Schulinternes Curriculum Informatik

Die folgenden Kompetenzen aus dem Bereich Kommunizieren und Kooperieren werden in allen Unterrichtsvorhaben aller Stufen 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).

Übersicht über die Unterrichtsvorhaben

<u>Einführungsphase</u>

Planungsgrundlage: 3 Ustd. pro Woche, 40 Wochen, davon 75% entsprechen 90 UStd. pro Schuljahr.

Einführungsphase				
Unterrichts-vorhaben	**Inhaltsfeld** **Inhaltliche Schwerpunkte**	**Schwerpunkte der Kompetenzentwicklung** Die Schülerinnen und Schüler	**Methoden- und Medienkompetenzen**	**Vernetzungen**
EF.1 *Einführung in die Nutzung von Informatiksysteme und in grundlegende Begrifflichkeiten* *3 Ustd.*	*Informaiksysteme* Inhaltliche Schwerpunkte: 1. *Prinzipieller Aufbau und Arbeitsweise eines Rechners* *(a) EVA Prinzip* *(b) von-Neumann-Architektur* 2. *Speichern von Daten mit informatischen Systemen am Beispiel der Schulrechner* *→ Netzwerk*	• beschreiben und erläutern den Aufbau und die Arbeitsweise singulärer Rechner am Beispiel der „Von-NeumannArchitektur“ (A), • nutzen die im Unterricht eingesetzten Informatiksysteme selbstständig, sicher, zielführend und verantwortungsbewusst (D)	*Bestandteile eines Rechners identifizieren und kategorisieren* *Einführung in Nutzung des Schulnetzwerkes*	*Christliche Schwerpunktsetzung:* • *Möglichkeiten zum Fächerübergereifenden Unterricht* • *Berufsorientierung* • *Europaschule* • *„Grüne“ Schule* •

EF.2 *Grundlagen der objektorientierten Analyse, Modellierung und Implementierung anhand von statischen Grafikszenen* *12 Ustd.*	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.* 2. *Analyse von Klassen didaktischer Lernumgebungen* *(a) Objektorientierte Programmierung als modularisiertes Vorgehen* *(b) Teilanalyse der Klassen der didaktischen Lernumgebungen 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* *(d) Methodenaufrufe mit Parameterübergabe zur Manipulation von Objekteigenschaften (z.B. Farbe, Position, Drehung)*	• 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.* *Materialien: Dokumentation der didaktischen Bibliothek SAS* *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 Die Schülerinnen und Schüler bilden das Emblem der olympischen Spiele mit Hilfe von SAS-Objekten nach*	

EF.3 *Grundlagen der objektorientierten Programmierung und algorithmischer Grundstrukturen in Java anhand von einfachen Animationen* *30 Ustd.*	1. *Bewegungsanimationen am Beispiel einfacher grafischer Objekte (GLObjekte)* *(a) Kontinuierliche Verschiebung eines GLObjekts 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 GLObjekten mit Hilfsvariablen* *(e) Meldungen zur Kollision zweier GLObjekte mit Hilfe von Abstandsberechnungen und Verzweigungen (IF-Anweisungen)* 2. *Erstellen und Verwalten größerer Mengen einfacher grafischer Objekte (GLObjekte)* *(a) Erzeugung von Objekten mit Hilfe von Zählschleifen (FOR-Schleife)* *(b) Verwaltung von Objekten in eindimensionalen Feldern (Arrays)* *(c) Animation von Objekten, die in eindimensionalen Feldern (Arrays) verwaltet werden* *(d) Vertiefung: Verschiedene Feldbeispiele*	• analysieren und erläutern einfache Algorithmen und Programme (A), • entwerfen einfache Algorithmen und stellen sie umgangssprachlich und grafisch dar (M), • ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modellieren Klassen mit ihren Attributen, ihren Methoden und Assoziationsbeziehungen (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare Datensammlungen zu (M), • ordnen Klassen, Attributen und Methoden ihren Sichtbarkeitsbereich zu (M), • modifizieren einfache Algorithmen und Programme (I), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), • implementieren Algorithmen unter Verwendung von Variablen und Wertzuweisungen, Kontrollstrukturen sowie Methodenaufrufen (I), • implementieren einfache Algorithmen unter Beachtung der Syntax und Semantik einer Programmiersprache (I), • testen Programme schrittweise anhand von Beispielen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I).	*Projekt: z.B. „Pfeil"-Wurfspiel Die Schülerinnen und Schüler realisieren mit Objekten der SAS-Umgebung ein Spiel, bei dem ein Ball über den Bildschirm bewegt und auf eine runde Zielscheibe geworfen werden soll. Alternativ: z.B. rollende Kugeln* *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 Landemarkierungen werden verschiedene Lauflichter realisiert. Projekt: z.B. Schachbrett Die Schülerinnen und Schüler realisieren mit Hilfe mehrerer Rechtecken ein Schachbrett.* *Projekt: z.B. Kerzensimulation Die Schülerinnen und Schüler modellieren und erstellen*	

