Schulinterner Lehrplan zum Kernlehrplan für die gymnasiale Oberstufe

Informatik

(Stand: 05.08.2022)

Inhalt

Seite

1 Die Fachgruppe Informatik des Städtischen Gymnasiums Petershagen 3

2 Entscheidungen zum Unterricht 3

2.1 Unterrichtsvorhaben 3
2.1.1 Übersichtsraster Unterrichtsvorhaben 5
2.1.2 Konkretisierte Unterrichtsvorhaben 8
2.2 Grundsätze der fachmethodischen und fachdidaktischen Arbeit 43
2.3 Grundsätze der Leistungsbewertung und Leistungsrückmeldung 44
2.3.1 Beurteilungsbereich Klausuren 44
2.3.2 Beurteilungsbereich Sonstige Mitarbeit 46

3 Entscheidungen zu fach- und unterrichtsübergreifenden Fragen 47

4 Qualitätssicherung und Evaluation 48

1 Die Fachgruppe Informatik des Städtischen Gymnasiums Petershagen

Das Städtische Gymnasium Petershagen ist eine Schule im ländlichen Raum, die derzeit von rund 1000 Schülerinnen und Schülern besucht wird. Das Einzugsgebiet der Schule umfasst die Stadt Petershagen (Kreis Minden-Lübbecke), aber auch die umliegenden niedersächsischen Gemeinden. Einige Schülerinnen und Schüler stammen aus Minden.

Derzeit unterrichten rund 90 Lehrerinnen und Lehrer an der Schule. Davon besitzen vier Lehrkräfte die Fakultas für den Informatikunterricht in Sekundarstufe II und zwei Lehrkräfte die Fakultas für den Informatikunterricht in Sekundarstufe I. Der Informatikunterricht im Jahrgang 6 wurde in den letzten Jahren noch von zwei weiteren Lehrkräften fachfremd durchgeführt.

Das Fach Informatik wird in Klasse 6 in zwei Wochenstunden und im Wahlpflichtbereich II (9/10 Klasse) in jeweils drei Wochenstunden unterrichtet.

Bei Informatikkursen in der Einführungsphase wird Wert daraufgelegt, dass keine Informatikkenntnisse aus der Sekundarstufe I zum erfolgreichen Durchlaufen des Kurses erforderlich sind, so dass eine Belegung eines Informatikkurses in Sekundarstufe I keine zwingende Voraussetzung für die Wahl des Faches Informatik in der Oberstufe ist. Dennoch kann es nicht ausbleiben, dass im Wahlpflichtbereich II auch vereinzelt Inhalte aus Sekundarstufe II vorweggenommen werden.

In der Unterstufe wird Informatik im Klassenverbund unterrichtet. Im Wahlpflichtbereich werden ein Informatikkurs, in EF zwei bzw. drei Kurse und in der Qualifikationsphase in der Regel pro Jahrgang ein Leistungskurs und ein Grundkurs eingerichtet.

Der Unterricht am städtischen Gymnasium Petershagen wird im 45-Minuten-Takt durchgeführt. Die Kursblockung der Oberstufe sieht für 3-stündige Grundkurse in aller Regel eine Doppel- und eine Einzelstunde vor. Für 5-stündige Leistungskurse sind zwei Doppelstunden und eine Einzelstunde vorgesehen.

Die gemeinsame Entwicklung von Unterrichtsmaterialien und Instrumenten der Leistungsbewertung ist ein wichtiger Beitrag der Fachgruppe Informatik zur Qualitätssicherung des Unterrichts. Die Fachkonferenz wird dieses Curriculum mindestens jährlich evaluieren und gegebenenfalls modifizieren.

2 Entscheidungen zum Unterricht

2.1 Unterrichtsvorhaben

Die Darstellung der Unterrichtsvorhaben im schulinternen Lehrplan besitzt den Anspruch, sämtliche im Kernlehrplan angeführten Kompetenzen abzudecken. Dies entspricht der Verpflichtung jeder Lehrkraft, Schülerinnen und Schülern Lerngelegenheiten zu ermöglichen, so dass alle Kompetenzerwartungen des Kernlehrplans von ihnen erfüllt werden können.

Die entsprechende Umsetzung erfolgt auf zwei Ebenen: der Übersichts- und der Konkretisierungsebene.

Im „Übersichtsraster Unterrichtsvorhaben“ (Kapitel 2.1.1) wird die für alle Lehrerinnen und Lehrer gemäß Fachkonferenzbeschluss verbindliche Verteilung der Unterrichtsvorhaben dargestellt. Das Übersichtsraster dient dazu,

den Kolleginnen und Kollegen einen schnellen Überblick über die Zuordnung der Unterrichtsvorhaben zu den einzelnen Jahrgangsstufen sowie den im Kernlehrplan genannten Kompetenzen, Inhaltsfeldern und inhaltlichen Schwerpunkten zu verschaffen. Der ausgewiesene Zeitbedarf versteht sich als grobe Orientierungsgröße, die nach Bedarf über- oder unterschritten werden kann. Um Freiraum für Vertiefungen, besondere Schülerinteressen, aktuelle Themen bzw. die Erfordernisse anderer besonderer Ereignisse (z.B. Praktika, Kursfahrten o.ä.) zu erhalten, wurden im Rahmen dieses schulinternen Lehrplans ca. 75 Prozent der Bruttounterrichtszeit verplant.

Während der Fachkonferenzbeschluss zum „Übersichtsraster Unterrichtsvorhaben“ zur Gewährleistung vergleichbarer Standards sowie zur Absicherung von Lerngruppenübertritten und Lehrkraftwechseln für alle Mitglieder der Fachkonferenz Bindekraft entfalten soll, beinhaltet die Ausweisung „konkretisierter Unterrichtsvorhaben“ (Kapitel 2.1.2) Beispiele und Materialien, die empfehlenden Charakter haben. Referendarinnen und Referendaren sowie neuen Kolleginnen und Kollegen dienen diese vor allem zur standardbezogenen Orientierung in der neuen Schule, aber auch zur Verdeutlichung von unterrichtsbezogenen fachgruppeninternen Absprachen zu didaktisch-methodischen Zugängen, fächerübergreifenden Kooperationen, Lernmitteln und -orten sowie vorgesehenen Leistungsüberprüfungen, die im Einzelnen auch den Kapiteln 2.2 bis 2.3 zu entnehmen sind.

Da in den folgenden Unterrichtsvorhaben Inhalte in der Regel anhand von Problemstellungen in Anwendungskontexten bearbeitet werden, werden in einigen Unterrichtsvorhaben jeweils mehrere Inhaltsfelder angesprochen.

2.1.1 Übersichtsraster Unterrichtsvorhaben

Unterrichtsvorhaben E-I	*Unterrichtsvorhaben E-II*	*Unterrichtsvorhaben E-III*	*Unterrichtsvorhaben E-IV*
Thema: *Geschichte der digitalen Datenverarbeitung und die Grundlagen des Datenschutzes* *Nutzung von Informatiksystemen*	**Thema**: *Grundlagen der objektorientierten Analyse, Modellierung und der Programmierung und algorithmischer Grundstrukturen in Java anhand anschaulicher Beispiele*	**Thema**: *Analyse, Entwurf und Implementierung einfacher Algorithmen mit Hilfe einer grafischen Benutzeroberfläche*	**Thema**: *Modellierung und Implementierung von Klassen- und Objektbeziehungen anhand von grafischen Animationen*
Zentrale Kompetenzen: • Argumentieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren	**Zentrale Kompetenzen:** • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren	**Zentrale Kompetenzen:** • Argumentieren • Modellieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren	**Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren
Inhaltsfelder: • Informatik, Mensch und Gesellschaft • Informatiksysteme	**Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten	**Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten	**Inhaltsfelder:** • Daten und ihre Strukturierung • Formale Sprachen und Automaten
Inhaltliche Schwerpunkte: • Geschichte der automatischen Datenverarbeitung • Datenschutz • Aufbau informatischer Systeme (Von-Neumann-Rechner, EVA-Prinzip)	**Inhaltliche Schwerpunkte:** • Objekte und Klassen • Objektorientierte Modellierung • Syntax und Semantik einer Programmiersprache • Analyse, Entwurf und Implementierung einfacher Algorithmen	**Inhaltliche Schwerpunkte:** • Erstellung einer GUI mit bzw.ohne GUI Builder • Nutzung der Klassenbibliotheken von Java • Analyse, Entwurf und Implementierung einfacher Algorithmen • Algorithmen zum Suchen und Sortieren	**Inhaltliche Schwerpunkte:** • Objekte und Klassen • Objektorientierte Modellierung • Syntax und Semantik einer Programmiersprache
Zeitbedarf: 6 Stunden	**Zeitbedarf**: 24 Stunden	**Zeitbedarf**: 24 Stunden	**Zeitbedarf**: 34 Stunden

