EUREGIO-GESAMTSCHULE

Schulinterner Lehrplan
für die Oberstufe
im Fach

Informatik

Stand: August 2023

Inhalt

	Seite
1. Die Fachgruppe Informatik an der Euregio-Gesamtschule	**3**
2. Entscheidungen zum Unterricht	**4**
2.1.1 Übersichtsraster Unterrichtsvorhaben Einführungsphase	5
2.1.2 Übersichtsraster Unterrichtsvorhaben Qualifikationsphase GK	18
2.1.3 Übersichtsraster Unterrichtsvorhaben Qualifikationsphase LK	36
2.2 Grundsätze der fachmethodischen und fachdidaktischen Arbeit	64
2.3 Grundsätze der Leistungsbewertung und Leistungsrückmeldung	66
2.4 Lehr- und Lernmittel	68
3 Entscheidungen zu fach- und unterrichtsübergreifenden Fragen	**69**
4 Qualitätssicherung und Evaluation	**70**

1 Die Fachgruppe Informatik an der Euregio-Gesamtschule

Die Euregio-Gesamtschule liegt im Ortsteil Epe der Stadt Gronau. Sie ist die einzige weiterführende Schule in diesem Ortsteil. Die Schülerschaft kommt sowohl aus Gronau als auch aus Epe und ist daher heterogen, was den sozialen und ethnischen Hintergrund betrifft.

Die Euregio-Gesamtschule ist eine Ganztagsschule, die in der Sekundarstufe I vierzügig unterrichtet wird. Die gymnasiale Oberstufe führt den Unterricht der Gesamtschule aus der Sekundarstufe I fort. In die Einführungsphase der Sekundarstufe II gehen durchschnittlich 45 Schülerinnen und Schüler über.

In der Regel werden in der Einführungsphase ein oder zwei Grundkurse eingerichtet, aus denen sich für die Qualifikationsphase Leistungs- und Grundkurse entwickeln. Dabei werden die Kurse meist als Kooperationskurse mit dem Gymnasium Gronau angelegt. Die Grundlagen für die Informatikkenntnisse werden in der Unter- und Mittelstufe gelegt. In Jahrgang 5 ist das Tastenschreiben ein eigenes Fach, welches halbjährlich unterrichtet wird. In Jahrgang 6 wird das Fach Informatik unterrichtet, wie es im Kernlehrplan für die Sekundarstufe I ausgewiesen wird. Vertiefend können Schülerinnen und Schüler das Fach Informatik ab Klasse 8 als Teil des EGS-Unterrichts (Ergänzungsstunden) wählen. Dieser Unterricht ist allerdings keine Voraussetzung für die Wahl des Faches Informatik in der Oberstufe.

Den im Kernlehrplan ausgewiesenen Zielen allgemeine Prinzipien und Methoden zur Erforschung komplexer Phänomene bereitzustellen, trägt die Fachschaft besonderer Rechnung, indem sie Aufgaben mit lebensweltlichem Bezug als Ausgangspunkt wählt. Denn prozessorgesteuerte Geräte, Softwareprodukte und durch deren Einsatz bestimmte Verfahrensweisen und Prozesse beeinflussen und verändern unser Leben mit hoher Dynamik. Die Schülerinnen und Schüler erweitern in der aktiven Auseinandersetzung mit komplexen Problemstellungen Kompetenzen, die sie zum selbstständigen informatischen Problemlösen befähigen.[1]

2 Entscheidungen zum Unterricht

2.1 Unterrichtsvorhaben

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

2.1.1 Übersichtsraster Unterrichtsvorhaben Einführungsphase

Unterrichtsvorhaben EF-I

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

Zeitbedarf: ca. 3 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
Prinzipieller Aufbau und Arbeitsweise eines Rechners (a) *EVA Prinzip* (b) *von-Neumann-Architektur* Speichern von Daten mit informatischen Systemen am Beispiel der Schulrechner → Netzwerk	Die Schülerinnen und Schüler • 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)	Bestandteile eines Rechners identifizieren und kategorisieren Einführung in Nutzung des Schulnetzwerkes

Unterrichtsvorhaben EF-II

Thema: Grundlagen der objektorientierten Analyse, Modellierung und Implementierung anhand von statischen Grafikszenen

Zeitbedarf: ca. 12 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Identifikation von Objekten *(a) Am Beispiel eines lebensweltnahen Beispiels werden Objekte im Sinne der Objektorientierten Modellierung eingeführt.* *(b) Manche Objekte sind prinzipiell typgleich und werden so zu einer Objektsorte bzw. Objektklasse zusammengefasst.*	Die Schülerinnen und Schüler • 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), • implementieren einfache Algorithmen unter Beachtung der Syntax und Semantik einer Programmiersprache (I), • stellen den Zustand eines Objekts dar (D).	Material: z.B. Werkzeugkoffer Schülerinnen und Schüler betrachten einen Werkzeugkoffer als Menge gleichartiger Objekte, die in einer Klasse mit Attributen und Methoden zusammengefasst werden können.
2. Analyse von Klassen didaktischer Lernumgebungen *(a) Objektorientierte Programmierung als modularisiertes Vorgehen* *(b) Teilanalyse der Klassen der didaktischen Lernumgebungen SAS*		Materialien: Dokumentation der didaktischen Bibliothek SAS
3. Implementierung dreidimensionaler, statischer Szenen (a) Grundaufbau einer Java-Klasse (b) Konzeption einer Szene mit Kamera, Licht und sichtbaren Objekten (c) Deklaration und Initialisierung von Objekten		Projekt: z.B. Skulpturengarten Schülerinnen und Schüler erstellen ein Programm, das mit Hilfe von geometrischen Objekten der SAS-Umgebung einen Skulpturengarten auf den Bildschirm bringt. Projekt: z.B. Olympische Ringe

(d) Methodenaufrufe mit Parameterübergabe zur Manipulation von Objekteigen-schaften (z.B. Farbe, Position, Drehung)		Die Schülerinnen und Schüler bilden das Emblem der olympischen Spiele mit Hilfe von SAS-Objekten nach.

Unterrichtsvorhaben EF-III

Thema: Grundlagen der objektorientierten Programmierung und algorithmischer Grundstrukturen in Java anhand von einfachen Animationen

Zeitbedarf: ca. 30 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Bewegungsanimationen am Beispiel einfacher grafischer Objekte (a) Kontinuierliche Verschiebung eines Objekts mit Hilfe einer Schleife (While-Schleife) (b) Tastaturabfrage zur Realisierung einer Schleifenbedingung für eine Animationsschleife (c) Mehrstufige Animationen mit mehreren sequenziellen Schleifen (d) Berechnung von Abständen zwischen Objekten mit Hilfsvariablen (e) Meldungen zur Kollision zweier Objekte mit Hilfe von Abstandsberechnungen und Verzweigungen (IF-Anweisungen)	Die Schülerinnen und Schüler • 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	Projekt: z.B. „Dart“-Wurfspiel Die Schülerinnen und Schüler realisieren mit Objekten der SAS-Umgebung ein Spiel, bei dem ein Pfeil über den Bildschirm bewegt und auf eine runde Zielscheibe geworfen werden soll. Alternativ: z.B. rollende Kugeln
2. Erstellen und Verwalten größerer Mengen einfacher grafischer Objekte (Objekte) (a) Erzeugung von Objekten mit Hilfe von Zählschleifen (FOR-Schleife) (b) Verwaltung von Objekten in		Projekt: z.B. Hubschrauberlandeplatz Die Schülerinnen und Schüler realisieren einen runden Hubschrauberlandeplatz und eine Reihe von Landemarkierungen, die in einem Feld verwaltet werden. Mit Hilfe der

eindimensionalen Feldern (Arrays) (c) Animation von Objekten, die in eindimensionalen Feldern (Arrays) verwaltet werden (d) Vertiefung: Verschiedene Feldbeispiele	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).	Landemarkierungen werden verschiedene Lauflichter realisiert. Projekt: z.B. Schachbrett Die Schülerinnen und Schüler realisieren mit Hilfe mehrerer Quader ein Schachbrett. Beispiel: Magischer Würfel Die Schülerinnen und Schüler erstellen einen großen Würfel, der aus mehreren kleineren, farbigen Würfeln besteht.
3. Modellierung und Animation komplexerer grafisch repräsentierbarer Objekte (a) Modellierung eines Simulationsprogramms mit eigenen Klassen, die sich selbst mit Hilfe von einfachen Objekten zeigen mit Hilfe eines Implementationsdiagramms (b) Implementierung eigener Methoden mit und ohne Parameterübergabe (c) Realisierung von Zustandsvariablen (d) Thematisierung des Geheimnisprinzips und des Autonomitätsprinzips von Objekten (e) Animation mit Hilfe des Aufrufs von selbstimplementierten Methoden		Projekt: z.B. Kerzensimulation Die Schülerinnen und Schüler modellieren und erstellen eine Klasse, mit deren Hilfe Kerzen simuliert werden können. Eine Kerze kann angezündet und gelöscht werden. Abgesehen davon brennen Kerzen abhängig von ihrer Dicke unterschiedlich schnell ab. Projekt: z.B. Uhren Die Schülerinnen und Schüler erstellen eine Simulation mehrerer Uhren für verschiedene Zeitzonen. Projekt: z.B. Ampeln Die Schülerinnen und Schüler erstellen eine Ampelkreuzung mit mehreren Ampelanlagen an einem Bahnübergang.