Einführungsphase				
Unterrichts-vorhaben	**Inhaltsfeld** **Inhaltliche Schwerpunkte**	**Schwerpunkte der Kompetenzentwicklung** Die Schülerinnen und Schüler	**Methoden- und Medienkompetenzen**	**Vernetzungen**
	3. *Modellierung und Animation komplexerer grafisch repräsentierbarer Objekte* *(a) Modellierung eines Simulationsprogramms mit eigenen Klassen, die sich selbst mit Hilfe von einfachen GLObjekten 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* *(f) Vertiefung: Weitere Projekte*		*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*	

EF.4 *Modellierung und Implementierung von Klassen- und Objektbeziehungen anhand von grafischen Spielen und Simulationen* *30 Ustd.*	*1. Vertiefung des Referenzbegriffs und Einführung des Prinzips der dynamischen Referenzierung* *(a) Einführung der SAS-Objektselektion mit der Maus* *(b) Einführung der Klasse View als 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.* *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 Kollision* *(d) Ergänzung einer Kollisionsabfrage durch zusätzliche Assoziationsbeziehungen in Diagramm, Dokumentation und Quellcode* *(e) Verallgemeinerung der neuen Verwendung von Objektreferenzen*	• analysieren und erläutern eine objektorientierte Modellierung (A), • stellen die Kommunikation zwischen Objekten grafisch dar (M), • ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modellieren Klassen mit ihren Attributen, ihren Methoden und Assoziationsbeziehungen (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare Datensammlungen zu (M), • ordnen Klassen, Attributen und Methoden ihren Sichtbarkeitsbereich zu (M), • modellieren Klassen unter Verwendung von Vererbung (M), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), • testen Programme schrittweise anhand von Beispielen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • modifizieren einfache Algorithmen und Programme (I), • stellen Klassen, Assoziations- und Vererbungsbeziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen durch Beschreibung der Funktionalität der Methoden (D).	*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.* *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 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*	

	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* *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* *(b) Spezialisierung der Klasse zu Unterklassen mit verschiedenen Animationen durch Überschreiben der entsprechenden Animationsmethode* *(c) Reflexion des Prinzips der späten Bindung*		*entwickeln ein Autospiel, bei dem ein Auto durch einen Wald fahren und mit Bäumen kollidieren kann.* *Projekt: z.B. Schneemann Die Schülerinnen und Schüler erstellen eine Simulation von Schneemännern, die unterschiedliche Kopfbedeckungen tragen.* *Projekt: z.B. Flummibälle Die Schülerinnen und Schüler entwickeln eine Simulation von Flummibällen, bei der unterschiedliche Bälle unterschiedliche Bewegungen durchführen. Projekt: z.B. Weihnachtsbaum Die Schülerinnen und Schüler entwickeln eine Simulation eines Weihnachtsbaums mit Hilfe einer abstrakten Klasse Schmuck.*	

EF.5 *Such- und Sortieralgorithmen anhand kontextbezogener Beispiele* *12 Ustd.*	1. *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* 2. *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* 3. *Systematisierung von Algorithmen und Effizienzbetrachtungen* *(a) Formulierung und Erläuterung von mehreren Algorithmen im Pseudocode (Sortieren durch Einfügen, Sortieren durch Vertauschen, Sortieren durch Auswählen)* *(b) Bewertung von Algorithmen anhand der Anzahl der nötigen Vergleiche*	• 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* *Material: z.B.: Kartenspiele* *Projekt: z.B. Bücherregal Schülerinnen und Schüler visualisieren die Abläufe der Sortieralgorithmen*	

Einführungsphase				
Unterrichts-vorhaben	**Inhaltsfeld** **Inhaltliche Schwerpunkte**	**Schwerpunkte der Kompetenzentwicklung** Die Schülerinnen und Schüler	**Methoden- und Medienkompetenzen**	**Vernetzungen**
	(c) Effizienzbetrachtungen an einem konkreten Beispiel bezüglich der Rechenzeit und des Speicherplatzbedarfs			
EF.6 *Digitalisierung und Grundlagen des Datenschutzes* *6 Ustd.*	1. *Binärcodierungen* *(a) natürliche Zahlen,* *(b) Ganzzahlen,* *(c) Zeichen* 2. *Betrachtung des Themas Datenschutz Selbstentdeckendes Erkunden:* *(a) Umfang von Datensammlungen,* *(b) Vorgehen bei Datensammlungen,* *(c) wirtschaftlichen Interessen bei Datensammlungen* *(d) Motivation der Akteure Problematisierung und Anknüpfung an die Lebenswelt der 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)* *Material: Onlinespiel DataDealer*	

Qualifikationsphase GK

Planungsgrundlage: 3 Ustd. pro Woche, 40 Wochen, davon 75% entsprechen 90 UStd. pro Schuljahr.