Qualifikationsphase 1 (Grundkurs)				
Unterrichtsvorhaben Q1-I **Thema**: *Wiederholung der objektorientierten Modellierung und Programmierung anhand einer kontextbezogenen Problemstelung*	*Unterrichtsvorhaben Q1-II* **Thema:** *Modellierung und Implementierung von Anwendungen mit dynamischen, linearen Datenstrukturen (Stack, Queue und List)*	*Unterrichtsvorhaben Q1-III* **Thema**: *Suchen und Sortieren auf linearen Datenstrukturen*	*Unterrichtsvorhaben Q1-IV* **Thema**: *Modellierung und Implementierung von Anwendungen mit dynamischen, nichtlinearen Datenstrukturen*	*Unterrichtsvorhaben Q1-V* **Thema**: *Sicherheit und Datenschutz in Netzstrukturen*
Zentrale Kompetenzen: • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren	**Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren	**Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren	**Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren	**Zentrale Kompetenzen:** • Argumentieren • Darstellen und Interpretieren Kommunizieren und Kooperieren
Inhaltsfelder: • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten • Informatiksysteme	**Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten	**Inhaltsfelder:** • Algorithmen • Formale Sprachen und Automaten	**Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten	**Inhaltsfelder:** • Informatiksysteme Informatik, Mensch und Gesellschaft
Inhaltliche Schwerpunkte: • Objekte und Klassen • Objektorientierte Modellierung • Analyse, Entwurf und Implementierung von Algorithmen • Syntax und Semantik einer Programmiersprache • Nutzung von Informatiksystemen	**Inhaltliche Schwerpunkte:** • Objekte und Klassen • Analyse, Entwurf und Implementierung von Algorithmen • Algorithmen in ausgewählten informatischen Kontexten • Syntax und Semantik einer Programmiersprache	**Inhaltliche Schwerpunkte:** • Analyse, Entwurf und Implementierung von Algorithmen • Algorithmen in ausgewählten informatischen Kontexten • Syntax und Semantik einer Programmiersprache	**Inhaltliche Schwerpunkte:** • Objekte und Klassen • Analyse, Entwurf und Implementierung von Algorithmen • Algorithmen in informatischen Kontexten • Syntax und Semantik einer Programmiersprache	**Inhaltliche Schwerpunkte:** • Einzelrechner und Rechnernetzwerke • Kryptologie • Sicherheit • Nutzung von Informatiksystemen, Wirkungen der Automatisierung
Zeitbedarf: 10 Stunden	**Zeitbedarf**: 20 Stunden	**Zeitbedarf**: 15 Stunden	**Zeitbedarf**: 24 Stunden	**Zeitbedarf**: 15 Stunden

Qualifikationsphase 2 (Grundkurs)			
Unterrichtsvorhaben Q2-I **Thema:** *Modellierung und Nutzung von relationalen Datenbanken in Anwendungskontexten* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** • Datenbanken • Algorithmen in ausgewählten informatischen Kontexten • Syntax und Semantik einer Programmiersprache Sicherheit **Zeitbedarf**: 24 Stunden	*Unterrichtsvorhaben Q2-II* **Thema:** *Endliche Automaten und formale Sprachen* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Endliche Automaten und formale Sprachen **Inhaltliche Schwerpunkte:** • Endliche Automaten • Grammatiken regulärer Sprachen • Möglichkeiten und Grenzen von Automaten und formalen Sprachen **Zeitbedarf**: 20 Stunden	*Unterrichtsvorhaben Q2-III* **Thema**: *Prinzipielle Arbeitsweise eines Computers und Grenzen der Automatisierbarkeit* **Zentrale Kompetenzen:** • Argumentieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Informatiksysteme • Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** • Einzelrechner und Rechnernetzwerke • Grenzen der Automatisierung **Zeitbedarf**: 20 Stunden	*Unterrichtsvorhaben Q2-IV* **Thema**: *Wiederholung und Vertiefung ausgewählter Kompetenzen und Inhalte des ersten Jahrs der Qualifikationsphase*

Qualifikationsphase 1 (Leistungskurs)				
Unterrichtsvorhaben Q1-I **Thema**: *Wiederholung der objektorientierten Modellierung und Programmierung anhand einer kontextbezogenen Problemstellung mit einer Benutzeroberfläche (GUI).* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten • Informatiksysteme **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Objektorientierte Modellierung • Analyse, Entwurf und Implementierung von Algorithmen • Syntax und Semantik einer Programmiersprache • Nutzung von Informatiksystemen **Zeitbedarf**: 15 Stunden	*Unterrichtsvorhaben Q1-II* **Thema:** *Modellierung und Implementierung von dynamischen Datenstrukturen und von Anwendungen mit dynamischen linearen Datenstrukturen in kontextbezogenen Problemstellungen* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Analyse, Entwurf und Implementierung von Algorithmen • Algorithmen in ausgewählten informatischen Kontexten • Syntax und Semantik einer Programmiersprache **Zeitbedarf**: 25 Stunden	*Unterrichtsvorhaben Q1-III* **Thema**: *Modellierung, Implementierung, Analyse und Beurteilung von Such- und Sortierverfahren unterschiedlicher Komplexitätsklassen in kontextbezogenen Problemstellungen* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Algorithmen • Formale Sprachen und Automaten **Inhaltliche Schwerpunkte:** • Analyse, Entwurf und Implementierung von Algorithmen • Algorithmen in ausgewählten informatischen Kontexten • Syntax und Semantik einer Programmiersprache **Zeitbedarf**: 20 Stunden	*Unterrichtsvorhaben Q1-IV* **Thema**: *Modellierung und Implementierung von dynamischen nicht-linearen Datenstrukturen und von Anwendungen mit dynamischen nicht-linearen Datenstrukturen in kontextbezogenen Problemstellungen* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Analyse, Entwurf und Implementierung von Algorithmen • Algorithmen in informatischen Kontexten • Syntax und Semantik einer Programmiersprache **Zeitbedarf**: 40 Stunden	*Unterrichtsvorhaben Q1-V* **Thema**: *Grundlagen der Netzkommunikation sowie Modellierung und Implementierung von Client-Server-Anwendungen in kontextbezogenen Problemstellungen* *Entwicklung eines Netzwerkspiels* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren Kommunizieren und Kooperieren **Inhaltsfelder:** • Informatiksysteme Algorithmen **Inhaltliche Schwerpunkte:** • Algorithmen in ausgewählten Kontexten Nutzung von Informatiksystemen, Wirkungen der Automatisierung **Zeitbedarf**: 35 Stunden