(f) Vertiefung: Weitere Projekte		

Unterrichtsvorhaben EF-IV

Thema: Modellierung und Implementierung von Klassen- und Objektbeziehungen anhand von grafischen Spielen und Simulationen

Zeitbedarf: ca. 30 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Vertiefung des Referenzbegriffs und Einführung des Prinzips der dynamischen Referenzierung (a) Einführung der Objektselektion mit der Maus (b) Einführung der Oberklasse aller sichtbaren Objekte in SAS (c) Steuerung einfacher grafischer Objekte über eine Referenz aktuell, die jeweils durch eine Klickselektion mit der Maus auf ein neues Objekt gesetzt werden kann.	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 Datensammlungen zu (M), • ordnen Klassen, Attributen und Methoden ihren Sichtbarkeitsbereich zu (M), • modellieren Klassen unter Verwendung von	Projekt: z.B. Seifenblasen Die Schülerinnen und Schüler entwickeln ein Spiel, bei dem Seifenblasen über den Bildschirm schweben und durch anklicken mit der Maus zum Zerplatzen gebracht werden können. Projekt: z.B. Sonnensystem Die Schülerinnen und Schüler entwickeln eine Simulation des Sonnensystems bei der Daten zum angeklickten Planeten ausgegeben werden.
2. Entwicklung eines Spiels mit der Notwendigkeit von Kollisionskontrollen zwischen zwei oder mehr grafischen Objekten (a) Modellierung des Spiels ohne Berücksichtigung der Kollision mit Hilfe eines Implementationsdiagramms (b) Dokumentation der Klassen des Projekts (c) Implementierung eines Prototypen ohne		Projekt: z.B. Ufospiel Die Schülerinnen und Schüler entwickeln die Simulation eines Ufos, das Asteroiden ausweichen soll mit denen eine Kollision möglich ist. Projekt: z.B. Billardkugeln

Kollision (d) Ergänzung einer Kollisionsabfrage durch zusätzliche Assoziationsbeziehungen in Diagramm, Dokumentation und Quellcode (e) Verallgemeinerung der neuen Verwendung von Objektreferenzen	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).	Die Schülerinnen und Schüler entwickeln ein Spiel, bei dem tickende Billardkugeln mit einer beweglichen Box eingefangen werden sollen. Projekt: z.B. Autospiel Die Schülerinnen und Schüler entwickeln ein Autospiel, bei dem ein Auto durch einen Wald fahren und mit Bäumen kollidieren kann.
3. Erarbeitung einer Simulation mit grafischen Objekten, die sich durch unterschiedliche Ergänzungen voneinander unterscheiden (Vererbung durch Spezialisierung ohne Überschreiben von Methoden) (a) Analyse und Erläuterung einer Basisversion der grafischen Klasse (b) Realisierung von grafischen Erweiterungen zur Basisklasse mit und ohne Vererbung (c) Verallgemeinerung und Reflexion des Prinzips der Vererbung am Beispiel der Spezialisierung		Projekt: z.B. Schneemann Die Schülerinnen und Schüler erstellen eine Simulation von Schneemännern, die unterschiedliche Kopfbedeckungen tragen.
4. Entwicklung einer komplexeren Simulation mit grafischen Elementen, die unterschiedliche Animationen durchführen (Vererbung mit Überschreiben von Methoden) (a) Analyse und Erläuterung einer einfachen grafischen Animationsklasse		Projekt: z.B. Flummibälle Die Schülerinnen und Schüler entwickeln eine Simulation von Flummibällen, bei der unterschiedliche Bälle unterschiedliche Bewegungen durchführen. Projekt: z.B. Weihnachtsbaum

(b) Spezialisierung der Klasse zu Unterklassen mit verschiedenen Animationen durch Überschreiben der entsprechenden Animationsmethode (c) Reflexion des Prinzips der späten Bindung		Die Schülerinnen und Schüler entwickeln eine Simulation eines Weihnachtsbaums mit Hilfe einer abstrakten Klasse Schmuck.

Unterrichtsvorhaben EF-V

Thema: Such- und Sortieralgorithmen anhand kontextbezogener Beispiele

Zeitbedarf: ca. 12 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
Binäre Suche auf sortierten Daten (a) Suchaufgaben im Alltag und im Kontext informatischer Systeme (b) Evtl. Simulationsspiel zum effizienten Suchen mit binärer Suche (c) Implementierung der binären Suche Effizienzbetrachtungen zur binären Suche	Die Schülerinnen und Schüler • 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).	Projekt: z.B. Helferprogramm für das Zahlenratespiel Projekt: z.B. NutzerIDs am Kopierer
Explorative Erarbeitung eines Sortierverfahrens (a) Sortierprobleme im Kontext informatischer Systeme und im Alltag (z.B. Dateisortierung, Tabellenkalkulation, Telefonbuch, Bundesligatabelle, usw.) (b) Vergleich zweier Elemente als Grundlage eines Sortieralgorithmus (c) Erarbeitung eines Sortieralgorithmus durch die Schülerinnen und Schüler		Material: z.B.: Kartenspiele
Systematisierung von Algorithmen und Effizienzbetrachtungen (a) Formulierung und Erläuterung von mehreren Algorithmen im Pseudocode (Sortieren durch Einfügen, Sortieren		Projekt: z.B. Bücherregal Schülerinnen und Schüler visualisieren die Abläufe der Sortieralgorithmen

durch Vertauschen, Sortieren durch Auswählen) (b) Bewertung von Algorithmen anhand der Anzahl der nötigen Vergleiche (c) Effizienzbetrachtungen an einem konkreten Beispiel bezüglich der Rechenzeit und des Speicherplatzbedarfs		

Unterrichtsvorhaben EF-VI

Thema: Digitalisierung und Grundlagen des Datenschutzes

Zeitbedarf: ca. 6 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
Binärcodierungen (a) natürliche Zahlen, (b) Ganzzahlen, (c) Zeichen	Die Schülerinnen und Schüler • stellen ganze Zahlen und Zeichen in Binärcodes dar (D), • interpretieren Binärcodes als Zahlen und Zeichen (D), • bewerten anhand von Fallbeispielen die Auswirkungen des Einsatzes von Informatiksystemen (A),	(Kann jederzeit an passender Stelle in das Unterrichtsgeschehen eingefügt werden)
Betrachtung des Themas Datenschutz Selbstentdeckendes Erkunden: **1.** Umfang von Datensammlungen, **2.** Vorgehen bei Datensammlungen, **3.** wirtschaftlichen Interessen bei Datensammlungen **4.** Motivation der Akteure Problematisierung und Anknüpfung an die Lebenswelt der Schülerinnen und Schüler		Material: Onlinespiel DataDealer (Kann jederzeit an passender Stelle in das Unterrichtsgeschehen eingefügt werden)

2.1.2 Übersichtsraster Unterrichtsvorhaben Qualifikationsphase GK

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

Die Schülerinnen und Schüler

- verwenden die Fachsprache bei der Kommunikation über informatische Sachverhalte (K),
- nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Dateien unter Berücksichtigung der Rechteverwaltung (K),
- organisieren und koordinieren kooperatives und eigenverantwortliches Arbeiten (K),
- strukturieren den Arbeitsprozess, vereinbaren Schnittstellen und führen Ergebnisse zusammen (K),
- beurteilen Arbeitsorganisation, Arbeitsabläufe und Ergebnisse (K),
- präsentieren Arbeitsabläufe und -ergebnisse adressatengerecht (K).

Unterrichtsvorhaben Q1-I:

Thema: Kryptographie

Zeitbedarf: 20 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
• Erarbeitung und Implementierung von klassischen symmetrischen Verschlüsselungsverfahren. **(a)** Zeichenkettenoperationen **(b)** GUI-Entwicklung mit dem Java-Editor **(c)** ASCII-Codierung **(d)** symmetrische Verschlüsselung • Erarbeitung von asymmetrischen Verschlüsselungverfahren.	Die Schülerinnen und Schüler • analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme systematisch anhand von Beispielen (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • analysieren und erläutern Eigenschaften und Einsatzbereiche symmetrischer und asymmetrischer Verschlüsselungsverfahren (A),	Cäsar-Verfahren Vigenere-Verfahren Häufigkeitsanalyse Matheprisma: Modul RSA

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

Unterrichtsvorhaben Q1-II:

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

Zeitbedarf: 30 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Die Datenstruktur Schlange im Anwendungskontext unter Implementierung einer Klasse Warteschlange **(a)** Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen **(b)** Erarbeitung der Funktionalität einer Warteschlange **(c)** Modellierung und Implementierung der Datenstruktur Warteschlange und der Anwendung.	Die Schülerinnen und Schüler • 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), • analysieren und erläutern objektorientierte Modellierungen (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),	*Einstieg:* Obstampel Analogie Korb mit Haken zu Knotenklasse nutzen. Die Obstampelsimulation soll programmiert werden. *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

	• modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M), • stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D),	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.
2. Die Datenstruktur Stapel im Anwendungskontext unter Nutzung der Klasse Stack **(a)** Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen **(b)** Erarbeitung der Funktionalität der Klasse Stack **(c)** Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse Stack	• dokumentieren Klassen (D), • stellen die Kommunikation zwischen Objekten grafisch dar (D). • 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),	*Beispiel:* PostFixRechner Es soll ein Taschenrechner mit GUI erstellt werden, der nach dem Postfix-Prinzip mit Hilfe eines Stapel arbeitet. *Beispiel:* Integeraddition Ganzzahlen außerhalb des durch den Datentyp int abgedeckten Bereichs sollen mit Hilfe von Stapeln addiert werden.
3. Die Datenstruktur lineare Liste im Anwendungskontext unter Nutzung der Klasse List **(a)** Erarbeitung der Vorteile der Klasse List im Gegensatz zu den bereits bekannten linearen Strukturen **(b)** Modellierung und Implementierung einer kontextbezogenen Anwendung unter Verwendung der Klasse List.	• testen Programme systematisch anhand von Beispielen (I), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), • stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D).	*Einstieg:* Erarbeitung der Operationen der Klasse List und einfacher Iterationen mit Hilfe der Listendemo. *Beispiel:* ToDoListe