Qulaifikationsphase Grundkurs				
Unterrichts-vorhaben	**Inhaltsfeld** **Inhaltliche Schwerpunkte**	**Schwerpunkte der Kompetenzentwicklung** Die Schülerinnen und Schüler	**Methoden- und Medienkompetenzen**	**Vernetzungen**
Q1.1 *Kryptographie* *? Ustd.*	1. *Erarbeitung und Implementierung von klassischen symmetrischen Verschlüsselungsverfahren.* *(a) Zeichenkettenoperationen* *(b) GUI-Entwicklung mit dem Java-Editor* *(c) ASCII-Codierung* *(d) symmetrische Verschlüsselung* 2. *Erarbeitung von asymmetrischen Verschlüsselungsverfahren.*	• 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*	

Q1.2 *Modellierung und Implementierung von Anwendungen mit dynamischen, linearen Datenstrukturen* *? Ustd.*	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.* 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* 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*	• 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), • 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), • 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	*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*	

	(b) Modellierung und Implementierung einer kontextbezogenen Anwendung unter Verwendung der Klasse List. 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.*	Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • 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)	*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.* *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.* *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. *Mögliches Beispiel: Skispringen Ein Skispringer 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*	

Qulaifikationsphase Grundkurs				
Unterrichts-vorhaben	**Inhaltsfeld** **Inhaltliche Schwerpunkte**	**Schwerpunkte der Kompetenzentwicklung** Die Schülerinnen und Schüler	**Methoden- und Medienkompetenzen**	**Vernetzungen**
			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.	

Qulaifikationsphase Grundkurs				
Unterrichts-vorhaben	**Inhaltsfeld** **Inhaltliche Schwerpunkte**	**Schwerpunkte der Kompetenzentwicklung** Die Schülerinnen und Schüler	**Methoden- und Medienkompetenzen**	**Vernetzungen**
Q1.3 *von-Neumann-Architektur und Ausführung maschinennaher Programme.* *? Ustd.*	1. *Von-Neumann-Architektur und die Ausführung maschinennaher Programme* *(a) prinzipieller Aufbau einer von Neumann-Architektur mit CPU, Rechenwerk, Steuerwerk, Register und Hauptspeicher* *(b) einige maschinennahe Befehlen und ihre Repräsentation in einem BinärCode, der in einem Register gespeichert werden kann* *(c) Analyse und Erläuterung der Funktionsweise eines einfachen maschinennahen Programms*	• 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.*	

Q1.4 *Suchen und Sortieren auf linearen Datenstrukturen* *? Ustd.*	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)* 2. *Sortieren in Listen und Arrays - Entwicklung und Implementierung von iterativen und rekursiven Sortierverfahren* *(a) Entwicklung und Implementierung eines einfachen Sortierverfahrens für eine Liste* *(b) Implementierung eines einfachen Sortierverfahrens für ein Feld* *(c) 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 Speicherbedarfs bei beiden Sortierverfahren*	• 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), • 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).	*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.*	

Qulaifikationsphase Grundkurs				
Unterrichts-vorhaben	**Inhaltsfeld** **Inhaltliche Schwerpunkte**	**Schwerpunkte der Kompetenzentwicklung** Die Schülerinnen und Schüler	**Methoden- und Medienkompetenzen**	**Vernetzungen**
	(c) Beurteilung der Effizienz der beiden Sortierverfahren			

Q1.5 *Modellierung und Nutzung von relationalen Datenbanken in Anwendungskontexten* *? Ustd.*	*1. Nutzung von relationalen Datenbanken* *(a) 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* *(b) 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)* *(c) Vertiefung an einem weiteren Datenbankbeispiel* *2. Modellierung von relationalen Datenbanken* *(a) Entity-Relationship-Diagramm*	• erläutern die Eigenschaften und den Aufbau von Datenbanksystemen unter dem Aspekt der sicheren Nutzung (A), • analysieren und erläutern die Syntax und Semantik einer Datenbankabfrage (A), • analysieren und erläutern eine Datenbankmodellierung (A), • erläutern die Eigenschaften normalisierter Datenbankschemata (A), • bestimmen Primär- und Sekundärschlüssel (M), • ermitteln für anwendungsbezogene Problemstellungen Entitäten, zugehörige Attribute, Relationen und Kardinalitäten (M), • modifizieren eine Datenbankmodellierung (M), • modellieren zu einem Entity-Relationship-Diagramm ein relationales Datenbankschema (M), • bestimmen Primär- und Sekundärschlüssel (M), • überführen Datenbankschemata in vorgegebene Normalformen (M), • verwenden die Syntax und Semantik einer Datenbankabfragesprache, um Informationen aus einem 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),	*Beispiel: Fehlstundendatenbank* *Beispiel: Präsidentendatenbank* *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.* *Auszug aus dem Bundesdatenschutzgeset z Volkszählungsurteil passende Fallbeispiele*	