Qualifikationsphase 2 (Leistungskurs)			
Unterrichtsvorhaben Q2-I **Thema:** *Modellierung, Implementierung und Nutzung von relationalen Datenbanken in Anwendungskontexten* *Projektorientierte Softwareentwicklung am Beispiel einer Anwendung mit Datenbankanbindung mit JAVA*	*Unterrichtsvorhaben Q2-II* **Thema:** *Grundlagen von Automaten und formalen Sprachen sowie die Modellierung und Implementierung eines Parsers zu einer formalen Sprache*	*Unterrichtsvorhaben Q2-III* **Thema**: *Sicherheit und Datenschutz in Informatiksystemen sowie Grenzen und Auswirkungen der Automatisierung*	*Unterrichtsvorhaben Q2-IV* **Thema**: *Prinzipielle Arbeitsweise eines Computers sowie Modellierung und Implementierung eines Scanners, Parsers und Interpreters für eine einfache maschinennahe Programmiersprache*
Zentrale Kompetenzen: • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren Kommunizieren und Kooperieren	**Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren Kommunizieren und Kooperieren	**Zentrale Kompetenzen:** • Argumentieren • Darstellen und Interpretieren Kommunizieren und Kooperieren	**Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Kommunizieren und Kooperieren
Inhaltsfelder: • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten Informatik, Mensch und Gesellschaft	**Inhaltsfelder:** Automaten und formale Sprachen	**Inhaltsfelder:** • Informatiksysteme Informatik, Mensch und Gesellschaft	**Inhaltsfelder:** • Informatiksysteme • Algorithmen • Formale Sprachen und Automaten • Informatik, Mensch und Gesellschaft
Inhaltliche Schwerpunkte: • Analyse, Entwurf und Implementierung von Algorithmen • Datenbanken • Algorithmen in ausgewählten informatischen Kontexten • Syntax und Semantik einer Programmiersprache Sicherheit	**Inhaltliche Schwerpunkte:** • Endliche Automaten und Kellerautomaten • Grammatiken regulärer und kontextfreien Sprachen • Scanner, Parser und Interpreter für eine reguläre Sprache Möglichkeiten und Grenzen von Automaten und formalen Sprachen	**Inhaltliche Schwerpunkte:** • Einzelrechner und Rechnernetzwerke • Kryptologie • Sicherheit Nutzung von Informatiksystemen, Wirkungen der Automatisierung	**Inhaltliche Schwerpunkte:** • Analyse, Entwurf und Implementierung von Algorithmen • Scanner, Parser und Interpreter für eine reguläre Sprache • Einzelrechner und Rechnernetzwerke
Zeitbedarf: 35 Stunden	**Zeitbedarf**: 30 Stunden	**Zeitbedarf**: 15 Stunden	**Zeitbedarf**: 20 Stunden

2.1.2 Konkretisierte Unterrichtsvorhaben

Im Folgenden sollen die im *Unterkapitel 2.1.1* aufgeführten Unterrichtsvorhaben konkretisiert werden.

Verbindliche Festlegungen der Fachkonferenz:
Die Fachkonferenz des Städtischen Gymnasiums Petershagen hat Themen, die Sequenzierung der Unterrichtsvorhaben (erste Tabellenspalte) und die ausgewiesenen Kompetenzen (zweite Tabellenspalte) verbindlich vereinbart. Alle Mitglieder der Fachkonferenz haben sich darauf verständigt, in ihrem Unterricht Lerngelegenheiten anzubieten, so dass Schülerinnen und Schüler diese Kompetenzen im Rahmen der festgelegten Unterrichtssequenzen erwerben oder vertiefen können.

Unterrichtliche Anregungen:
Die angeführten Beispiele, Medien und Materialien sind dagegen Vorschläge bzw. Hilfen für die Lehrkräfte. In diesen Bereichen sind Abweichungen von den vorgeschlagenen Vorgehensweisen möglich.

In der Qualifikationsphase werden die Unterrichtsvorhaben unter Berücksichtigung der Vorgaben für das Zentralabitur Informatik in NRW konkretisiert. Diese sind zu beziehen unter der Adresse

http://www.standardsicherung.schulministerium.nrw.de/abitur-gost/fach.php?fach=15 (abgerufen: 30. 04. 2014)

Einführungsphase

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

Die Schülerinnen und Schüler

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

Unterrichtsvorhaben EF-I: *Geschichte der digitalen Datenverarbeitung und die Grundlagen des Datenschutzes, Nutzung von Informatiksystemen*

Zeitbedarf: 6 Stunden

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Geschichte der digitalen Datenverarbeitung (a) Selbstständige Erarbeitung von Themen zur Geschichte der digitalen Datenverarbeitung in Kleingruppen (Referate) (b) Vorstellung und Diskussion durch Schülerinnen und Schüler	Die Schülerinnen und Schüler • bewerten anhand von Fallbeispielen die Auswirkungen des Einsatzes von Informatiksystemen (A), • erläutern wesentliche Grundlagen der Geschichte der digitalen Datenverarbeitung (A), • nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation. (K). • stellen ihre Ergebnisse in Form von Referaten bzw. Plakate dar (D) • nutzen die im Unterricht eingesetzten Informatiksysteme selbstständig, sicher, zielführend und verantwortungsbewusst (D),	*Beispiel:* Plakate, Referate zu • Geschichte der Digitalisierung: • Berühmte Persönlichkeiten in der Informatik • Auswirkungen der Digitalisierung: Veränderungen der Arbeitswelt und Datenschutz
2. Vertiefung des Themas Datenschutz (a) Erarbeitung grundlegender Begriffe des Datenschutzes (b) Problematisierung und Anknüpfung an die Lebenswelt der Schülerinnen und Schüler (c) Diskussion und Bewertung von Fallbeispielen aus dem Themenbereich „Datenschutz“		*Beispiel:* *Fallbeispiele aus dem aktuellen Tagesgeschehen* *Materialien:* Materialblatt zum Bundesdatenschutzgesetz
3. Aufbau informatischer Systeme (a) Herleitung der „Von-Neumann-Architektur“ (b) Identifikation des EVA-Prinzips (Eingabe-Verarbeitung-Ausgabe) als Prinzip der Verarbeitung von Daten und Grundlage der „Von-Neumann-Architektur“		*Materialien*: Demonstrationshardware (ausrangierter Schulrechner)
4. Information, deren Kodierung und Speicherung (a) Informatik als Wissenschaft der Verarbeitung von Informationen (b) Vereinbarung von Richtlinien zur Datenspeicherung auf den Schulrechnern (z.B. Ordnerstruktur, Dateibezeichner usw.		

Unterrichtsvorhaben EF-II: *Grundlagen der objektorientierten Analyse, Modellierung und der objektorientierten Programmierung und algorithmischer Grundstrukturen in Java anhand anschaulicher Beispiele*

Zeitbedarf: 24 Stunden

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Klassen und Objekte (a) Am Beispiel eines lebensweltnahen Beispiels werden Objekte im Sinne der Objektorientierten Modellierung eingeführt. (b) Objekte werden mit Objektkarten visualisiert und mit sinnvollen Attributen und „Fähigkeiten“, d.h. Methoden versehen. (c) 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 Beziehungen untereinander (M), • stellen Klassen, Assoziations- und Vererbungsbeziehungen in Diagrammen grafisch dar (D), • stellen den Zustand eines Objekts dar (D). • analysieren und erläutern einfache Algorithmen und Programme (A), • ordnen Klassen, Attributen und Methoden ihren Sichtbarkeitsbereich zu (M), • implementieren einfache Algorithmen unter Verwendung von Variablen und Wertzuweisungen, Kontrollstrukturen sowie Methodenaufrufen (I), • modifizieren und testen Programme schrittweise anhand von Beispielen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I). • dokumentieren Klassen durch Beschreibung der Funktionalität der Methoden (D).	*Beispiel:* Entwicklung eines Modells anhand eines lebensweltnahen Szenario: Autoverkauf, Vogelschwarm *Materialien:* BlueJ, Greenfoot, GLOOP, eclipse, Java-Editor, UMLed
2. Objektorientierte Modellierung (a) Entwurf der Implementationsdiagramme (b) Beziehungen unter Objekten (Vererbung und Assoziation) (c) Zugriffsrechte und Multiplizitäten		*Beispiel:* Skulpturengarten, Olympische Ringe
3. Implementierung einfacher Java-Klassen (a) Grundaufbau einer Java-Klasse (b) Deklaration und Initialisierung von Objekten (c) Konstruktoren (d) Attribute und Methoden (e) Kontrollstrukturen (*while*- und *for*-Schleifen, *if*-Verzweigungen)		

Unterrichtsvorhaben EF-III: *Analyse, Entwurf und Implementierung einfacher Algorithmen mit Hilfe einer grafischen Benutzeroberfläche mit bzw. ohne GUI Builder*

