Automaten ε-Übergänge und nichtdeterministische endliche Automaten	*Beispiele:* Cola-Automat, Geldspielautomat, Roboter, Zustandsänderung eines Objekts „Auto", Akzeptor für bestimmte Zahlen, Akzeptor für Teilwörter in längeren Zeichenketten, Akzeptor für Terme
2. Untersuchung und Entwicklung von Grammatiken regulärer Sprachen Erarbeitung der formalen Darstellung regulärer Grammatiken Untersuchung, Modifikation und Entwicklung von Grammatiken inkl. ε-Produktionen Entwicklung von endlichen Automaten zum Erkennen regulärer Sprachen die durch Grammatiken gegeben werden Entwicklung regulärer Grammatiken zu endlichen Automaten	*Beispiele:* reguläre Grammatik für Wörter mit ungerader Parität, Grammatik für Wörter, die bestimmte Zahlen repräsentieren, Satzgliederungsgrammatik
3. Grenzen endlicher Automaten	*Beispiel*: Klammerausdrücke, a^nb^n im Vergleich zu $(ab)^n$

Unterrichtsvorhaben Q2-III

Thema: Prinzipielle Arbeitsweise eines Computers und Grenzen der Automatisierbarkeit

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

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

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

Zeitbedarf: 12 Stunden

Zu entwickelnde Kompetenzen: Die Schüler*innen

erläutern die Ausführung eines einfachen maschinennahen Programms sowie die Datenspeicherung auf einer „Von-Neumann-Architektur" (A),

untersuchen und beurteilen Grenzen des Problemlösens mit Informatiksystemen (A).

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Beispiele, Medien, Materialien
1. Von-Neumann-Architektur und die Ausführung maschinennaher Programme Prinzipieller Aufbau einer von Neumann-Architektur mit CPU, Rechenwerk, Steuerwerk, Register und Hauptspeicher Einige maschinennahe Befehle und ihre Repräsentation in einem Binär-Code, der in einem Register	*Beispiele:* Addition von 4 zu einer eingegeben Zahl mit einem Rechnermodell

Unterrichtssequenzen	Beispiele, Medien, Materialien
gespeichert werden kann Analyse und Erläuterung der Funktionsweise eines einfachen maschinennahen Programms	
2. Grenzen der Automatisierbarkeit Vorstellung des Halteproblems Unlösbarkeit des Halteproblems Beurteilung des Einsatzes von Informatiksystemen hinsichtlich prinzipieller Möglichkeiten und prinzipieller Grenzen	

Unterrichtsvorhaben Q2-IV

Thema: Wiederholung und Vertiefung ausgewählter Kompetenzen und Inhalte des ersten Jahrs der Qualifikationsphase und Vorbereitung auf die Abiturprüfungen

3 Grundsätze der Leistungsbewertung

Die Grundlage der Leistungsbewertung im Fach Informatik bilden §48 SchulG, §13 APO-GOSt sowie Kapitel 3 des Kernlehrplans Informatik. Des Weiteren gilt für uns das Dispositiv des Lernens: Lernen und Denken des Einzelnen fördern.

Unsere Lernerfolgsüberprüfungen sollen darauf ausgerichtet sein,

- Schüler*innen Hilfe und Diagnostik zur Förderung ihrer Kompetenzen zu geben,
- Kompetenzen zu vertiefen und zu erweitern,
- den Lernprozess zu evaluieren,
- den Schüler*innen Rückmeldung zu ihrem Kenntnis- und Kompetenzstand zu geben.

Dabei bezieht sich die Leistungsbewertung insgesamt auf die im Zusammenhang mit dem Unterricht erworbenen Kompetenzen und nutzt unterschiedliche Formen der Lernerfolgsüberprüfung. Die Kriterien für die Notengebung sowie der Beurteilung in den Bereichen sonstige Mitarbeit und Klausuren werden den Schüler*innen zu Beginn des Schuljahres transparent gemacht.

3.1 Klausuren

Die folgende grobe Zuordnung der Notenstufen wird bei der Bewertung von Klausuren zugrunde gelegt. Eine punktgenaue feine Zuordnung obliegt der Lehrkraft:

Erreichter Prozentsatz	**Punkte**	**Note**
ab ca. 70%	10 – 15	gut bis sehr gut
ab ca. 40%	4 – 9	ausreichend bis befriedigend
ab 20%	1 – 3	mangelhaft
unter 20%	0	ungenügend

In der Einführungsphase wird eine Klausur pro Halbjahr geschrieben, in der Qualifikationsphase eine pro Quartal. Sie ist zweistündig (95 Minuten), im zweiten Jahr der Qualifikationsphase dreistündig (160 Minuten) und kann praktische Anteile enthalten, die am PC bearbeitet werden.

3.2 Sonstige Mitarbeit

Qualität, Kontinuität, Intensität, Selbstständigkeit und Lernfortschritt in den Unterrichtsstunden sind entscheidende Grundlagen der Beurteilung im Bereich „Sonstige Mitarbeit". Pro Quartal wird eine eigene Note ermittelt.

Mögliche Arbeitsformen der „Sonstigen Mitarbeit" sind:

- Beiträge im Unterrichtsgespräch,
- Erarbeitung und Implementation von Algorithmen,
- Umgang mit den verfügbaren Systemen,
- Hausaufgaben,

- Referate und Präsentationen,
- Protokolle und Mitschriften,
- Mitarbeit im Team,
- Beiträge zu Projektarbeiten,
- Beiträge zu Gruppenarbeiten,
- schriftliche Übungen

Der Einsatz der jeweiligen Arbeitsformen ergibt sich aus dem Unterricht und der Lerngruppe. Insofern kann eine generelle Festlegung der Bedeutung der verschiedenen Arbeitsformen für die Bildung der Kursabschnittsnote nicht vorgenommen werden. Lediglich die Projektarbeit kann im Informatikunterricht etwas höher angesiedelt werden, da viele der behandelten Themen zu Software-Projekten führen, die die SchülerInnen allein, zu Zweit oder auch in größeren Teams realisieren sollen.

In welchem Umfang einzelne Referate, Präsentationen oder schriftliche Übungen in die Gesamtnote zur sonstigen Mitarbeit einfließen, obliegt der Beurteilung und dem pädagogischen Ermessen der Lehrkraft. Eine Maximalgrenze von schriftlichen Übungen gibt es im Fach Informatik nicht.