	• *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)* 3. *Fallbeispiele zum Datenschutz*	• 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 gesellschaftlicher Werte unter Berücksichtigung unterschiedlicher Interessenlagen (A), • nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zum Erschließen, zur Aufbereitung und Präsentation fachlicher Inhalte (D)		
Q2.1 *Modellierung und Implementierun*	*1. Analyse von Baumstrukturen in verschiedenen Kontexten*	• erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), • analysieren und erläutern Algorithmen und Programme (A),	*Beispiel: Morsebaum Das Morsealphabet wird anhand des Kriteriums Punkt (links) und Strich*	

g von Anwendungen mit dynamischen, nichtlinearen Datenstrukturen *? Ustd.*	*(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* *(e) Traversierung eines Binärbaums im Pre-, In- und Postorderdurchlauf* 3. *Die Datenstruktur binärer Suchbaum im Anwendungskontext unter Verwendung der Klasse BinarySearchTree* *(a) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen*	• beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M), • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M), • verwenden bei der Modellierung geeigneter Problemstellungen die Möglichkeiten der Polymorphie (M), • entwickeln iterative und rekursive Algorithmen unter Nutzung der Konstruktionsstrategien „Modularisierung“ und „Teilen und Herrschen“ (M), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • modifizieren Algorithmen und Programme (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme systematisch anhand von Beispielen (I), • stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D),	*(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. 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.* *Mögliches Beispiel: Suchbaum mit Ganzzahlen Mögliches*	

Qulaifikationsphase Grundkurs				
Unterrichts-vorhaben	**Inhaltsfeld** **Inhaltliche Schwerpunkte**	**Schwerpunkte der Kompetenzentwicklung** Die Schülerinnen und Schüler	**Methoden- und Medienkompetenzen**	**Vernetzungen**
	(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* 4. *Übung und Vertiefungen der Verwendung von Binärbäumen oder binären Suchbäumen anhand weiterer Problemstellungen*	• stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D).	*Beispiel: Informatikerbaum als Suchbaum In einem binären Suchbaum werden die Namen und die Geburtsdaten von Informatikern lexikographisch geordnet abgespeichert.* *Mögliche Beispiel: Codierungsbäume oder Huffman-Codierung mögliches Beispiel: Buchindex*	

Q2.2 *Endliche Automaten und formale Sprachen* *? Ustd.*	*1 Endliche Automaten* *(a) Vom Automaten in den Schülerinnen und Schülern bekannten Kontexten zur formalen Beschreibung eines endlichen Automaten* *(b) Untersuchung, Darstellung und Entwicklung endlicher Automaten* 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* 3 *Grenzen endlicher Automaten und 4. Grenzen der Automatisierbarkeit* *(a) Vorstellung des Halteproblems* *(b) Unlösbarkeit des Halteproblems* *(c) Beurteilung des Einsatzes von Informatiksystemen hinsichtlich prinzipieller Möglichkeiten und prinzipieller Grenzen*	• analysieren und erläutern die Eigenschaften endlicher Automaten einschließlich ihres Verhaltens auf bestimmte Eingaben (A), • analysieren und erläutern Grammatiken regulärer Sprachen (A), • zeigen die Grenzen endlicher Automaten und regulärer Grammatiken im Anwendungszusammenhang auf (A), • ermitteln die formale Sprache, die durch eine Grammatik erzeugt wird (A), • entwickeln und modifizieren zu einer Problemstellung endliche Automaten (M), • entwickeln und modifizieren zu einer Problemstellung endliche Automaten (M), • entwickeln zur akzeptierten Sprache eines Automaten die zugehörige Grammatik (M), • entwickeln zur Grammatik einer regulären Sprache einen zugehörigen endlichen Automaten (M), • modifizieren Grammatiken regulärer Sprachen (M), • entwickeln zu einer regulären Sprache eine Grammatik, die die Sprache erzeugt (M), • stellen endliche Automaten in Tabellen oder Graphen dar und überführen sie in die 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).	*Mögliche Beispiele: Fair-Trade-Autoat, Cola-Automat, Videorekorder, Akzeptor für bestimmte Zahlen, Akzeptor für Teilwörter in längeren Zeichenketten, Akzeptor für Terme* *Mögliche Beispiele: reguläre Grammatik für Wörter mit ungerader Parität, Grammatik für Wörter, die bestimmte Zahlen repräsentieren, Satzgliederungsgrammatik* *Beispiele: Klammerausdrücke,* $a^n b^n$ *im Vergleich zu* $(ab)^n$ *Halteproblem*	

Qulaifikationsphase Grundkurs				
Unterrichts-vorhaben	**Inhaltsfeld** **Inhaltliche Schwerpunkte**	**Schwerpunkte der Kompetenzentwicklung** Die Schülerinnen und Schüler	**Methoden- und Medienkompetenzen**	**Vernetzungen**
Q2.3 *Netzstrukturen* *Ustd.*	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*	• beschreiben und erläutern Topologien, die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken (A),	*Material: FILIUS*	

Qulaifikationsphase Grundkurs				
Unterrichts-vorhaben	**Inhaltsfeld** **Inhaltliche Schwerpunkte**	**Schwerpunkte der Kompetenzentwicklung** Die Schülerinnen und Schüler	**Methoden- und Medienkompetenzen**	**Vernetzungen**
Q2.4 *Wiederholung und Vertiefung ausgewählter Kompetenzen und Inhalte des ersten Jahres der Qualifikationsphase*		•		

Qualifikationsphase LK

Planungsgrundlage: 5 Ustd. pro Woche, 40 Wochen, davon 75% entsprechen 150 UStd. pro Schuljahr.