Zeitbedarf: 20 Stunden

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Programmierung einfacher Anwendungen mit Hilfe einer GUI mit bzw. ohne GUI Builder	Die Schülerinnen und Schüler • nutzen Java Klassenbibliotheken, indem sie eine GUI erstellen (I), • analysieren Such- und Sortieralgorithmen und wenden sie auf Beispiele an (D), • implementieren Such- und Sortieralgorithmen (I), • verwalten Zahlen in einem Array • beurteilen die Effizienz von Algorithmen am Beispiel von Sortierverfahren hinsichtlich Zeit und Speicherplatzbedarf (A).	*Beispiele*: Zahlenraten, Telefon, Taschenrechner, Memory, etc.
2. Erarbeitung mehrerer Sortierverfahren und Effizienzbetrachtungen (a) Formulierung mehrerer Algorithmen im Pseudocode (BubbleSort, SelectionSort, InsertionSort) (b) Implementierung von BubbleSort (c) Bewertung von Algorithmen anhand der Anzahl der nötigen Vergleiche und Vertauschungen (d) Sortierprobleme im Alltag (z.B. Dateisortierung, Tabellenkalkulation, Telefonbuch,usw)		*Beispiele:* (1) Entdecken der Sortierverfahren anhand von Spielkarten (*BubbleSort, SelectionSort, InsertionSort*) (2) Implementieren eines Sortierverfahrens mit Hilfe einer GUI. (3) Implementieren der Algorithmen zur Umwandlung zwischen Dezimal- und Dualzahlen
3. Lineare und binäre Suche (a) Suchaufgaben im Alltag und im Kontext informatischer Systeme (b) Implementierung der linearen Suche (c) Effizienzbetrachtungen zur linearen und binären Suche		*Beispiele:* (1) Simulationsspiel zur linearen und binären Suche mit Karten. (2) Implementieren des Suchens nach einer Zahl in einem Zahlenarray.

Unterrichtsvorhaben EF-IV: *Modellierung und Implementierung von Klassen- und Objektbeziehungen anhand von grafischen Animationen für graphisch repräsentierbare Objekte*

Zeitbedarf: 30 Stunden

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Modellierung grafisch repräsentierbarer Objekte (a) Entwurf der Implementationsdiagramme für graphische Objekte (Kreise, Kugeln, Rechtecke,…) (b) Aufstellen der Beziehungen unter den graphischen Objekten (Vererbung und Assoziation)	Die Schülerinnen und Schüler • analysieren und modellieren Klassen mit ihren Attributen, ihren Methoden, Vererbung- und Assoziationsbeziehungen (M), • stellen Klassen, Assoziations- und Vererbungsbeziehungen in Dia-grammen grafisch dar (D), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen und Objekttypen zu (M), • implementieren Klassen in Java auch unter Nutzung dokumentierter Klassenbibliotheken (I), • ordnen Klassen, Attributen und Methoden ihren Sichtbarkeitsbereich zu (M), • dokumentieren Klassen durch Beschreibung der Funktionalität der Methoden (D), • modifizieren und testen Programme schrittweise anhand von Beispielen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • präsentieren ihre Ergebnisse (D)	*Beispiel*: Seifenblasen, Sonnensystem (GLOOP) *Beispiel*: Modellierung graphischer Objekte mit Java *Materialien: BlueJ, GLOOP, eclipse, Java-Editor, UMLed*
2. Implementierung von Klassen für die einfachen graphischen Objekte unter Verwendung von Vererbung (a) Analyse und Erläuterung einer Basisversion der grafischen Klasse (b) Realisierung von grafischen Erweiterungen zur Basisklasse mit und ohne Vererbung (c) Analyse und Implementierung einer einfachen grafischen Animationsklasse		*Beispiele:* (1) Implementierung einer Animation mit GLOOP (Ufospiel , Billardkugeln, Autospiel, Schneemann). (2) Implementierung einer Animation mit graphischen Objekten mit Java ohne GLOOP.
3. Modellierung und Animation komplexerer grafisch repräsentierbarer Objekte (a) Analyse und Modellierung eines komplexen Animationsprogramms mit graphischen Elementen (b) Implementierung des Animationsprogramms. (c) Präsentation der Ergebnisse zu dem entwickelten Animationsprogramm.		*Beispiele:* die Schüler und Schülerinnen entwickeln ihre eigenen Animationen

Qualifikationsphase

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 (Grundkurs): Wiederholung der objektorientierten Modellierung und Programmierung

Zeitbedarf: 10 Stunden

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Wiederholung und Erweiterung der objektorientierten Modellierung und Programmierung durch Analyse und Erweiterung eines kontextbezogenen Beispiels (a) Analyse der Problemstellung (b) Analyse der Modellierung (Implementationsdiagramm) (c) Beziehungen unter Objekten (Vererbung und Assoziation) (d) Zugriffsrechte und Multiplizitäten (e) Abstrakte Klassen (f) Polymorphie (g) Dokumentation von Klassen (h) Implementierung der Anwendung oder von Teilen der Anwendung	Die Schülerinnen und Schüler • analysieren und erläutern objektorientierte Modellierungen (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M), • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen (D)	*Beispiel: Wetthüpfen* Für ein Wetthüpfen zwischen einem Hasen, einem Hund und einem Vogel werden die Tiere gezeichnet. Alle Tiere springen wiederholt nach links. Die Höhe und Weite jedes Hüpfers ist zufällig. Evtl. marschieren sie anschließend hintereinander her. *Beispiel: Tannenbaum* Ein Tannenbaum soll mit verschiedenen Arten von Schmuckstücken versehen werden, die durch unterschiedliche geometrische Objekte dargestellt werden. Es gibt Kugeln, Päckchen in der Form von Würfeln und Zuckerringe in Form von Toren. Ein Prototyp, der bereits mit Kugeln geschmückt werden kann, kann zur Verfügung gestellt werden. Da alle Schmuckstücke über die Funktion des Auf- und Abschmückens verfügen sollen, liegt es nahe, dass entsprechende Methoden in einer gemeinsamen Oberklasse realisiert werden. *Beispiel: Grußkarten* Grußkarten sollen modelliert und implementiert werden. Die Oberklasse *Karte* soll *abstract* sein und ihre Unterklassen sind *Valentin*, *Feiertag* und *Geburtstag*. Die Unterklassen erben konkrete (nicht abstrakte) Attribute wie z.B. *empfänger* und implementieren die abstrakte Methode *gruß()*.

Unterrichtsvorhaben Q1-II (Grundkurs): *Modellierung und Implementierung von Anwendungen mit dynamischen, linearen Datenstrukturen*

Zeitbedarf: 20 Stunden

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Die Datenstruktur Schlange im Anwendungskontext unter Nutzung der Klasse `Queue` (a) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen (b) Erarbeitung der Funktionalität der Klasse `Queue` (c) Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse `Queue`	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), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M), • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modifizieren Algorithmen und Programme (I), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I),	*Beispiel:* **Erweiterte Warteschlange** (Anwendung der Vererbung) Die erweiterte Warteschlange soll von der Queue erben und zusätzliche Funktionen bereitstellen (Daten ausgeben, löschen, zählen usw.) *Beispiel:* **Patientenwarteschlange** 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.
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`		*Beispiel:* **Erweiterter Stapel** (Anwendung der Vererbung) Der erweiterte Stapel soll von dem Stack erben und zusätzliche Funktionen bereitstellen (Daten ausgeben, löschen usw.) *Beispiel:* **Heftstapel** In einem Heftstapel soll das Heft einer Schülerin gefunden werden. *Beispiel*: **Kisten stapeln** In einem Stapel nummerierter Kisten soll eine bestimmte Kiste gefunden und an einen Kunden geliefert werden. Dazu müssen Kisten auf verschiedene Stapel gestapelt und wieder zurückgestellt werden. *Beispiel*: **Umgekehrt Polnische Notation (UPN)** Die sog. UPN (*Umgekehrt-Polnische-Notation*) bzw. *Postfix-Notation* eines Terms setzt den Operator hinter die Operanden. Um einen Term aus der gewohnten Infixschreibweise in einen Term in UPN umzuwandeln oder um den Wert des Terms zu berechnen, kann ein Stack verwendet 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`.	• 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).	*Beispiel:* **Erweiterte Liste** (Anwendung der Vererbung) Die erweiterte Liste soll von der Liste erben und zusätzliche Funktionen bereitstellen (Daten ausgeben, löschen usw.) *Beispiel*: **Abfahrtslauf** Bei einem Abfahrtslauf kommen die Skifahrer nacheinander an und werden nach ihrer Zeit in eine Rangliste eingeordnet. Diese Rangliste wird in einer Anzeige ausgegeben. Ankommende Abfahrer müssen an jeder Stelle der Struktur, nicht nur am Ende oder Anfang eingefügt werden können. *Beispiel*: **Schule** Schüler (Name, Vorname, Geburtsjahr, Geschlecht) sollen in einem Kurs (Liste) gespeichert und verwaltet werden.
4. Vertiefung - Anwendungen von Listen, Stapeln oder Schlangen in mindestens einem weiteren Kontext		*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. *Beispiel*: **Autos an einer Ampel zur Zufahrtsregelung** Es soll eine Ampel zur Zufahrtsregelung in Java simuliert werden. An einem geradlinigen, senkrecht von unten nach oben verlaufenden Straßenstück, das von Autos nur einspurig in eine Richtung befahren werden kann, ist ein Haltepunkt markiert, an dem die Ampel steht. Bei einem Klick auf eine Schaltfläche mit der Aufschrift „Heranfahren“ soll ein neues Auto an den Haltepunkt heranfahren bzw. bis an das letzte Auto, das vor dem Haltepunkt wartet. Grünphasen der Ampel werden durch einen Klick auf eine Schaltfläche mit der Aufschrift „Weiterfahren“ simuliert. In jeder Grünphase darf jeweils nur ein Auto weiterfahren. Die anderen Autos rücken nach.