		Es sollen Aufgaben mit Priorität in einer Liste gespeichert werden. Hier wird auch das Prinzip der Prioritätswarteschlange deutlich.
4. Vertiefung - Anwendungen von Listen, Stapeln oder Schlangen in mindestens einem weiteren Kontext **(a)** Die Vertiefung kann in Form von Aufgaben oder weiteren Projekten geschehen.		*Mögliches Beispiel:* Skispringen Ein Skispringen hat folgenden Ablauf: Nach dem Sprung erhält der Springer eine Punktzahl und wird nach dieser Punktzahl in eine Rangliste eingeordnet. Die besten 30 Springer qualifizieren sich für den zweiten Durchgang. Sie starten in umgekehrter Reihenfolge gegenüber der Platzierung auf der Rangliste. Nach dem Sprung erhält der Springer wiederum eine Punktzahl und wird nach der Gesamtpunktzahl aus beiden Durchgängen in die endgültige Rangliste eingeordnet. *Mögliches Beispiel*: Rangierbahnhof Auf einem Güterbahnhof gibt es drei Gleise, die nur zu einer Seite offen sind. Wagons können also von einer Seite auf das Gleis fahren und nur rückwärts wieder hinausfahren. Die Wagons tragen Nummern, wobei die Nummer jedoch erst eingesehen werden

		kann, wenn der Wagon der vorderste an der offenen Gleisseite ist. (Zwischen den Wagons herumzuturnen, um die anderen Wagonnummern zu lesen, wäre zu gefährlich.) Zunächst stehen alle Wagons unsortiert auf einem Gleis. Ziel ist es, alle Wagons in ein anderes Gleis zu fahren, so dass dort die Nummern der Wagons vom Gleisende aus aufsteigend in richtiger Reihenfolge sind. Zusätzlich zu diesen beiden Gleisen gibt es ein Abstellgleis, das zum Rangieren benutzt werden kann.

Unterrichtsvorhaben Q2-III:

Thema: von-Neumann-Architektur und Ausführung maschinennaher Programme.

Zeitbedarf: 9 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien oder Materialien
1. Von-Neumann-Architektur und die Ausführung maschinennaher Programme a) prinzipieller Aufbau einer von Neumann-Architektur mit CPU, Rechenwerk, Steuerwerk, Register und Hauptspeicher a) einige maschinennahe Befehlen und ihre Repräsentation in einem Binär-Code, der in einem Register gespeichert werden kann b) Analyse und Erläuterung der Funktionsweise eines einfachen maschinennahen Programms	Die Schülerinnen und Schüler • erläutern die Ausführung eines einfachen maschinennahen Programms sowie die Datenspeicherung auf einer „Von-Neumann-Architektur“ (A),	*Material*: Simulationsprogramm Johnny *Beispiel*: Addition zweier Zahlen, Multiplikation durch Addition abbilden

Unterrichtsvorhaben Q1-IV:

Thema: Suchen und Sortieren auf linearen Datenstrukturen

Zeitbedarf: 15 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Suchen von Daten in Listen und Arrays **(a)** Lineare Suche in Listen und in Arrays **(b)** Binäre Suche in Arrays als Beispiel für rekursives Problemlösen **(c)** Untersuchung der beiden Suchverfahren hinsichtlich ihrer Effizienz (Laufzeitverhalten, Speicherbedarf)	Die Schülerinnen und Schüler • analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A), • entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“ und „Teilen und Herrschen“ (M), • modifizieren Algorithmen und Programme (I), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I),	*Beispiel*: Bundesjugendspiele Anhand der Läuferliste eines Laufwettbewerbs mit erreichten Zeiten sollen verschiedene Aufgaben gelöst werden: • Finden des schnellsten Läufers. • Finden des schnellsten Mannes. • Berechnen der Durchschnittszeit. • Sortieren nach der Laufzeit oder alphabetisch. *Beispiel:* Simulationsprogramm mit einem Array von int-Werten An diesem Beispiel können große Datenmengen simuliert werden und damit empirische Laufzeitanalysen vorgenommen werden.
2. Sortieren in Listen und Arrays - Entwicklung und Implementierung von iterativen und rekursiven Sortierverfahren • Entwicklung und Implementierung eines einfachen Sortierverfahrens für eine Liste • Implementierung eines einfachen Sortierverfahrens für ein Feld • Entwicklung eines rekursiven Sortierverfahren für eine Liste (z.B. Quicksort)		
3. Untersuchung der Effizienz der Sortierverfahren „Sortieren durch direktes Einfügen“ und „Quicksort“ auf linearen Listen		

(a) Grafische Veranschaulichung der Sortierverfahren **(b)** Untersuchung der Anzahl der Vergleichsoperationen und des Speicherbedarf bei beiden Sortierverfahren **(c)** Beurteilung der Effizienz der beiden Sortierverfahren	• 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), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme systematisch anhand von Beispielen (I), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D).	

Unterrichtsvorhaben Q1-V:

Thema: Modellierung und Nutzung von relationalen Datenbanken in Anwendungskontexten

Zeitbedarf: 30 Stunden

Sequenzierung des Unterrichtsvorhabens

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**

1. Nutzung von relationalen Datenbanken
• Aufbau von Datenbanken und Grundbegriffe
  • Entwicklung von Fragestellungen zur vorhandenen Datenbank
  • Analyse der Struktur der vorgegebenen Datenbank und Erarbeitung der Begriffe Tabelle, Attribut, Datensatz, Datentyp, Primärschlüssel, Fremdschlüssel, Datenbankschema
• SQL-Abfragen
  • Analyse vorgegebener SQL-Abfragen und Erarbeitung der Sprachelemente von SQL (SELECT (DISTINCT) …FROM, WHERE, AND, OR, NOT) auf einer Tabelle
  • Analyse und Erarbeitung von SQL-Abfragen auf einer und mehrerer Tabelle zur Beantwortung der Fragestellungen (JOIN, UNION, AS, GROUP BY,ORDER BY, ASC, DESC, COUNT, MAX, MIN, SUM, Arithmetische Operatoren: +, -, *, /, (…), Vergleichsoperatoren: =, <>, >, <, >=, <=, LIKE, BETWEEN, IN, IS NULL)
	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),
		Beispiel: Fehlstundendatenbank
		Beispiel: Präsidentendatenbank

• Vertiefung an einem weiteren Datenbankbeispiel	• modellieren zu einem Entity-Relationship-Diagramm ein relationales Datenbankschema (M), • bestimmen Primär- und Sekundärschlüssel (M), • überführen Datenbankschemata in vorgegebene Normalformen (M), • verwenden die Syntax und Semantik einer Datenbankabfragesprache, um Informationen aus einen Datenbanksystem zu extrahieren (I), • ermitteln Ergebnisse von Datenbankabfragen über mehrere verknüpfte Tabellen (D), • stellen Entitäten mit ihren Attributen und die Beziehungen zwischen Entitäten in einem Entity-Relationship-Diagramm grafisch dar (D), • überprüfen Datenbankschemata auf vorgegebene Normalisierungseigenschaften (D). • untersuchen und bewerten anhand von Fallbeispielen 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	
2. Modellierung von relationalen Datenbanken **(a)** Entity-Relationship-Diagramm • 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 **(b)** Entwicklung einer Datenbank aus einem Datenbankentwurf • Modellierung eines relationalen Datenbankschematas zu einem Entity-Relationship-Diagramm inklusive der Bestimmung von Primär- und Sekundärschlüsseln **(c)** Redundanz, Konsistenz und Normalformen • 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: 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 9 im Differenzierungsbereich im Fach Informatik die Note „sehr gut“ erhalten hat. Dazu ist die Datenbank zu modellieren, ggf. zu normalisieren und im Datenbanksystem umzusetzen. Weiter sollen sinnvolle Abfragen entwickelt werden und das Thema Datenschutz besprochen werden.

3. Fallbeispiele zum Datenschutz	gesellschaftlicher Werte unter Berücksichtigung unterschiedlicher Interessenlagen (A), • nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zum Erschließen, zur Aufbereitung und Präsentation fachlicher Inhalte (D).	Auszug aus dem Bundesdatenschutzgesetz Volkszählungsurteil passende Fallbeispiele

Unterrichtsvorhaben Q2-I:

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

Zeitbedarf: 26 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Analyse von Baumstrukturen in verschiedenen Kontexten **(a)** Grundlegende Begriffe (Grad, Tiefe, Höhe, Blatt, Inhalt, Teilbaum, Ebene, Vollständigkeit) **(b)** Aufbau und Darstellung von binären Bäumen anhand von Baumstrukturen in verschiedenen Kontexten **2. Die Datenstruktur Binärbaum im Anwendungskontext unter Nutzung der Klasse BinaryTree** **(a)** Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen im Anwendungskontext **(b)** Modellierung eines Entwurfsdiagramms und Entwicklung eines Implementationsdiagramms **(c)** Erarbeitung der Klasse BinaryTree und beispielhafte Anwendung der Operationen **(d)** Implementierung der Anwendung oder von Teilen der Anwendung	Die Schülerinnen und Schüler • 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),	*Beispiel:* Morsebaum Das Morsealphabet wird anhand des Kriteriums Punkt (links) und Strich (rechts) in einem Binärbaum codiert. Mit Hilfe dieses Morsebaumes sollen Morsenachrichten decodiert und codiert werden. *Mögliches Beispiel:* Termbaum Der Aufbau von Termen wird mit Hilfe von binären Baumstrukturen verdeutlicht. *Mögliches Beispiel*: Ahnenbaum Die binäre Baumstruktur ergibt sich daraus, dass jede Person genau einen Vater und eine Mutter hat. *Mögliches Beispiel*: Entscheidungsbäume Um eine Entscheidung zu treffen, werden mehrere Fragen mit ja oder nein beantwortet.