Qulaifikationsphase Leistungskurs				
Unterrichts-vorhaben	**Inhaltsfeld** **Inhaltliche Schwerpunkte**	**Schwerpunkte der Kompetenzentwicklung** Die Schülerinnen und Schüler	**Methoden- und Medienkompetenzen**	**Vernetzungen**
Q1.1 *Kryptographie* *20 Ustd.*	1. *Erarbeitung und Implementierung von klassischen symmetrischen Verschlüsselungsverfahren.* *(a) Zeichenkettenoperationen* *(b) GUI-Entwicklung mit dem Java-Editor* *(c) ASCII-Codierung* *(d) symmetrische Verschlüsselung* 2. *Erarbeitung von asymmetrischen Verschlüsselungsverfahren.*	• 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*	

Q1.2 *Modellierung und Implementierung dynamischer linearer Datenstrukturen und Implementierung von Anwendungen unter Verwendung dynamischer, linearer Datenstrukturen* *30 Ustd.*	*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.* *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* *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*	• 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 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),	*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*	

	(b) Modellierung und Implementierung einer kontextbezogenen Anwendung unter Verwendung der Klasse List. *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.*	• 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.* *Beispiel: PostFixRechner Es soll ein Taschenrechner mit GUI erstellt werden, der nach dem Postfix-Prinzip mit Hilfe eines Stapels arbeitet. Beispiel: Integeraddition Ganzzahlen außerhalb des durch den Datentyp int abgedeckten Bereichs sollen mit Hilfe von Stapeln addiert werden.* *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. *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*	

Qulaifikationsphase Leistungskurs				
Unterrichts-vorhaben	**Inhaltsfeld** **Inhaltliche Schwerpunkte**	**Schwerpunkte der Kompetenzentwicklung** Die Schülerinnen und Schüler	**Methoden- und Medienkompetenzen**	**Vernetzungen**
			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.	

Q1.3 *Suchen und Sortieren auf linearen Datenstrukturen* *15 Ustd.*	*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)* *2. Sortieren in Listen und Arrays - Entwicklung und Implementierung von iterativen und rekursiven Sortierverfahren* *(a) Entwicklung und Implementierung eines einfachen Sortierverfahrens für eine Liste* *(b) Implementierung eines einfachen Sortierverfahrens für ein Feld* *(c) 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*	• 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“ , „Teilen und Herrschen“ und „Backtracking“(M), • modifizieren Algorithmen und Programme (I), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen 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 mit Hilfe von Testanwendungen (I), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D).	*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.*	

Qulaifikationsphase Leistungskurs				
Unterrichts-vorhaben	**Inhaltsfeld** **Inhaltliche Schwerpunkte**	**Schwerpunkte der Kompetenzentwicklung** Die Schülerinnen und Schüler	**Methoden- und Medienkompetenzen**	**Vernetzungen**
	(c) Beurteilung der Effizienz der beiden Sortierverfahren			

Q1.4 *Modellierung und Implementierung von Baumstrukturen und Implementierung von Anwendungen unter Verwendung von Baumstrukturen* *20 Ustd.*	*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* *(e) Traversierung eines Binärbaums im Pre-, In- und Postorderdurchlauf* *3. Die Datenstruktur binärer Suchbaum im Anwendungskontext unter Verwendung der Klasse BinarySearchTree*	• 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), • 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),	*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. 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,*	

Qulaifikationsphase Leistungskurs				
Unterrichts-vorhaben	**Inhaltsfeld** **Inhaltliche Schwerpunkte**	**Schwerpunkte der Kompetenzentwicklung** Die Schülerinnen und Schüler	**Methoden- und Medienkompetenzen**	**Vernetzungen**
	(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* *4. Übung und Vertiefungen der Verwendung von Binärbäumen oder binären Suchbäumen anhand weiterer Problemstellungen*	• 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), • 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).	*stehen im rechten Teilbaum.* *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.* *Mögliche Beispiel: Codierungsbäume oder Huffman-Codierung mögliches Beispiel: Buchindex*	