Unterrichtsvorhaben Q1-III (Grundkurs): *Suchen und Sortieren auf linearen Datenstrukturen*

Zeitbedarf: 15 Stunden

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), • 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:* **Karteiverwaltung** Für ein Adressverwaltungsprogramm soll eine Methode zum Suchen einer Adresse geschrieben werden. *Beispiel*: **Bundesjugendspiele** Die Teilnehmer an Bundesjugendspielen nehmen an drei Disziplinen teil und erreichen dort Punktzahlen. Diese werden in einer Wettkampfkarte eingetragen und an das Wettkampfbüro gegeben. Zur Vereinfachung sollte sich das Modell auf die drei Disziplinen „Lauf”, „Sprung“ und „Wurf“ beschränken. Im Wettkampfbüro wird das Ergebnis erstellt. Das Programm soll dafür zunächst den Besten einer Disziplin heraussuchen können und später das gesamte Ergebnis nach gewissen Kriterien sortieren können.
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		
3. Untersuchung der Effizienz der Sortierverfahren „Sortieren durch direktes Einfügen“ und „Quicksort“ auf Arrays (a) Grafische Veranschaulichung der Sortierverfahren (b) Untersuchung der Anzahl der Vergleichsoperationen und des Speicherbedarf bei beiden Sortierverfahren (c) Beurteilung der Effizienz der beiden Sortierverfahren		

Unterrichtsvorhaben Q1-IV (Grundkurs): *Modellierung und Implementierung von Anwendungen mit dynamischen, nichtlinearen Datenstrukturen*

Zeitbedarf: 24 Stunden

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Die Datenstruktur Binärbaum im Anwedungskontext unter Nutzung der Klasse `BinaryTree` (a) Grundlegende Begriffe (Grad, Höhe, Blatt, Inhalt, Teilbaum, Ebene, Vollständigkeit) (b) Traversierung eines Binärbaums im Pre-, In- und Postorderdurchlauf (c) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen im Anwendungskontext (d) Modellierung eines Entwurfsdiagramms und Entwicklung eines Implementationsdiagramms (e) Erarbeitung der Klasse `BinaryTree` und beispielhafte Anwendung der Operationen (f) 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), • 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), • 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).	*Beispiel:* **Termbaum** Der Aufbau von Termen wird mit Hilfe von binären Baumstrukturen verdeutlicht. *Beispiel*: **Ahnenbaum** Die binäre Baumstruktur ergibt sich daraus, dass jede Person genau einen Vater und eine Mutter hat. *Beispiel*: **Entscheidungsbäume** Um eine Entscheidung zu treffen, werden mehrere Fragen mit „ja“ oder „nein“ beantwortet. *Beispiel*: **Morsebaum** Morse hat Buchstaben als Folge von Punkten und Strichen codiert. Diese Codierungen können in einem Binärbaum dargestellt werden.
2. Die Datenstruktur binärer Suchbaum im Anwendungskontext unter Verwendung der Klasse BinarySearchTree (a) Grafische Darstellung eines binären Suchbaums und Erarbeitung der Struktureigenschaften, Suchen, Einfügen und Löschen von Elementen. (b) Erstellen eines Suchbaums, Modellierung eines Entwurfsdiagramms und Entwicklung eines Implementationsdiagramm, (c) Erarbeitung der Klasse BinarySearchTree und Einführung des Interface Item zur Realisierung einer geeigneten Ordnungsrelation (d) Implementierung der Anwendung oder von Teilen der Anwendung inklusive einer sortierten Ausgabe des Baums.		*Beispiel*: **Informatikerbaum** als *Suchbaum* In einem binären *Suchbaum* werden die Namen und die Geburtsdaten von Informatikern lexikographisch geordnet abgespeichert. Funktionalitäten: Einfügen, Suchen, Ausgabe der Daten. *Beispiel*: **Stichwortverzeichnis** Es soll eine Anwendung entwickelt werden, die anhand von Stichworten und zugehörigen Seitenzahlen ein Stichwortregister erstellt. *Beispiel*: **Wörterbuch** Es soll ein Wörterbuch (englisch-> deutsch bzw. deutsch->englisch) modelliert und programmiert werden. Neue Wörter werden in der GUI eingegeben und in einem *BinarySearchTree* abgespeichert.

Unterrichtsvorhaben Q1-V (Grundkurs): *Sicherheit und Datenschutz in Netzstrukturen*

Zeitbedarf: 15 Stunden

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Daten in Netzwerken und Sicherheitsaspekte in Netzen sowie beim Zugriff auf Datenbanken (a) Beschreibung einer Client-Server-Struktur (b) Netztopologien als Grundlage von Client-Server-Strukturen und TCP/IP-Schichtenmodell als Beispiel für eine Paketübermittlung in einem Netz (c) Protokolle (HTTP, POP2, SMPT, IMAP, DNS, telnet, usw.) (d) Aufbau der IP-Adresse, Subnetzmaske	Die Schülerinnen und Schüler • beschreiben und erläutern Topologien, die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken (A), • analysieren und erläutern Eigenschaften und Einsatzbereiche symmetrischer und asymmetrischer Verschlüsselungsverfahren (A), • untersuchen und bewerten anhand von Fallbeispielen die Auswirkungen des Einsatzes von Informatiksystemen, die Sicherheit von Informatiksystemen sowie die Einhaltung der Datenschutzbestimmungen und des Urheberrechts (A), • untersuchen und bewerten Problemlagen, die sich aus dem Einsatz von Informatiksystemen ergeben, hinsichtlich rechtlicher Vorgaben, ethischer Aspekte und gesellschaftlicher Werte unter Berücksichtigung unterschiedlicher Interessenlagen (A), • nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zum Erschließen, zur Aufbereitung und Präsentation fachlicher Inhalte (D).	*Beispiel*: **E-Mail** Betrachtung des TCP/IP-Schichtenmodels am Beispiel des Verschickens einer E-Mail. *Beispiel*: **Online-Suchmaschine** Betrachtung des TCP/IP-Schichtenmodels am Beispiel der Benutzung einer Online-Suchmaschine.
2. Symmetrische und asymmetrische kryptografische Verfahren Cäsar-, Vigenère-, RSA-Verfahren		*Beispiel*: **Programmierung der Cäsar- und Vigenère-Verfahren**
3. Fallbeispiele zur Datenschutzproblematik und zum Urheberrecht		

Unterrichtsvorhaben Q2-I (Grundkurs): *Modellierung und Nutzung von relationalen Datenbanken in Anwendungskontexten*