(e) Traversierung eines Binärbaums im Pre-, In- und Postorderdurchlauf	• 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).	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.
3. Die Datenstruktur binärer Suchbaum im Anwendungskontext unter Verwendung der Klasse BinarySearchTree **(a)** Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen **(b)** Modellierung eines Entwurfsdiagramms und Entwicklung eines Implementationsdiagramm, **(c)** grafische Darstellung eines binären Suchbaums und Erarbeitung der Struktureigenschaften **(d)** Erarbeitung der Klasse BinarySearchTree und Einführung des Interface Item zur Realisierung einer geeigneten Ordnungsrelation **(e)** Implementierung der Anwendung oder von Teilen der Anwendung inklusive einer sortierten Ausgabe des Baums		*Mögliches Beispiel:* Suchbaum mit Ganzzahlen *Mögliches Beispiel*: Informatikerbaum als *Suchbaum* In einem binären *Suchbaum* werden die Namen und die Geburtsdaten von Informatikern lexikographisch geordnet abgespeichert.
4. Übung und Vertiefungen der Verwendung von Binärbäumen oder binären Suchbäumen anhand weiterer Problemstellungen		*Mögliche Beispiel:* Codierungsbäume oder Huffman-Codierung *mögliches Beispiel*: Buchindex

Unterrichtsvorhaben Q2-II:

Thema: Endliche Automaten und formale Sprachen

Zeitbedarf: 21 Stunden
Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien oder Materialien
1. Endliche Automaten • Vom Automaten in den Schülerinnen und Schülern bekannten Kontexten zur formalen Beschreibung eines endlichen Automaten • Untersuchung, Darstellung und Entwicklung endlicher Automaten	Die Schülerinnen und Schüler • 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),	*Mögliche Beispiele:* Cola-Automat, Videorekorder, Akzeptor für bestimmte Zahlen, Akzeptor für Teilwörter in längeren Zeichenketten, Akzeptor für Terme
2. Untersuchung und Entwicklung von Grammatiken regulärer Sprachen **(a)** Erarbeitung der formalen Darstellung regulärer Grammatiken **(b)** Untersuchung, Modifikation und Entwicklung von Grammatiken **(c)** Entwicklung von endlichen Automaten zum Erkennen regulärer Sprachen die durch Grammatiken gegeben werden **(d)** Entwicklung regulärer Grammatiken zu endlichen Automaten		*Mögliche Beispiele:* reguläre Grammatik für Wörter mit ungerader Parität, Grammatik für Wörter, die bestimmte Zahlen repräsentieren, Satzgliederungsgrammatik
3. Grenzen endlicher Automaten und **4. Grenzen der Automatisierbarkeit** (a) Vorstellung des Halteproblems		*Beispiele:* Klammerausdrücke, a^nb^n im Vergleich zu $(ab)^n$

(b) Unlösbarkeit des Halteproblems (c) Beurteilung des Einsatzes von Informatiksystemen hinsichtlich prinzipieller Möglichkeiten und prinzipieller Grenzen	• 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 jeweils andere Darstellungsform (D), • ermitteln die Sprache, die ein endlicher Automat akzeptiert (D). • beschreiben an Beispielen den Zusammenhang zwischen Automaten und Grammatiken (D). • untersuchen und beurteilen Grenzen des Problemlösens mit Informatiksystemen (A).	Halteproblem

Unterrichtsvorhaben Q2-III:

Thema: Netzstrukturen

Zeitbedarf: 9 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Daten in Netzwerken und Sicherheitsaspekte in Netzen sowie beim Zugriff auf Datenbanken **(a)** Beschreibung eines Datenbankzugriffs im Netz anhand eines Anwendungskontextes und einer Client-Server-Struktur zur Klärung der Funktionsweise eines Datenbankzugriffs **(b)** Netztopologien als Grundlage von Client-Server-Strukturen und TCP/IP-Schichtenmodell als Beispiel für eine Paketübermittlung in einem Netz **(c)** Vertraulichkeit, Integrität, Authentizität in Netzwerken sowie symmetrische und asymmetrische kryptografische Verfahren (Cäsar-, Vigenère-, RSA-Verfahren) als Methoden Daten im Netz verschlüsselt zu übertragen	Die Schülerinnen und Schüler • beschreiben und erläutern Topologien, die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken (A),	*Material:* FILIUS

Unterrichtsvorhaben Q2-IV:

Wiederholung und Vertiefung ausgewählter Kompetenzen und Inhalte des ersten Jahres der Qualifikationsphase

2.1.3 Übersichtsraster Unterrichtsvorhaben Qualifikationsphase LK

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

Die Schülerinnen und Schüler

- verwenden die Fachsprache bei der Kommunikation über informatische Sachverhalte (K),
- nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Dateien unter Berücksichtigung der Rechteverwaltung (K),
- organisieren und koordinieren kooperatives und eigenverantwortliches Arbeiten (K),
- strukturieren den Arbeitsprozess, vereinbaren Schnittstellen und führen Ergebnisse zusammen (K),
- beurteilen Arbeitsorganisation, Arbeitsabläufe und Ergebnisse (K),
- präsentieren Arbeitsabläufe und -ergebnisse adressatengerecht (K).

Unterrichtsvorhaben Q1-I:

Thema: Kryptographie

Zeitbedarf: 20 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. **Erarbeitung und Implementierung von klassischen symmetrischen Verschlüsselungsverfahren.** 1 Zeichenkettenoperationen 1 GUI-Entwicklung mit dem Java-Editor 2 ASCII-Codierung 3 symmetrische Verschlüsselung **2. Erarbeitung von asymmetrischen Verschlüsselungverfahren.**	Die Schülerinnen und Schüler • analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme systematisch anhand von Beispielen (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • analysieren und erläutern Eigenschaften und Einsatzbereiche symmetrischer und asymmetrischer Verschlüsselungsverfahren (A), • wenden eine didaktisch orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I),	Cäsar-Verfahren Vigenere-Verfahren Häufigkeitsanalyse Matheprisma: Modul RSA

Unterrichtsvorhaben Q1-II:

Thema: Modellierung und Implementierung dynamischer linearer Datenstrukturen und Implementierung von Anwendungen unter Verwendung dynamischer, linearer Datenstrukturen

Zeitbedarf: 30 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Die Datenstruktur Schlange im Anwendungskontext unter Implementierung einer Klasse Warteschlange • Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen • Erarbeitung der Funktionalität einer Warteschlange • Modellierung und Implementierung der Datenstruktur Warteschlange und der Anwendung.	Die Schülerinnen und Schüler • modellieren Klassen mit ihren Attributen, Methoden und Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), • erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), • analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache	*Einstieg:* Obstampel Analogie Korb mit Haken zu Knotenklasse nutzen. Die Obstampelsimulation soll programmiert werden. *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

	Datentypen, Objekttypen oder 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 Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (I), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I), • stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D).	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.
3. 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:* PostFixRechner Es soll ein Taschenrechner mit GUI erstellt werden, der nach dem Postfix-Prinzip mit Hilfe eines Stapel arbeitet. *Beispiel:* Integeraddition Ganzzahlen außerhalb des durch den Datentyp int abgedeckten Bereichs sollen mit Hilfe von Stapeln addiert werden.
4. 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.		*Einstieg:* Erarbeitung der Operationen der Klasse List und einfacher Iterationen mit Hilfe der Listendemo. *Beispiel:* ToDoListe

		Es sollen Aufgaben mit Priorität in einer Liste gespeichert werden. Hier wird auch das Prinzip der Prioritätswarteschlange deutlich.
5. Vertiefung - Anwendungen von Listen, Stapeln oder Schlangen in mindestens einem weiteren Kontext • Die Vertiefung kann in Form von Aufgaben oder weiteren Projekten geschehen.		*Mögliches Beispiel:* Skispringen Ein Skispringen hat folgenden Ablauf: Nach dem Sprung erhält der Springer eine Punktzahl und wird nach dieser Punktzahl in eine Rangliste eingeordnet. Die besten 30 Springer qualifizieren sich für den zweiten Durchgang. Sie starten in umgekehrter Reihenfolge gegenüber der Platzierung auf der Rangliste. Nach dem Sprung erhält der Springer wiederum eine Punktzahl und wird nach der Gesamtpunktzahl aus beiden Durchgängen in die endgültige Rangliste eingeordnet. *Mögliches Beispiel*: Rangierbahnhof Auf einem Güterbahnhof gibt es drei Gleise, die nur zu einer Seite offen sind. Wagons können also von einer Seite auf das Gleis fahren und nur rückwärts wieder hinausfahren. Die Wagons tragen Nummern, wobei die Nummer jedoch erst eingesehen werden

		kann, wenn der Wagon der vorderste an der offenen Gleisseite ist. (Zwischen den Wagons herumzuturnen, um die anderen Wagonnummern zu lesen, wäre zu gefährlich.) Zunächst stehen alle Wagons unsortiert auf einem Gleis. Ziel ist es, alle Wagons in ein anderes Gleis zu fahren, so dass dort die Nummern der Wagons vom Gleisende aus aufsteigend in richtiger Reihenfolge sind. Zusätzlich zu diesen beiden Gleisen gibt es ein Abstellgleis, das zum Rangieren benutzt werden kann.