Q1.5 *Modellierung, Implementierung und Nutzung von relationalen Datenbanken in Anwendungskontexten* *30 Ustd.*	*1. Nutzung von relationalen Datenbanken* *(a) 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* *(b) 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)* *(c) Vertiefung an einem weiteren Datenbankbeispiel* *2. Modellierung von relationalen Datenbanken* *(a) Entity-Relationship-Diagramm*	• erläutern die Eigenschaften und den Aufbau von Datenbanksystemen unter dem Aspekt der sicheren Nutzung (A), • analysieren und erläutern die Syntax und Semantik einer Datenbankabfrage (A), • analysieren und erläutern eine Datenbankmodellierung (A), • erläutern die Eigenschaften normalisierter Datenbankschemata (A), • bestimmen Primär- und Sekundärschlüssel (M), • ermitteln für anwendungsbezogene Problemstellungen Entitäten, zugehörige Attribute, Relationen und Kardinalitäten (M), • modifizieren eine Datenbankmodellierung (M), • modellieren zu einem Entity-Relationship-Diagramm ein relationales Datenbankschema (M), • 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 einem Datenbanksystem zu extrahieren (I), • ermitteln Ergebnisse von Datenbankabfragen über mehrere verknüpften Tabellen (D), • stellen Entitäten mit ihren Attributen und die Beziehungen zwischen Entitäten in einem	*Beispiel: Fehlstundendatenbank* *Beispiel: Präsidentendatenbank* *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.* *Auszug aus dem Bundesdatenschutzgeset z Volkszählungsurteil passende Fallbeispiele*	

Qulaifikationsphase Leistungskurs				
Unterrichts-vorhaben	**Inhaltsfeld** **Inhaltliche Schwerpunkte**	**Schwerpunkte der Kompetenzentwicklung** Die Schülerinnen und Schüler	**Methoden- und Medienkompetenzen**	**Vernetzungen**
	• *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)* *3. Fallbeispiele zum Datenschutz*	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 gesellschaftlicher Werte unter Berücksichtigung unterschiedlicher Interessenlagen (A), • nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zum Erschließen, zur Aufbereitung und Präsentation fachlicher Inhalte (D).		

Qulaifikationsphase Leistungskurs				
Unterrichts-vorhaben	**Inhaltsfeld** **Inhaltliche Schwerpunkte**	**Schwerpunkte der Kompetenzentwicklung** Die Schülerinnen und Schüler	**Methoden- und Medienkompetenzen**	**Vernetzungen**
Q1.6 *Informatik, Mensch und Gesellschaft - Sicherheit und Datenschutz in Informatiksysteme n, Auswirkungen und Grenzen der Automatisierun g* *10 Ustd.*	*1. Einführung in die Problemfelder Datenschutz, Urheberrecht und moralische Verantwortung* *(a) Vorstellung eines komplexen Fallbeispiels* *(b) Erarbeitung von Interesse verschiedener Interessensgruppen im Hinblick auf die Problemfelder* *(c) Erster Bewertungsversuch auf Grundlage des Vorwissens von Schülerinnen und Schülern* *2. Erarbeitung grundlegender Positionen (ggf. in zieldifferenten Gruppen) und Anwendung auf Fallbeispiele* *(a) Erarbeitung von Grundideen des Datenschutzes (z.B. personenbezogene Daten, informationelle Selbstbestimmung, Datensparsamkeit usw.)* *(b) Erarbeitung von Grundideen des Urheberrechts anhand eines verbreiteten Lizenzsystems* *(c) Erarbeitung eines einfachen moralischen Bewertungsmaßstabes* *(d) Anwendung auf Fallbeispiele*	• 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.*	

Q1.7 *Projektorientierte Softwareentwicklung am Beispiel einer Anwendung mit Datenbankanbindung* *15 Ustd.*	*1. Analyse einer lebensweltnahen Problemstellung im Hinblick auf die Entwicklung eines Java-Programms mit Datenbankanbindung* *(a) Entwicklung einer Programmidee* *(b) Analyse des Problembereichs* *(c) Entwicklung eines Anforderungskatalogs für das zu entwickelnde Programm* *2. Modellierung einer datenbankgestützten Problemlösung unter Berücksichtigung des MVC-Prinzips* *(a) Strukturierung nach dem MVC-Prinzip* *(b) Modellierung der Datenbank (ER-Diagramm, Datenbankschema)* *(c) Modellierung der Kontrollklassen und Entwicklung von SQL-Anweisungen entsprechend des Anforderungskatalogs* *(d) Modellierung einer grafischen Benutzungsoberfläche* *3. Umsetzung des Softwareprojektes* *(a) Umsetzung der Datenbank in einem Datenbankmanagementsystem* *(b) Implementierung der Kontrollklassen mit Anbindung an*	• 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), • 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-Relations - hip-Diagramm grafisch dar (D), • modellieren zu einem Entity-Relations - hip-Diagramm ein relationales Datenbankschema (M),	*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* *Sportfestverwaltung* *Verwaltung von Aufgaben, Sportereignissen und Ergebnissen des Schulsportfestes* *Materialverwaltung* *Materialien für Vertretungsstunden sollen verwaltet werden.*	

	die Datenbank unter Verwendung didaktischer Klassen *(c) Integration in den Prototypen der Benutzeroberfläche*	• bestimmen Primär- und Sekundärschlüssel (M), • implementieren ein relationales Datenbankschema als Datenbank (I), • analysieren und erläutern eine Datenbankmodellierung (A), • erläutern die Eigenschaften normalisierter Datenbankschemata (A), • überprüfen Datenbankschemata auf vor - gegebene 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 verknüpften 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		