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. 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 ER-Diagramms (b) **Entwicklung einer Datenbank aus einem Datenbankentwurf** • Modellierung eines relationalen Datenbankschematas zu einem ER-Diagramm inklusive der Bestimmung von Primär- und Sekundärschlüsseln (c) **Redundanz, Konsistenz und Normalformen** • Untersuchung einer Datenbank hinsichtlich Konsistenz und Redundanz in einer Anwendungssituation • Überführung der Datenbankschemata in die 1. bis 3. Normalform	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), • modellieren zu einem ER-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 ER-Diagramm grafisch dar (D),	*Beispiel:* **Fahrradverleih** Der Fahrradverleih *BTR* verleiht Fahrräder. Die Kunden sind bei *BTR* registriert (Name, Adresse, Telefon). *BTR* kennt von den Fahrradfirmen den Namen und die Telefonnummer. Kunden von *BTR* können CityBikes, Treckingräder und Mountainbikes ausleihen. *Beispiel:* **Buchungssystem** In einer Schule können die Lehrer Medienräume, Beamer, Laptops, Kameras für einen bestimmten Zeitpunkt buchen, der durch Datum und die Schulstunde festgelegt ist. *Beispiel:* **Schulverwaltung** In einer Software werden die Schulhalbjahre, Jahrgangsstufen, Kurse, Klassen, Schüler, Lehrer und Noten einer Schule verwaltet. *Beispiel:* **VideoCenter** *VideoCenter* ist eine Datenbank zur Verwaltung der Kunden, der Videos und der Ausleihe. *Beispiel*: **Schulbuchausleihe** Datenbank, die Daten einer Schulbuch-Ausleihe enthält: Entleiher, Bücher und Ausleihvorgänge. *Beispiel*: **Kursverwaltung** In der Datenbank *Kursverwaltung* werden Daten für Studenten, Professoren und Kurse verwaltet. *Beispiel*: **Musikschule** In der Datenbank *Musikschule* werden Schüler, Lehrer, Instrumente und Kurse verwaltet. *Beispiel*: **Wein** In der Datenbank werden Daten für den Verkauf von Wein (Artikel, Lieferant, Kunde) verwaltet.
2. Nutzung von relationalen Datenbanken (a) **Aufbau von Datenbanken und Grundbegriffe** (Analyse der Struktur der Datenbank und Erarbeitung der Begriffe Tabelle, Attribut, Datensatz, Datentyp, Primärschlüssel, Fremdschlüssel, Datenbankschema) (b) **SQL-Abfragen** • SQL-Abfragen (`SELECT (DISTINCT) …FROM, WHERE, AND, OR, NOT`) auf einer Tabelle • 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. 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 **4. Umsetzung des Softwareprojektes** *a)* Umsetzung der Datenbank in einem Datenbankmanagementsystem *b)* Implementierung der Kontrollklassen mit Anbindung an die Datenbank unter Verwendung didaktischer Klassen *c)* Integration in den Prototypen der Benutzeroberfläche	• modifizieren Algorithmen und Programme (I), • testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I), • 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), • verwenden die Syntax und Semantik einer Datenbankabfragesprache, um Informationen aus einem Datenbanksystem zu extrahieren (I) • wenden didaktisch orientierte Entwicklungsumgebungen zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I), • 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).	*Materialien*: Sqliteman, SQLiteadmin

Zeitbedarf: **24 Stunden**

Unterrichtsvorhaben Q2-II (Grundkurs): *Endliche Automaten und formale Sprachen*

Zeitbedarf: 20 Stunden

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien oder Materialien
1. Grammatiken und Automaten (a) Grammatiken als Konzept zur Formalisierung von Sprachen (b) Endliche Automaten als Maschinenmodell zur Erkennung von Sprachen (c) Grenzen endlicher Automaten: Sprachen, die nicht von Automaten erkannt werden können (d) Reguläre Grammatiken und Sprachen und deren Zusammenhang mit endlichen 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), • 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).	*Beispiele für Sprachen:* Wörter, die mit bestimmten Teilwörtern anfangen bzw. enden bzw. sie enthalten; Wörter gerader bzw. ungerader Länge; Terme; Palindrome *Beispiele für nicht von Automaten erkennbare Sprachen:* $a^n b^n$ (im Vergleich zu $a^n b^m$) ; Palindrome *Materialien*: Excorciser (eine Software zur Entwicklung von Automaten)

Unterrichtsvorhaben Q2-III (Grundkurs): *Prinzipielle Arbeitsweise eines Computers und Grenzen der Automatisierbarkeit*

Zeitbedarf: 20 Stunden

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 b) einige maschinennahe Befehlen und ihre Repräsentation in einem Binär-Code, der in einem Register gespeichert werden kann c) 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), • untersuchen und beurteilen Grenzen des Problemlösens mit Informatiksystemen (A).	*Beispiel:* Addition von 4 zu einer eingegeben Zahl mit einem Rechnermodell *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.3 –Von-Neumann-Architektur und maschinennahe Programmierung (Download Q2-III.1)
2. Grenzen der Automatisierbarkeit a) Vorstellung des Halteproblems b) Unlösbarkeit des Halteproblems c) Beurteilung des Einsatzes von Informatiksystemen hinsichtlich prinzipieller Möglichkeiten und prinzipieller Grenzen		*Beispiel:* Halteproblem *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.3 - Halteproblem (Download Q2-III.2)

Unterrichtsvorhaben Q2-IV (Grundkurs):

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

Qualifikationsphase - LEISTUNGSKURS

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 (Leistungskurs): *Wiederholung der objektorientierten Modellierung und Programmierung anhand einer kontextbezogenen Problemstellung unter Berücksichtigung der Gestaltung einer Benutzungsoberfläche*

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Entwicklung eines Anforderungskatalogs **2. Entwurf einer grafischen Benutzungsoberfläche** **3. Klassenmodellierung und Visualisierung von Objektkommunikation** a) Analyse der Problemstellung b) Analyse der Modellierung (Implementationsdiagramm) c) Beziehungen unter Objekten (Vererbung und Assoziation) d) Zugriffsrechte und Multiplizitäten e) Abstrakte Klassen f) Polymorphie g) Dokumentation von Klassen **4. Projektentwicklung und Test** a) Implementierung der Anwendung mit grafischer Benutzungsoberfläche b) Test einzelner Klassen der Anwendung mit Hilfe von Testanwendungen	Die Schülerinnen und Schüler • analysieren und erläutern OOM (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbez. unter Angabe von Multiplizitäten (M), • ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M), • entwickeln einfache Benutzungsoberflächen zur Kommunikation mit einem Informatiksystem (M), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • wenden eine didaktisch orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme systematisch anhand von Beispielen und mithilfe von Testanwendungen(I), • stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen (D), • stellen die Kommunikation zwischen Objekten grafisch dar (D), • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Daten, zur Organisation von Arbeitsabläufen sowie zur Verteilung und Zusammenführung von Arbeitsanteilen (K).	*Beispiel:* **Taschenrechner** Schülerinnen und Schüler entwickeln eine Taschenrechneranwendung. Dabei kann es sich um einen Rechner mit einfachen Grundrechenarten bis hin zu einem wissenschaftlichen Taschenrechner mit umfangreicheren Funktionen handeln. Die Anforderungen sind von den Schülerinnen und Schülern selbst zu definieren. *Beispiel:* **Schulverwaltungssystem** Schülerinnen und Schüler entwickeln eine Anwendung für die Verwaltung der Daten für Schüler und Lehrer. Die Klassen *Schüler* und *Lehrer* erben von der Klasse *Person*. In der Klasse *Schule* werden Lehrer und Schüler in jeweils einem Array gespeichert. Die Angaben zu den Lehrern und Schülern können aus den gegebenen Daten zufällig erzeugt werden. Anschließend sollen die Daten in der Konsole ausgegeben werden. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator (Download LK-Q1-I.1) (Download LK-Q1-I.2)

Unterrichtsvorhaben Q1-II (Leistungskurs): Modellierung und Implementierung von Anwendungen mit dynamischen, linearen Datenstrukturen