Unterrichtsvorhaben Q1-III:

Thema: Suchen und Sortieren auf linearen Datenstrukturen

Zeitbedarf: 15 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
*1.***Suchen von Daten in Listen und Arrays** **5.** Lineare Suche in Listen und in Arrays **6.** Binäre Suche in Arrays als Beispiel für rekursives Problemlösen **7.** Untersuchung der beiden Suchverfahren hinsichtlich ihrer Effizienz (Laufzeitverhalten, Speicherbedarf)	Die Schülerinnen und Schüler **1.** analysieren und erläutern Algorithmen und Programme (A), **2.** beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), **3.** beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A), **4.** entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“ , „Teilen und Herrschen“ und „Backtracking“(M), **5.** modifizieren Algorithmen und Programme (I), **6.** implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), **7.** implementieren und erläutern iterative und rekursive Such- und Sortierverfahren unterschiedlicher Komplexitätsklassen (Speicherbedarf und Laufzeitverhalten)(I),	*Beispiel*: Bundesjugendspiele Anhand der Läuferliste eines Laufwettbewerbs mit erreichten Zeiten sollen verschiedene Aufgaben gelöst werden: • Finden des schnellsten Läufers. • Finden des schnellsten Mannes. • Berechnen der Durchschnittszeit. • Sortieren nach der Laufzeit oder alphabetisch. *Beispiel:* Simulationsprogramm mit einem Array von int-Werten An diesem Beispiel können große Datenmengen simuliert werden und damit empirische Laufzeitanalysen vorgenommen werden.
2. Sortieren in Listen und Arrays - Entwicklung und Implementierung von iterativen und rekursiven Sortierverfahren **3.** Entwicklung und Implementierung eines einfachen Sortierverfahrens für eine Liste **4.** Implementierung eines einfachen Sortierverfahrens für ein Feld **5.** Entwicklung eines rekursiven Sortierverfahren für eine Liste (z.B. Quicksort)		

3. Untersuchung der Effizienz der Sortierverfahren „Sortieren durch direktes Einfügen“ und „Quicksort“ auf linearen Listen ● Grafische Veranschaulichung der Sortierverfahren ● Untersuchung der Anzahl der Vergleichsoperationen und des Speicherbedarf bei beiden Sortierverfahren ● Beurteilung der Effizienz der beiden Sortierverfahren	**1.** nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), **2.** interpretieren Fehlermeldungen und korrigieren den Quellcode (I), **3.** testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I), **4.** stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D).	

Unterrichtsvorhaben Q1-IV:

Thema: Modellierung und Implementierung von Baumstrukturen und Implementierung von Anwendungen unter Verwendung von Baumstrukturen

Zeitbedarf: 20 Stunden

Sequenzierung des Unterrichtsvorhabens

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Analyse von Baumstrukturen in verschiedenen Kontexten • Grundlegende Begriffe (Grad, Tiefe, Höhe, Blatt, Inhalt, Teilbaum, Ebene, Vollständigkeit) • Aufbau und Darstellung von binären Bäumen anhand von Baumstrukturen in verschiedenen Kontexten **2. Die Datenstruktur Binärbaum im Anwendungskontext unter Nutzung der Klasse BinaryTree** • Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen im Anwendungskontext • Modellierung eines Entwurfsdiagramms und Entwicklung eines Implementationsdiagramms • Erarbeitung der Klasse BinaryTree und beispielhafte Anwendung der Operationen • Implementierung der Anwendung oder von Teilen der Anwendung	Die Schülerinnen und Schüler • erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), • analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A), • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare und nichtlineare Datensammlungen zu (M), • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M),	*Beispiel:* Morsebaum Das Morsealphabet wird anhand des Kriteriums Punkt (links) und Strich (rechts) in einem Binärbaum codiert. Mit Hilfe dieses Morsebaumes sollen Morsenachrichten decodiert und codiert werden. *Mögliches Beispiel:* Termbaum Der Aufbau von Termen wird mit Hilfe von binären Baumstrukturen verdeutlicht. *Mögliches Beispiel*: Ahnenbaum Die binäre Baumstruktur ergibt sich daraus, dass jede Person genau einen Vater und eine Mutter hat. *Mögliches Beispiel*: Entscheidungsbäume Um eine Entscheidung zu treffen, werden mehrere Fragen mit ja oder nein beantwortet.

• Traversierung eines Binärbaums im Pre-, In- und Postorderdurchlauf	• 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“ und „Backtra-ing“(M), • entwickeln mit didaktisch orientierten Entwicklungsumgebungen einfache Benutzungsoberflächen zur Kommunikation mit einem Informatiksystem (M), • implementieren Operationen dynamischer (linearer oder nichtlinearer) Datenstrukturen (I), • implementieren und erläutern iterative und rekursive Such- und Sortierverfahren unterschiedlicher Komplexitätsklassen (Speicherbedarf und Laufzeitverhalten) (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme systematisch anhand von Beispielen und mithilfe von Testanwendungen(I), • wenden didaktisch orientierte Entwicklungsumgebungen zur Demonstration, zum Entwurf, zur	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.
• **Die Datenstruktur binärer Suchbaum im Anwendungskontext unter Verwendung der Klasse BinarySearchTree** • Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen • Modellierung eines Entwurfsdiagramms 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		*Mögliches Beispiel:* Suchbaum mit Ganzzahlen *Mögliches Beispiel*: Informatikerbaum als *Suchbaum* In einem binären *Suchbaum* werden die Namen und die Geburtsdaten von Informatikern lexikographisch geordnet abgespeichert.
• **Übung und Vertiefungen der Verwendung von Binärbäumen oder binären Suchbäumen anhand weiterer Problemstellungen**		*Mögliche Beispiel:* Codierungsbäume oder Huffman-Codierung

	Implementierung und zum Test von Informatiksystemen an (I), • stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D), • nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zur Erschließung, Aufbereitung und Präsentation fachlicher Inhalte (D),	*mögliches Beispiel*: Buchindex

Unterrichtsvorhaben Q1-V:

Thema: Modellierung, Implementierung und Nutzung von relationalen Datenbanken in Anwendungskontexten

Zeitbedarf: 30 Stunden

Sequenzierung des Unterrichtsvorhabens

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Nutzung von relationalen Datenbanken, Aufbau von Datenbanken und Grundbegriffe • Entwicklung von Fragestellungen zur vorhandenen Datenbank • Analyse der Struktur der vorgegebenen Datenbank und Erarbeitung der Begriffe Tabelle, Attribut, Datensatz, Datentyp, Primärschlüssel, Fremdschlüssel, Datenbankschema **2. SQL-Abfragen**	Die Schülerinnen und Schüler • erläutern die Eigenschaften und den Aufbau von Datenbanksystemen unter dem Aspekt der sicheren Nutzung (A), • analysieren und erläutern die Syntax und Semantik einer Datenbankabfrage (A), • analysieren und erläutern eine Datenbankmodellierung (A),	*Beispiel*: Fehlstundendatenbank *Beispiel*: Präsidentendatenbank

<table>
<tr>
<td>

- Analyse vorgegebener SQL-Abfragen und Erarbeitung der Sprachelemente von SQL (SELECT (DISTINCT) …FROM, WHERE, AND, OR, NOT) auf einer Tabelle
- Analyse und Erarbeitung von SQL-Abfragen auf einer und mehrerer Tabelle zur Beantwortung der Fragestellungen (JOIN, UNION, AS, GROUP BY,ORDER BY, ASC, DESC, COUNT, MAX, MIN, SUM, Arithmetische Operatoren: +, -, *, /, (…), Vergleichsoperatoren: =, <>, >, <, >=, <=, LIKE, BETWEEN, IN, IS NULL)

3. Vertiefung an einem weiteren Datenbankbeispiel

</td>
<td rowspan="2">

- erläutern die Eigenschaften normalisierter Datenbankschemata (A),
- bestimmen Primär- und Sekundärschlüssel (M),
- ermitteln für anwendungsbezogene Problemstellungen Entitäten, zugehörige Attribute, Relationen und Kardinalitäten (M),
- modifizieren eine Datenbankmodellierung (M),
- modellieren zu einem Entity-Relationship-Diagramm ein relationales Datenbankschema (M),
- bestimmen Primär- und Sekundärschlüssel (M),
- implementieren ein relationales Datenbankschema als Datenbank (I),
- 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).

</td>
<td></td>
</tr>
<tr>
<td>

2. Modellierung von relationalen Datenbanken

1) Entity-Relationship-Diagramm

- 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

2) Entwicklung einer Datenbank aus einem Datenbankentwurf

- Modellierung eines relationalen Datenbankschematas zu einem Entity-Relationship-Diagramm inklusive der Bestimmung von Primär- und Sekundärschlüsseln

</td>
<td>

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 9 im Differenzierungsbereich im Fach Informatik die Note „sehr gut“ erhalten hat. Dazu ist die Datenbank zu modellieren, ggf. zu normalisieren und im Datenbanksystem umzusetzen.

</td>
</tr>
</table>

3) Redundanz, Konsistenz und Normalformen • 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)	• 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).	Weiter sollen sinnvolle Abfragen entwickelt werden und das Thema Datenschutz besprochen werden.
3. Fallbeispiele zum Datenschutz		Auszug aus dem Bundesdatenschutzgesetz Volkszählungsurteil passende Fallbeispiele

Unterrichtsvorhaben Q1-VI:

Thema: Informatik, Mensch und Gesellschaft - Sicherheit und Datenschutz in Informatiksystemen, Auswirkungen und Grenzen der Automatisierung