Qulaifikationsphase Leistungskurs				
Unterrichts-vorhaben	**Inhaltsfeld** **Inhaltliche Schwerpunkte**	**Schwerpunkte der Kompetenzentwicklung** Die Schülerinnen und Schüler	**Methoden- und Medienkompetenzen**	**Vernetzungen**
		aus einem Datenbanksystem zu extrahieren (I), • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Daten, zur Organisation von Arbeitsabläufen so - wie 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).		

Q2.1 *Modellierung und Implementierung von Graphen und Implementierung von Algorithmen auf Graphen in Anwendungssituationen* *25 Ustd.*	*1. Analyse von Graphen in verschiedenen Kontexten* *(a) Grundlegende Begriffe (Graph, gerichtet – ungerichtet, Knoten, Kanten, Kantengewicht)* *(b) Aufbau und Darstellung von Graphen anhand von Graphenstrukturen in verschiedenen Kontexten (Adjazenzmatrix, Adjazenzliste)* *2. Die Datenstruktur Graph im Anwendungskontext unter Nutzung der Klassen Graph, Vertex und Edge.* *(a) Erarbeitung der Klassen Graph, Vertex und Edge und beispielhafte Anwendung der Operationen* *(b) Bestimmung von Wegen in Graphen im Anwendungskontext (Tiefensuche, Breitensuche)* *(c) Bestimmung von kürzesten Wegen in Graphen im Anwendungskontext (Backtracking, Dijkstra).* *(d) Bestimmung von minimalen Spannbäumen eines Graphen im Anwendungskontext mithilfe des Prim- oder des Kruskal-Algorithmus*	• 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), • 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),	*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.* *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 Beispiel: Navigationsgraph Ausgehend von dem Problem der Suche eines beliebigen Weges zwischen zwei Knoten in*	

		• 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), • 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),	*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*	

Qulaifikationsphase Leistungskurs				
Unterrichts-vorhaben	**Inhaltsfeld** **Inhaltliche Schwerpunkte**	**Schwerpunkte der Kompetenzentwicklung** Die Schülerinnen und Schüler	**Methoden- und Medienkompetenzen**	**Vernetzungen**
			von der Lerngruppe erarbeitet und implementiert. In leistungsstarken Lerngruppen können auch beide Algorithmen behandelt werden. So kann deutlich werden, dass es u.U. zu einem Graphen mehrere minimale Spannbäume gibt.	

Q2.2 *Endliche Automaten und formale Sprachen sowie die Entwicklung eines Parsers zu einer formalen Sprache* *30 Ustd.*	*1. Endliche Automaten* *(a) Vom Automaten in den Schülerinnen und Schülern bekannten Kontexten zur formalen Beschreibung eines endlichen Automaten* *(b) Untersuchung, Darstellung und Entwicklung endlicher Automaten* *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.* *(e) Entwicklung eines Parsers für eine einfache reguläre Sprache.* *3. Grenzen endlicher Automaten und Grenzen der Automatisierbarkeit* *(a) Vorstellung des Halteproblems* *(b) Unlösbarkeit des Halteproblems* *4. Entwicklung eines Kellerautomaten als Antwort auf die Grenzen endlicher Automaten* *(a) Erweiterung eines DEA um eine einzelne Speichervariable zum*	• 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), • 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), • 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), • 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),	*Mögliche Beispiele: Cola-Automat, Videorekorder, Akzeptor für bestimmte Zahlen, Akzeptor für Teilwörter in längeren Zeichenketten, Akzeptor für Terme* *Mögliche Beispiele: reguläre Grammatik für Wörter mit ungerader Parität, Grammatik für Wörter, die bestimmte Zahlen repräsentieren, Satzgliederungsgrammatik* *Beispiele: Klammerausdrücke,* $a^n b^n$ *im Vergleich zu* $(ab)^n$ *Halteproblem*	

Qulaifikationsphase Leistungskurs				
Unterrichts-vorhaben	**Inhaltsfeld** **Inhaltliche Schwerpunkte**	**Schwerpunkte der Kompetenzentwicklung** Die Schülerinnen und Schüler	**Methoden- und Medienkompetenzen**	**Vernetzungen**
	Zählen von Eingabezeichen (z.B. Klammern) und Problematisierung dieses Ansatzes *(b) Entwicklung eines Automaten mit Kellerspeicher* *(c) Anwendung eines Kellerautomaten zur Syntaxüberprüfung auf Grundlage von nicht-regulären Grammatiken* *(d) Implementierung eines Kellerautomaten zur Syntaxüberprüfung (Backtracking)*	• entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“, „Teilen und Herrschen“ und „Backtracking“ (M).		