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Modellierung und Implementation einer Datenstruktur Schlange. **2. Die Datenstruktur Schlange im Anwendungskontext unter Nutzung der Klasse `Queue`** a) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen b) Erarbeitung der Funktionalität der Klasse `Queue` und des Konzepts der parametrisierten Klassen. **3. Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse `Queue`**	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 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),	*Beispiel:* **Erweiterte Warteschlange** (Anwendung der Vererbung) Die erweiterte Warteschlange soll von der Queue erben und zusätzliche Funktionen bereitstellen (Daten ausgeben, löschen, zählen usw.) *Beispiel:* **Patientenwarteschlange** mit einer GUI Sobald ein Patient in einer Arztpraxis eintrifft, werden sein Name und seine Krankenkasse erfasst. In der Klasse Wartezimmer werden die Patienten verwaltetet: Entfernen bzw. Hinzufügen eines Patienten, wenn er zur Behandlung gerufen wird). Patienten können sich zusätzlich in die Warteschlange zum Blutdruckmessen einreihen. Objekte werden von zwei Schlangen verwaltet. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator (Download LK-Q1-II.1) (Download LK-Q1-II.2)
4. 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`		*Beispiel:* **Erweiterter Stapel** (Anwendung der Vererbung) Der erweiterte Stapel soll von dem Stack erben und zusätzliche Funktionen bereitstellen (Daten ausgeben, löschen usw.) *Beispiel*: **Kisten stapeln** In einem Stapel nummerierter Kisten soll eine bestimmte Kiste gefunden und an einen Kunden geliefert werden. Dazu müssen Kisten auf verschiedene Stapel gestapelt und wieder zurückgestellt werden. *Beispiel*: **Umgekehrt Polnische Notation (UPN)** Die sog. UPN (*Umgekehrt-Polnische-Notation*) bzw. *Postfix-Notation* eines Terms setzt den Operator hinter die Operanden. Um einen Term aus der gewohnten Infixschreibweise in einen Term in UPN umzuwandeln oder um den Wert des Terms zu berechnen, kann ein Stack verwendet werden. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator (Download LK-Q1-II.3)

5. 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`	• 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).	*Beispiel:* **Erweiterte Liste** (Anwendung der Vererbung) Die erweiterte Liste soll von der Liste erben und zusätzliche Funktionen bereitstellen (Daten ausgeben, löschen usw.) *Beispiel*: **Abfahrtslauf** Bei einem Abfahrtslauf kommen die Skifahrer nacheinander an und werden nach ihrer Zeit in eine Rangliste eingeordnet. Diese Rangliste wird in einer Anzeige ausgegeben. Ankommende Abfahrer müssen an jeder Stelle der Struktur, nicht nur am Ende oder Anfang eingefügt werden können. *Beispiel:* **Todo-Liste** Es werden Aufgaben verwaltet, die in eine vom Benutzer gewünschte Reihenfolge angeordnet werden. Der Benutzer kann in der Todo-Liste navigieren, Einträge einfügen, bestehende Einträge ändern oder löschen. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator (Download LK-Q1-II.4)
6. Vertiefung – Anwendungen von Listen, Stapeln oder Schlangen in mindestens einem weiteren Kontext a) Modellierung und Implementierung einer weiteren Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse `Queue` b) Modellierung und Implementierung einer weiteren Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse `Stack` c) Modellierung und Implementierung einer weiteren kontextbezogenen Anwendung unter Verwendung der Klasse `List`		*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. *Beispiel*: **Autos an einer Ampel zur Zufahrtsregelung** Es soll eine Ampel zur Zufahrtsregelung simuliert werden. An einem geradlinigen Straßenstück, das von Autos nur einspurig in eine Richtung befahren werden kann, ist ein Haltepunkt markiert, an dem die Ampel steht. Bei einem Klick auf eine Schaltfläche mit der Aufschrift „Heranfahren“ soll ein neues Auto an den Haltepunkt heranfahren bzw. bis an das letzte Auto, das vor dem Haltepunkt wartet. Grünphasen der Ampel werden durch einen Klick auf eine Schaltfläche mit der Aufschrift „Weiterfahren“ simuliert. In jeder Grünphase darf jeweils nur ein Auto weiterfahren. Die anderen Autos rücken nach.

Unterrichtsvorhaben Q1-III (Leistungskurs): *Modellierung, Implementierung, Analyse und Beurteilung von Such- und Sortierverfahren unterschiedlicher Komplexitätsklassen*

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
4. 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 (Speicherbedarf, Anzahl der Vergleiche)	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“ , „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:* **Karteiverwaltung** Für ein Adressverwaltungsprogramm soll eine Methode zum Suchen einer Adresse geschrieben werden. *Beispiel:* **Benutzerverwaltung für einen Fotokopierer** Benutzerinnen und Benutzer des Kopierers geben eine vierstellige Ziffernkombination an, um sich zu legitimieren. Über diesen PIN-Code soll auch die Benutzerinnen und Benutzer des Kopierers identifizierbar sein, um auf sein Konto den Kopierverbrauch zu buchen. Die Benutzungsdaten werden in linearen Strukturen verwaltet, die nach den PIN-Nummern sortiert vorliegen. Die Benutzungsdaten sollen nach den Kriterien Kopierverbrauch, Namen und PIN-Nummern geordnet ausgegeben werden können. *Beispiel:* **Test- und Analyseumgebung für Sortieralgorithmen mit einer GUI** Listen mit unterschiedlich vielen Listenelementen und unterschiedlicher Vorsortierung werden bezüglich der Anzahl der Vergleiche von Listenelementen analysiert. Die untere Schranke für die Laufzeit wird bestimmt. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator (Download Q1-III.1) (Download Q1-III.2) (Download Q1-III.3)
5. Sortieren in Listen und Arrays - Entwicklung und Implementierung von iterativen und rekursiven Sortierverfahren a) Entwicklung und Implementierung einfacher Sortierverfahren (BubbleSort, SelectionSort, InsertionSort) für eine Liste b) Implementierung der einfachen Sortierverfahrens für ein Feld *c)* Entwicklung eines rekursiven Sortierverfahren für ein Feld (z.B. Quicksort)		
6. 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 bei beiden Sortierverfahren c) Beurteilung der Effizienz der beiden Sortierverfahren, untere Schranke für die Laufzeit von Sortieralgorithmen		

Unterrichtsvorhaben Q1-IV (Leistungskurs): *Modellierung und Implementierung von dynamischen nicht-linearen Datenstrukturen und von Anwendungen mit dynamischen nicht-linearen Datenstrukturen in kontextbezogenen Problemstellungen.*

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Die Datenstruktur Binärbaum im Anwendungskontext unter Nutzung der Klasse `BinaryTree<ContentType>` a) Definition eines Binärbaums und grundlegende Begriffe b) Erarbeitung der Klasse `BinaryTree<ContentType>` und beispielhafte Anwendung der Operationen c) Implementierung der Traversierung eines Binärbaums im Pre-, In- und Postorderdurchlauf *d)* Modellierung und Implementierung einer Anwendung unter Verwendung der Datenstruktur Binärbaum	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), • 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 „Backtracking“(M),	*Beispiel*: **MorseBaum** Die Buchstaben werden in einem Morsebaum als Folge von Punkten und Strichen codiert. *Beispiel:* **Termbaum** Der Aufbau von Termen wird mit Hilfe von binären Baumstrukturen verdeutlicht. *Beispiel*: **Ahnenbaum** Die binäre Baumstruktur ergibt sich daraus, dass jede Person genau einen Vater und eine Mutter hat. *Beispiel*: **Entscheidungsbäume** Um eine Entscheidung zu treffen, werden mehrere Fragen mit ja oder nein beantwortet. *Materialien*: Ergänzungsmaterialien zum Lehrplannavigator (Download LK-Q1-V.3)
2.* Erarbeitung, Implementierung und Verwendung der Datenstruktur binärer Suchbaum im Anwendungskontext** a) Erarbeitung der Eigenschaften eines binären Suchbaums im Anwendungskontext b) Erarbeitung der Attribute und Methoden der generischen Klasse `BinarySearchTree<ContentType>` und des Interfaces `ComparableContent` c) Implementierung des Konstruktors und der Methode `search` der Klasse `BinarySearchTree<ContentType>` ***d) Implementierung eines Anwendungsbeispiels einschließlich der sortierten Ausgabe eines binären Suchbaumes		*Beispiel*: **Informatikerbaum als *Suchbaum*** In einem binären *Suchbaum* werden nach Namen sortiert die Namen und die Geburtsdaten von Informatikern lexikographisch geordnet abgespeichert. *Beispiel*: **Buchindex** Als weiteres Anwendungsbeispiel, das mehrere dynamische Datenstrukturen miteinander verknüpft, soll ein Programm modelliert und implementiert werden, das das Stichwortregister eines Buches verwaltet. Die Wörter werden in einem binären Suchbaum verwaltet, die zugehörigen Seitenzahlen als lineare Listen. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator (Download LK-Q1.V.4)