Zeitbedarf: 10 Stunden

Sequenzierung des Unterrichtsvorhabens

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien oder Materialien
1. Einführung in die Problemfelder Datenschutz, Urheberrecht und moralische Verantwortung • Vorstellung eines komplexen Fallbeispiels • Erarbeitung von Interesse verschiedener Interessensgruppen im Hinblick auf die Problemfelder • Erster Bewertungsversuch auf Grundlage des Vorwissens von Schülerinnen und Schülern	Die Schülerinnen und Schüler • 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). • untersuchen und beurteilen Grenzen des Problemlösens mit Informatiksystemen (A). • nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zur Erschließung, Aufbereitung und Präsentation fachlicher Inhalte (D).	*Mögliches Fallbeispiel: autonomes Fahren* Ein namhaftes Softwareunternehmen möchte den Automobilmarkt mit selbstfahrenden Autos revolutionieren. Der Quellcode zu Steuerung der Fahrzeuge ist Firmengeheimnis, zur Verbesserung der Fahrleistung werden Bewegungsprofile erstellt und noch ist nicht klar, wie das Fahrzeug bei drohenden Unfällen mit Personenschaden reagieren soll. Eine erfolgreiche Einführung der Technologie würde den Straßenverkehr sicherer, schneller und ökologischer machen, allerdings auch zu tausenden von Arbeitslosen durch Wegfall ganzer Berufszweige führen.
2. Erarbeitung grundlegender Positionen (ggf. in zieldifferenten Gruppen) und Anwendung auf Fallbeispiele • Erarbeitung von Grundideen des Datenschutzes (z.B. personenbezogene Daten, informationelle Selbstbestimmung, Datensparsamkeit usw.) • Erarbeitung von Grundideen des Urheberrechts anhand eines verbreiteten Lizenzsystems • Erarbeitung eines einfachen moralischen Bewertungsmaßstabes • Anwendung auf Fallbeispiele		

Unterrichtsvorhaben Q1-VII:

Thema: Projektorientierte Softwareentwicklung am Beispiel einer Anwendung mit Datenbankanbindung

Zeitbedarf: 15 Stunden

Sequenzierung des Unterrichtsvorhabens

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien oder Materialien
1. Analyse einer lebensweltnahen Problemstellung im Hinblick auf die Entwicklung eines Java-Programms mit Datenbankanbindung • Entwicklung einer Programmidee • Analyse des Problembereichs • Entwicklung eines Anforderungskatalogs für das zu entwickelnde Programm	Die Schülerinnen und Schüler • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modellieren Klassen mit ihren Attributen, Methoden und Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare und nichtlineare Datensammlungen zu (M), • ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M), • stellen die Kommunikation zwischen Objekten grafisch dar (D), • stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen (D),	Hier ist je nach Interessenlage des Kurses ein geeignetes Beispiel abzusprechen. Die folgenden Beispiele sind exemplarisch aufgeführt: *Quizspiel* Verwaltung von Quizfragen, Antworten und Ranglisten *Verwaltung der Schülerbücherei* Verwaltungsprogramm für das Einpflegen und Ausleihe von Büchern der Schülerbibliothek *Fehlstundenverwaltung* Verwaltungsprogramm für die Fehlstunden von Schülerinnen und Schülern

	• analysieren und erläutern objektorientierte Modellierungen (A), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), • ermitteln für anwendungsbezogene Problemstellungen Entitäten, zugehörige Attribute, Relationen und Kardinalitäten (M), • stellen Entitäten mit ihren Attributen und die Beziehungen zwischen Entitäten mit Kardinalitäten in einem Entity-Relationship-Diagramm grafisch dar (D),	*Sportfestverwaltung* Verwaltung von Aufgaben, Sportereignissen und Ergebnissen des Schulsportfestes *Materialverwaltung* Materialien für Vertretungsstunden sollen verwaltet werden.
2. Modellierung einer datenbankgestützten Problemlösung unter Berücksichtigung des MVC-Prinzips • Strukturierung nach dem MVC-Prinzip • Modellierung der Datenbank (ER-Diagramm, Datenbankschema) • Modellierung der Kontrollklassen und Entwicklung von SQL-Anweisungen entsprechend des Anforderungskatalogs • Modellierung einer grafischen Benutzungsoberfläche	• modellieren zu einem Entity-Relationship-Diagramm ein relationales Datenbankschema (M), • bestimmen Primär- und Sekundärschlüssel (M), • implementieren ein relationales Datenbankschema als Datenbank (I), • analysieren und erläutern eine Datenbankmodellierung (A), • erläutern die Eigenschaften normalisierter Datenbankschemata (A),	
3. Umsetzung des Softwareprojektes • Umsetzung der Datenbank in einem Datenbankmanagementsystem • Implementierung der Kontrollklassen mit Anbindung an die Datenbank unter Verwendung didaktischer Klassen • Integration in den Prototypen der Benutzeroberfläche	• überprüfen Datenbankschemata auf vorgegebene Normalisierungseigenschaften (D), • überführen Datenbankschemata in die 1. bis 3. Normalform (M), • analysieren und erläutern Algorithmen und Programme (A), • modifizieren Algorithmen und Programme (I),	

	• testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I), • ermitteln Ergebnisse von Datenbankabfragen über mehrere verknüpfte Tabellen (D), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • analysieren und erläutern die Syntax und Semantik einer Datenbankabfrage (A), • verwenden die Syntax und Semantik einer Datenbankabfragesprache, um Informationen aus einem Datenbanksystem zu extrahieren (I), • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Daten, zur Organisation von Arbeitsabläufen sowie zur Verteilung und Zusammenführung von Arbeitsanteilen (K), • wenden didaktisch orientierte Entwicklungsumgebungen zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I), • entwickeln mit didaktisch orientierten Entwicklungsumgebungen einfache Benutzungsoberflächen zur Kommunikation mit einem Informatiksystem (M),	

	• erläutern Eigenschaften und Aufbau von Datenbanksystemen unter dem Aspekt der sicheren Nutzung (A), • untersuchen und bewerten anhand von Fallbeispielen Auswirkungen des Einsatzes von Informatiksystemen sowie Aspekte der Sicherheit von Informatiksystemen, des Datenschutzes und des Urheberrechts (A).	

Unterrichtsvorhaben Q2-I:

Thema: Modellierung und Implementierung von Graphen und Implementierung von Algorithmen auf Graphen in Anwendungssituationen

Zeitbedarf: 25 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien oder Materialien
1. Analyse von Graphen in verschiedenen Kontexten • Grundlegende Begriffe (Graph, gerichtet – ungerichtet, Knoten, Kanten, Kantengewicht) • Aufbau und Darstellung von Graphen anhand von Graphenstrukturen in verschiedenen Kontexten (Adjazenzmatrix, Adjazenzliste)	Die Schülerinnen und Schüler • erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), • analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A), • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare und nichtlineare Datensammlungen zu (M), • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M),	*Beispiel Soziale Netzwerke* Anhand dieses Beispiels werden die Darstellungsformen eines Graphen als Adjazenzmatrix oder mit Adjazenzlisten eingeführt. Das Beispiel wird in beiden Darstellungsformen implementiert und einfache Algorithmen werden umgesetzt.
2. Die Datenstruktur Graph im Anwendungskontext unter Nutzung der Klassen Graph, Vertex und Edge. • Erarbeitung der Klassen Graph, Vertex und Edge und beispielhafte Anwendung der Operationen • Bestimmung von Wegen in Graphen im Anwendungskontext (Tiefensuche, Breitensuche) • Bestimmung von kürzesten Wegen in Graphen im Anwendungskontext (Backtracking, Dijkstra). • Bestimmung von minimalen Spannbäumen eines Graphen im		Beispiel: Soziale Netzwerke Ausgehend von dem Problem der Berechnung der Dichte eines sozialen Netzwerkes werden die Funktionalitäten der Methoden der Klassen Graph, Vertex und Edge erarbeitet und erste Beispiele modelliert und implementiert: • Konstruktion eines Beispielgraphen • Anzahl der Knoten • Summe der Kantengewichte • Anzahl der Nachbarn eines Knotens

Anwendungskontext mithilfe des Prim- oder des Kruskal-Algorithmus	• 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“ und „Backtraing“(M), • entwickeln mit didaktisch orientierten Entwicklungsumgebungen einfache Benutzungsoberflächen zur Kommunikation mit einem Informatiksystem (M), • implementieren Operationen dynamischer (linearer oder nichtlinearer) Datenstrukturen (I), • implementieren und erläutern iterative und rekursive Such- und Sortierverfahren unterschiedlicher Komplexitätsklassen (Speicherbedarf und Laufzeitverhalten) (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme systematisch anhand von Beispielen und mithilfe von Testanwendungen(I), • wenden didaktisch orientierte Entwicklungsumgebungen zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I),	Beispiel: Navigationsgraph Ausgehend von dem Problem der Suche eines beliebigen Weges zwischen zwei Knoten in einem Graphen werden die Traversierungsalgorithmen (Breiten- und Tiefensuche) erarbeitet. Ausgehend von der Tiefensuche in der Variante des Backtracking wird ein Algorithmus zur Bestimmung eines kürzesten Weges zwischen zwei Knoten implementiert. Anschließend wird der Dijkstra-Algorithmus auf mehrere Beispiele angewendet und implementiert. Der Vergleich der beiden Algorithmen unter Effizienzaspekten ist Bestandteil des Unterrichts. Beispiel: Versorgungsnetz Die Problemstellung, Verbraucher eines Dorfes möglichst kostengünstig an ein Versorgungsnetz (Kabel, Gas, Telefon) anzuschließen, motiviert die Behandlung des minimalen Spannbaumes eines Graphen. Mindestens einer der beiden Algorithmen wird von der Lerngruppe erarbeitet und implementiert. In leistungsstarken Lerngruppen können auch beide Algorithmen behandelt werden. So kann deutlich werden, dass

	• stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D), • nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zur Erschließung, Aufbereitung und Präsentation fachlicher Inhalte (D),	es u.U. zu einem Graphen mehrere minimale Spannbäume gibt.