Qulaifikationsphase Leistungskurs				
Unterrichts-vorhaben	**Inhaltsfeld** **Inhaltliche Schwerpunkte**	**Schwerpunkte der Kompetenzentwicklung** Die Schülerinnen und Schüler	**Methoden- und Medienkompetenzen**	**Vernetzungen**
Q2.3 *von-Neumann-Architektur und Ausführung maschinennaher Programme* *10 Ustd.*	*1. Von-Neumann-Architektur und die Ausführung maschinennaher Programme* *(a) prinzipieller Aufbau einer von Neumann-Architektur mit CPU, Rechenwerk, Steuerwerk, Register und Hauptspeicher* *(b) einige maschinennahe Befehle und ihre Repräsentation in einem Binär-Code, der in einem Register gespeichert werden kann* *(c) Analyse und Erläuterung der Funktionsweise eines einfachen maschinennahen Programms* *(d) Übersetzung der bekannten Kontrollstrukturen Verzweigung und Schleife in die Maschinennahe Programmierung*	• 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*	

Q2.4 *Grundlagen von Netzwerken und Implementierung von protokollbasierten Client-Server-Systemen in Anwendungskontexten* *20 Ustd.*	*1. Grundlagen den Datenübertragungen in Netzwerken* *(a) Schichtenmodell: Erarbeitung eines standardisierten Schichtenmodells für Netzwerkkommunikation* *(b) Topologien: Erarbeitung und Vergleich ausgewählter Netzwerktopologien* *(c) Subnetze und Routing: Analyse von Grundlagen der Adressierung in IPNetzwerken* *(d) Protokolle: Erarbeitung des beispielhaften Aufbaus eines Protokolls auf der Anwendungsschicht* *2. Analyse, Modellierung und Implementierung von Netzwerkanwendungen in Client-Server-Struktur* *(a) Nutzung einfacher Server-Dienste mittels der Klassen Connection und Client* *• Modellierung und Implementierung von Clients für einfach Serverdienste* *(b) Anbieten von Diensten mittels der Klasse Server* *• Analyse vorgegebener Implementationen einfacher Server* *• Modellierung und Implementierung eigener Server*	• 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, ihre Operationen und 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), • 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).	*Alle Inhalte werden an der Simualtionssoftware FILIUS erarbeite* *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 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*	

Qulaifikationsphase Leistungskurs				
Unterrichts-vorhaben	**Inhaltsfeld** **Inhaltliche Schwerpunkte**	**Schwerpunkte der Kompetenzentwicklung** Die Schülerinnen und Schüler	**Methoden- und Medienkompetenzen**	**Vernetzungen**
	(c) Entwicklung eines vollständigen Client-Server-Systems *• Protokollentwurf, Dialogorientierung* *• Bedeutung von Nebenläufigkeit* *• Implementierung*		*zwischen einzelnen Rechnern (einfacher Messenger), basierend auf selbst gewählten „Nicknames“.*	

Q2.5 *Entwicklung eines Netzwerkspiels unter Verwendung eines Vorgehensmodells aus der Softwareentwicklung* *20 Ustd.*	1. *Projekteinstieg / -planung (Spielauswahl, Zeitmanagement)* 2. *Projektumsetzung (informatische Analyse, Modellierung, Implementierung, Test (ggf. in Zyklen))* *(a) Analyse des Spiels mit dem Ziel einer späteren informatischen Umsetzung als Netzwerkspiel* *(b) Modellierung und Implementierung des Spiels als Netzwerkanwendung* *(c) Entwurf eines Protokolls zur Kommunikation zwischen Spielserver und -client basierend auf den erarbeiteten Spielregeln* *(d) Entwicklung von Entwurfs- und Implementationsdiagrammen für den Spielserver* *(e) Implementation und Test der Spielserver-Klasse und von dieser benötigter Fachklassen* *(f) Modellierung, Implementierung und Test des Spielclients* *(g) Test der Zusammenarbeit von Spielserver und Spielclient* *(h) Reflexion des Softwareprodukts* 3. *Projektreflexion (Reflexion der Projektplanung, Präsentation)*	• 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), • 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), • 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 Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I),	*Das umzusetzende Spiel wird mit dem Kurs abgesprochen.*	

Qulaifikationsphase Leistungskurs				
Unterrichts-vorhaben	**Inhaltsfeld** **Inhaltliche Schwerpunkte**	**Schwerpunkte der Kompetenzentwicklung** Die Schülerinnen und Schüler	**Methoden- und Medienkompetenzen**	**Vernetzungen**
		• 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-ServerKommunikation (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).		
Q2.6 *Wiederholung und Vertiefung ausgewählter Kompetenzen und Inhalte des ersten Jahrs der Qualifikationsphase*				