Unterrichtsvorhaben Q2-II:

Thema: Endliche Automaten und formale Sprachen sowie die Entwicklung eines Parsers zu einer formalen Sprache

Zeitbedarf: 30 Stunden
Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien oder Materialien
1. Endliche Automaten • Vom Automaten in den Schülerinnen und Schülern bekannten Kontexten zur formalen Beschreibung eines endlichen Automaten • Untersuchung, Darstellung und Entwicklung endlicher Automaten	Die Schülerinnen und Schüler • analysieren und erläutern die Eigenschaften endlicher Automaten und Kellerautomaten einschließlich ihres Verhaltens auf bestimmte Eingaben (A), • analysieren und erläutern Grammatiken regulärer und kontextfreier Sprachen (A), • erläutern die Grenzen endlicher Automaten und regulärer Sprachen im Anwendungszusammenhang (A),	*Mögliche Beispiele:* Cola-Automat, Videorekorder, 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		*Mögliche Beispiele:* reguläre Grammatik für Wörter mit ungerader Parität, Grammatik für Wörter, die bestimmte

• Entwicklung von endlichen Automaten zum Erkennen regulärer Sprachen die durch Grammatiken gegeben werden • Entwicklung regulärer Grammatiken zu endlichen Automaten. • Entwicklung eines Parsers für eine einfache reguläre Sprache.	• ermitteln die formale Sprache, die durch eine Grammatik erzeugt wird (A), • entwickeln und modifizieren zu einer Problemstellung endliche Automaten oder Kellerautomaten (M), • entwickeln zur akzeptierten Sprache eines Automaten die zugehörige Grammatik (M),	Zahlen repräsentieren, Satzgliederungsgrammatik
3. Grenzen endlicher Automaten und Grenzen der Automatisierbarkeit • Vorstellung des Halteproblems • Unlösbarkeit des Halteproblems	• entwickeln zur Grammatik einer regulären oder kontextfreien Sprache einen zugehörigen endlichen Automaten oder einen Kellerautomaten (M), • modifizieren Grammatiken regulärer und kontextfreier Sprachen (M), • entwickeln zu einer regulären oder kontextfreien Sprache eine Grammatik, die die Sprache erzeugt (M),	*Beispiele:* Klammerausdrücke, a^nb^n im Vergleich zu $(ab)^n$ Halteproblem
4. Entwicklung eines Kellerautomaten als Antwort auf die Grenzen endlicher Automaten • Erweiterung eines DEA um eine einzelne Speichervariable zum Zählen von Eingabezeichen (z.B. Klammern) und Problematisierung dieses Ansatzes • Entwicklung eines Automaten mit Kellerspeicher • Anwendung eines Kellerautomaten zur Syntaxüberprüfung auf Grundlage von nicht-regulären Grammatiken • Implementierung eines Kellerautomaten zur Syntaxüberprüfung (Backtracking)	• stellen endliche Automaten in Tabellen oder Graphen dar und überführen sie in die jeweils andere Darstellungsform (D), • ermitteln die Sprache, die ein endlicher Automat oder ein Kellerautomat akzeptiert (D), • beschreiben an Beispielen den Zusammenhang zwischen Automaten und Grammatiken (D), • entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“, „Teilen und Herrschen“ und „Backtracking“ (M).	

Unterrichtsvorhaben Q2-III:

Thema: von-Neumann-Architektur und Ausführung maschinennaher Programme.

Zeitbedarf: 10 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien oder 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 gespeichert werden kann • Analyse und Erläuterung der Funktionsweise eines einfachen maschinennahen Programms • Übersetzung der bekannten Kontrollstrukturen Verzweigung und Schleife in die Maschinennahe Programmierung	Die Schülerinnen und Schüler • erläutern die Ausführung eines einfachen maschinennahen Programms sowie die Datenspeicherung auf einer „Von-Neumann-Architektur“ (A),	*Material*: Simulationsprogramm Johnny *Beispiel*: Addition zweier Zahlen, Multiplikation durch Addition abbilden

Unterrichtsvorhaben Q2-IV:

Thema: Grundlagen von Netzwerken und Implementierung von protokollbasierten Client-Server-Systemen in Anwendungskontexten

Zeitbedarf: 20 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien oder Materialien
1. Grundlagen den Datenübertragung in Netzwerken • Schichtenmodell: Erarbeitung eines standardisierten Schichtenmodells für Netzwerkkommunikation • Topologien: Erarbeitung und Vergleich ausgewählter Netzwerktopologien • Subnetze und Routing: Analyse von Grundlagen der Adressierung in IP-Netzwerken • Protokolle: Erarbeitung des beispielhaften Aufbaus eines Protokoll auf der Anwendungsschicht	Die Schülerinnen und Schüler • beschreiben und erläutern Netzwerk-Topologien, die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken (A), • analysieren und erläutern Algorithmen und Programme (A), • analysieren und erläutern Protokolle zur Kommunikation in einem Client-Server-Netzwerk (A), • entwickeln und erweitern Protokolle zur Kommunikation in einem Client-Server-Netzwerk (M), • modifizieren Algorithmen und Programme (I) • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihr Operationen und Beziehungen (M), • modellieren Klassen mit ihren Attributen, Methoden und Assoziationsbeziehungen unter Angabe von Multiplizitäten (M),	Alle Inhalte werden an der Simualtionssoftware FILIUS erarbeitet
2. Analyse, Modellierung und Implementierung von Netzwerkanwendungen in Client-Server-Struktur • Nutzung einfacher Server-Dienste mittels der Klassen Connection und Client • Modellierung und Implementierung von Clients für einfach Serverdienste		*Beispiel: Echo- bzw. Time-Clients und –Server sowie eigene Erweiterungen* Anhand der Echo- und Time-Dienste (z.B. lokal im Schulnetz durch den Lehrenden zur Verfügung gestellt) erarbeiten die Schülerinnen und Schüler zunächst den die Funktionsweise bzw. den Aufbau einfacher Clients und

• Anbieten von Diensten mittels der Klasse Server
• Analyse vorgegebener Implementationen einfacher Server
• Modellierung und Implementierung eigener Server
 	
• Entwicklung eines vollständigen Client-Server-Systems
  • Protokollentwurf, Dialogorientierung
  • Bedeutung von Nebenläufigkeit
  • Implementierung
	• ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare und nichtlineare Datensammlungen zu (M),
	• analysieren und erläutern objektorientierte Modellierungen (A),
	• stellen Klassen und ihre Beziehungen grafisch dar (D),
	• implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I).
		verwenden dabei zunächst die Klasse Connection, später die Klasse Client.
		
		In einem zweiten Schritt implementieren die Schülerinnen und Schüler Daytime- und Echo-Client bzw. Erweiterungen/Abwandlungen derselben (ggf. mit Steigerung des Interaktionsgrades) selbst.
		
		Beispiel: Messenger-Dienst
		
		Abschließend entwickeln die Schülerinnen und Schüler ein Client-Server-System zum Versenden von Nachrichten zwischen einzelnen Rechnern (einfacher Messenger), basierend auf selbst gewählten „Nicknames“.

Unterrichtsvorhaben Q2-V:

Thema: Entwicklung eines Netzwerkspiels unter Verwendung eines Vorgehensmodells aus der Softwareentwicklung

Zeitbedarf: 20 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien oder Materialien
1. Projekteinstieg / -planung (Spielauswahl, Zeitmanagement) **2. Projektumsetzung (informatische Analyse, Modellierung, Implementierung, Test (ggf. in Zyklen))** • Analyse des Spiels mit dem Ziel einer späteren informatischen Umsetzung als Netzwerkspiel • Modellierung und Implementierung des Spiels als Netzwerkanwendung • Entwurf eines Protokolls zur Kommunikation zwischen Spielserver und -client basierend auf den erarbeiteten Spielregeln • Entwicklung von Entwurfs- und Implementationsdiagrammen für den Spielserver • Implementation und Test der Spielserver-Klasse und von dieser benötigter Fachklassen • Modellierung, Implementierung und Test des Spielclients • Test der Zusammenarbeit von Spielserver und Spielclient • Reflexion des Softwareprodukts	Die Schülerinnen und Schüler • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M), • verwenden bei der Modellierung geeigneter Problemstellungen Möglichkeiten der Polymorphie (M), • ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M), • stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D), • analysieren und erläutern objektorientierte Modellierungen (A),	Das umzusetzende Spiel wird mit dem Kurs abgesprochen.

3. Projektreflexion (Reflexion der Projektplanung, Präsentation)	• implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), • analysieren und erläutern Algorithmen und Programme (A), • modifizieren Algorithmen und Programme (I), • entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“ und „Teilen und Herrschen“ und „Backtracking“ (M), • implementieren iterative und rekursive Al gorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I). • analysieren und erläutern Algorithmen und Methoden zur Client-Server-Kommunikation (A), • entwickeln und implementieren Algorithmen und Methoden zur Client-Server-Kommunikation (I). • beschreiben und erläutern Netzwerk-Topologien, die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken (A), • analysieren und erläutern Protokolle zur Kommunikation in einem Client-Server-Netzwerk (A), • entwickeln und erweitern Protokolle zur Kommunikation in einem Client-Server-Netzwerk (M).	

Unterrichtsvorhaben Q2-VI:

Wiederholung und Vertiefung ausgewählter Kompetenzen und Inhalte des ersten Jahrs der Qualifikationsphase

2.2 Grundsätze der fachmethodischen und fachdidaktischen Arbeit

In Absprache mit der Lehrerkonferenz sowie unter Berücksichtigung des Schulprogramms hat die Fachkonferenz Informatik die folgenden fachmethodischen und fachdidaktischen Grundsätze beschlossen. In diesem Zusammenhang beziehen sich die Grundsätze 1 bis 15 auf fächerübergreifende Aspekte, die auch Gegenstand der Qualitätsanalyse sind, die Grundsätze 16 bis 26 sind fachspezifisch angelegt.

Überfachliche Grundsätze:

1) Geeignete Problemstellungen zeichnen die Ziele des Unterrichts vor und bestimmen die Struktur der Lernprozesse.
2) Inhalt und Anforderungsniveau des Unterrichts entsprechen dem Leistungsvermögen der Schüler/innen.
3) Die Unterrichtsgestaltung ist auf die Ziele und Inhalte abgestimmt.
4) Medien und Arbeitsmittel sind schülernah gewählt.
5) Die Schülerinnen und Schüler erreichen einen Lernzuwachs.
6) Der Unterricht fördert und fordert eine aktive Teilnahme der Schülerinnen und Schüler.
7) Der Unterricht fördert die Zusammenarbeit zwischen den Schülerinnen und Schüler und bietet ihnen Möglichkeiten zu eigenen Lösungen.
8) Der Unterricht berücksichtigt die individuellen Lernwege der einzelnen Schülerinnen und Schüler.
9) Die Schülerinnen und Schüler erhalten Gelegenheit zu selbstständiger Arbeit und werden dabei unterstützt.
10) Der Unterricht fördert strukturierte und funktionale Einzel-, Partner- bzw. Gruppenarbeit sowie Arbeit in kooperativen Lernformen.
11) Der Unterricht fördert strukturierte und funktionale Arbeit im Plenum.
12) Die Lernumgebung ist vorbereitet; der Ordnungsrahmen wird eingehalten.
13) Die Lehr- und Lernzeit wird intensiv für Unterrichtszwecke genutzt.
14) Es herrscht ein positives pädagogisches Klima im Unterricht.
15) Wertschätzende Rückmeldungen prägen die Bewertungskultur und den Umgang mit Schülerinnen und Schülern.

Fachliche Grundsätze:

16) Im Unterricht werden fehlerhafte Schülerbeiträge produktiv im Sinne einer Förderung des Lernfortschritts der gesamten Lerngruppe aufgenommen.
17) Der Unterricht ermutigt die Lernenden dazu, auch fachlich unvollständige Gedanken zu äußern und zur Diskussion zu stellen.
18) Die Bereitschaft zu problemlösenden Arbeiten wird durch Ermutigungen und Tipps gefördert und unterstützt.
19) Es wird genügend Zeit eingeplant, in der sich die Lernenden neues Wissen aktiv konstruieren und in der sie angemessene Grundvorstellungen zu neuen Begriffen entwickeln können.

20) Durch regelmäßiges wiederholendes Üben werden grundlegende Fertigkeiten reaktiviert.
21) Im Unterricht werden an geeigneter Stelle differenzierende Aufgaben eingesetzt.
22) Die Lernenden werden zu regelmäßiger, sorgfältiger und vollständiger Dokumentation und Speicherung der von ihnen bearbeiteten Aufgaben angehalten.
23) Im Unterricht wird auf einen angemessenen Umgang mit fachsprachlichen Elementen geachtet.
24) Das Vorhandensein eines privaten Rechners wird nicht vorausgesetzt und eine Bearbeitung von programmierbasierenden Hausaufgaben dementsprechend nicht gefordert.

2.3 Grundsätze der Leistungsbewertung und Leistungsrückmeldung

Auf der Grundlage von § 48 SchulG, § 13 APO-GOSt sowie Kapitel 3 des Kernlehrplans Informatik hat die Fachkonferenz im Einklang mit dem entsprechenden schulbezogenen Konzept die nachfolgenden Grundsätze zur Leistungsbewertung und Leistungsrückmeldung beschlossen. Die nachfolgenden Absprachen stellen die Minimalanforderungen an das lerngruppenübergreifende gemeinsame Handeln der Fachgruppenmitglieder dar. Bezogen auf die einzelne Lerngruppe kommen ergänzend weitere der in den Folgeabschnitten genannten Instrumente der Leistungsüberprüfung zum Einsatz.

Verbindliche Absprachen:

Klausuren:

- Klausuren können nach entsprechender Wiederholung im Unterricht auch Aufgabenteile enthalten, die Kompetenzen aus weiter zurückliegenden Unterrichtsvorhaben oder übergreifende prozessbezogene Kompetenzen erfordern.
- Alle Klausuren enthalten nach Möglichkeit auch Aufgaben mit Anforderungen im Sinne des Anforderungsbereiches III.
- Für die Aufgabenstellung der Klausuraufgaben werden die Operatoren der Aufgaben des Zentralabiturs verwendet. Diese sind mit den Schülerinnen und Schülern zu besprechen.
- Die Korrektur und Bewertung der Klausuren erfolgt anhand eines kriterienorientierten Bewertungsbogens, den die Schülerinnen und Schüler als Rückmeldung erhalten.

Im Unterricht:

- Schülerinnen und Schülern wird Gelegenheit gegeben, Projekte in Gruppen oder alleine durchzuführen und selbstständig vorzutragen.
- Sofern schriftliche Übungen (20 Minuten als Kompetenzüberprüfung bezüglich des unmittelbar zurückliegenden Unterrichtsvorhabens) gestellt werden sollen, verständigen sich dazu die Fachlehrkräfte paralleler Kurse und verfahren in diesen gleichartig.
- Es sollen möglichst eine Vielzahl verschiedener Überprüfungsformen angeboten werden, die vorher ausreichend eingeführt werden, unter anderem:

    1. Überprüfungsform I Analyse und Eingrenzung einer kontextbezogenen Problemstellung und Entwicklung eines Modells oder Teilmodells mit erläuternden Begründungen der Entwurfsentscheidungen
    2. Überprüfungsform II Analyse, Erläuterung und Modifikation eines vorgegebenen informatischen Modells sowie die vergleichende Beurteilung unterschiedlicher Entwürfe
    3. Überprüfungsform III Vollständige oder teilweise Implementation einer bereits modellierten Problemstellung
    4. Überprüfungsform IV Entwurf und formale Darstellung von Algorithmen zu einer vorgegebenen informatischen Problemstellung
    5. Überprüfungsform V Analyse und Erläuterung von vorgegebenen Algorithmen oder Programmausschnitten

6. Überprüfungsform VI Interpretation gegebener textueller, grafischer oder formaler Darstellungen informatischer Zusammenhänge und deren Überführung in eine andere Darstellungsform
7. Überprüfungsform VII Darstellung, Erläuterung und sachgerechte Anwendung von informatischen Begriffen, Verfahren und Lösungsstrategien
8. Überprüfungsform VIII Analyse und Beurteilung einer Problemlösung oder eines Informatiksystems nach vorgegebenen oder eigenen Kriterien
9. Überprüfungsform IX Analyse und Bewertung des Einsatzes eines Informatiksystems in Bezug auf ethische, rechtliche oder gesellschaftliche Fragestellungen

Verbindliche Instrumente:

Überprüfung der schriftlichen Leistung
Einführungsphase: Zwei Klausuren je Halbjahr

Dauer der Klausuren: 2 Unterrichtsstunden

Qualifikationsphase 1: Zwei Klausuren je Halbjahr

Dauer der Klausuren: 100 Minuten

Qualifikationsphase 2: Zwei Klausuren im ersten Halbjahr,
eine Klausur im zweiten Halbjahr.

Dauer der Klausuren: Q2.1 150 Minuten

Q2.2 225 Minuten

Überprüfung der sonstigen Leistung
In die Bewertung der sonstigen Mitarbeit fließen folgende Aspekte ein, die den Schülerinnen und Schülern bekanntgegeben werden müssen:

- Beteiligung am Unterrichtsgespräch (Quantität und Kontinuität)
- Qualität der Beiträge (inhaltlich und methodisch)
- Eingehen auf Beiträge und Argumentationen von Mitschülerinnen und -schülern, Unterstützung von Mitlernenden
- Umgang mit neuen Problemen, Beteiligung bei der Suche nach neuen Lösungswegen
- Selbstständigkeit im Umgang mit der Arbeit
- Umgang mit Arbeitsaufträgen (Hausaufgaben, Unterrichtsaufgaben…)
- Anstrengungsbereitschaft und Konzentration auf die Arbeit
- Beteiligung während kooperativer Arbeitsphasen
- Darstellungsleistung bei Referaten oder Plakaten und beim Vortrag von Lösungswegen
- Ergebnisse schriftlicher Übungen
- Anfertigen zusätzlicher Arbeiten, z.B. eigenständige Ausarbeitungen im Rahmen binnendifferenzierender Maßnahmen, Erstellung von Computerprogrammen

2.4 Lehr- und Lernmittel

Die Fachkonferenz Informatik hat beschlossen, das digitale Lehrwerk „Informatik 1“ bzw. „Informatik 2“ der Bibox von Westermann sowohl in der Einführungsphase als auch in der Qualifikationsphase zu nutzen.

3 Entscheidungen zu fach- und unterrichtsübergreifenden Fragen

Die Fachkonferenz Informatik hat sich im Rahmen des Schulprogramms und in Absprache mit den betreffenden Fachkonferenzen auf folgende, zentrale Schwerpunkte geeinigt.

Wettbewerbe

Alle Klassen der 5 bis 13 der Euregio-Gesamtschule sollen an dem Informatik-Biber-Wettbewerb teilnehmen. Die Teilnahme wird von der Fachschaft Informatik angeregt und organisiert. Insbesondere den Schülerinnen und Schülern der Oberstufe soll ermöglicht werden am Jugendwettbewerb oder am Bundeswettbewerb Informatik teilzunehmen.

4 Qualitätssicherung und Evaluation

Durch parallele Klausuren in den Grundkursen, durch Diskussion der Aufgabenstellung von Klausuren in Fachkonferenzen und eine regelmäßige Erörterung der Ergebnisse von Leistungsüberprüfungen wird ein hohes Maß an fachlicher Qualitätssicherung erreicht.

Das schulinterne Curriculum ist zunächst bis 2023 für den ersten Durchgang durch die gymnasiale Oberstufe nach Erlass des Kernlehrplanes verbindlich. Jeweils vor Beginn eines neuen Schuljahres, d.h. erstmalig nach Ende der Einführungsphase im Sommer 2021 werden in einer Sitzung der Fachkonferenz für die nachfolgenden Jahrgänge zwingend erforderlich erscheinende Veränderungen diskutiert und ggf. beschlossen, um erkannten ungünstigen Entscheidungen schnellstmöglich entgegenwirken zu können.