3. **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)	• 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), • 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), • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Daten, zur Organisation von Arbeitsabläufen sowie zur Verteilung und Zusammenführung von Arbeitsanteilen (K).	*Beispiel:* **Das Haus vom Nikolaus** Das Haus vom Nikolaus ist das bekannteste Beispiel für einen Graphen, für den ein Eulerweg, aber kein Eulerkreis existiert. Anhand dieses Beispiels werden die Grundbegriffe der Graphentheorie sowie die Darstellung eines Graphen als Adjazenzmatrix eingeführt. *Beispiel:* **Soziale Netzwerke** Da es in dem Graph eines sozialen Netzwerks im Verhältnis zu den Knoten nur wenige Kanten gibt, bietet sich dieses Beispiel zur Einführung der Darstellung eines Graphen in Form von Adjazenzlisten an. *Materialien*: Ergänzungsmaterialien zum Lehrplannavigator (Download LK-Q1.V.1)
4. **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.		*Beispiel:* **Soziale Netzwerke** • Konstruktion eines Beispielgraphen • Anzahl der Knoten • Summe der Kantengewichte • Anzahl der Nachbarn eines Knotens Beispiel: **Wegsuche** Suche nach einem beliebigen bzw. besten Weg zwischen zwei Knoten. Beispiel: **Versorgungsnetz** Die Problemstellung, Verbraucher eines Dorfes möglichst kostengünstig an ein Versorgungsnetz (Kabel, Gas, Telefon) anzuschließen, motiviert die Behandlung des minimalen Spannbaumes eines Graphen. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator (Download LK-Q1.V.2)

Unterrichtsvorhaben Q1-V (Leistungskurs): *Grundlagen der Netzwerkkommunikation sowie Modellierung und Implementierung von Client-Server-Anwendungen in kontextbezogenen Problemstellungen Entwicklung eines Netzwerkspiels*

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Grundlagen der Datenübertragung in Netzwerken (a) Schichtenmodell • Erarbeitung eines standardisierten Schichtenmodells für Netzwerkkommunikation (b) Grundlagen der Codierung • Erarbeitung einer eigenen, vereinfachten Codierung für die Bitübertragung (c) Topologien • Erarbeitung und Vergleich ausgewählter Netzwerktopologien (d) Routing • Analyse von Grundlagen der Adressierung in IP-Netzwerken (e) 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), • 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),	*Material:* **Aufgabensammlung** Anhand einer Sammlung von Aufgabenblättern (teils inkl. implementierter Begleitwerkzeuge) erarbeiten die Schülerinnen und Schüler im Anwendungskontext Grundlagen der Inhaltspunkte „OSI-Referenzmodell“, „Codierung“, „Topologien“, „Routing“ und „Protokolle“
2. Analyse, Modellierung und Implementierung von Netzwerkanwendungen in Client-Server-Struktur (a) Nutzung einfacher Server-Dienste mittels Client • Modellierung und Implementierung von Clients für einfach Serverdienste (b) Anbieten von Diensten mittels Server • Analyse vorgegebener Implementationen einfacher Server • Modellierung und Implementierung eigener Server		*Beispiel:* **Daytime-Clients und –Server** Anhand der Echo- und Daytime-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 selbst.

3. Entwicklung eines vollständigen Client-Server-Systems • Protokollentwurf, Dialogorientierung • Modellierung mittels Entwurfs- und Implementationsdiagramm • Bedeutung von Nebenläufigkeit • Implementierung	• implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I)	*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“.
4. Entwicklung eines Netzwerkspiels a) Projekteinstieg / -planung (Spielauswahl, Zeitmanagement) b) Analyse des Netzwerkspiels c) Modellierung und Implementierung des Spiels als Netzwerkanwendung d) Entwicklung einer Testanwendung e) Reflexion des Softwareprodukts		*Beispiel:* Wörterraten *Beispiel*: Schiffe versenken *Beispiel:* GraphColoringGame Zwei-Personen-Spiel, bei dem die beiden Personen abwechselnd am Zug sind. Das Spielfeld ist ein ungerichteter Graph ohne Mehrfachkanten. Die Knoten des Graphen können mit vorgegebenen Farben markiert werden. Die beiden Spielerinnen bzw. Spieler färben abwechselnd einen noch ungefärbten Knoten. Dabei muss beachtet werden, dass kein adjazenter Knoten mit derselben Farbe markiert wurde. Es verliert derjenige Spielerinn bzw. derjenige Spieler, die bzw. der unter diesen Bedingungen keinen Zug mehr machen kann. Sind alle Knoten korrekt gefärbt, endet das Spiel unentschieden. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator (Download LK-Q2-V.1)

Unterrichtsvorhaben Q2-I (Leistungskurs): *Modellierung, Implementierung und Nutzung von relationalen Datenbanken in Anwendungskontexten. Projektorientierte Softwareentwicklung am Beispiel einer Anwendung mit Datenbankanbindung mit JAVA*

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
5. 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, Relationenschema, 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 mehreren Tabelle zur Beantwortung der Fragestellungen (`JOIN` und Varianten, `UNION, AS, GROUP BY,ORDER BY, ASC, DESC, COUNT, MAX, MIN, SUM, AVG,` Arithmetische Operatoren: `+, -, *, /, (…)`, Vergleichsoperatoren: `=, <>, >, <, >=, <=, LIKE, BETWEEN, IN, IS NULL`) c) Vertiefung an einem weiteren Datenbankbeispiel	Die Schülerinnen und Schüler • 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), • 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), • analysieren und erläutern eine Datenbankmodellierung (A), • erläutern die Eigenschaften normalisierter Datenbankschemata (A), • überprüfen Datenbankschemata auf vorgegebene Normalisierungseigenschaften (D), • überführen Datenbankschemata in die 1. bis 3. Normalform (M), • ermitteln Ergebnisse von Datenbankabfragen über mehrere verknüpfte Tabellen (D),	*Beispiel:* **VideoCenter** VideoCenter ist die Simulation einer Online-Videothek für den Informatik-Unterricht mit Webfrontend zur Verwaltung der Kunden, der Videos und der Ausleihe. Außerdem ist es möglich direkt SQL-Abfragen einzugeben. Es ist auch möglich, die Datenbank herunterzuladen und lokal zu installieren. Unter `http://dokumentation.videocenter.schule.de/old/video/index.html` (abgerufen: 01. 02. 2016) *Beispiel*: **Schulbuchausleihe** Unter `www.brd.nrw.de/lerntreffs/informatik/structure/material/sek2/datenbanken.php` (abgerufen: 01. 02. 2016) wird eine Datenbank zur Verfügung gestellt, die Daten einer Schulbuch-Ausleihe enthält (über 1000 Entleiher, 200 Bücher mit mehreren tausend Exemplaren und viele Ausleihvorgänge). Die Datenbank kann in OpenOffice eingebunden werden.

6. Modellierung von relationalen Datenbanken a) Entity-Relationship-Diagramm • Ermittlung von Entitätstypen, Attributen zugehöriger Entitäten, Beziehungstypen 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 Datenbankschemas 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)	• 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), • erläutern Eigenschaften und Aufbau von Datenbanksystemen unter dem Aspekt der sicheren Nutzung (A). • 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),	*Beispiel:* **Fahrradverleih** Der Fahrradverleih *BTR (BikesToRent)* verleiht unterschiedliche Typen von Fahrrädern diverser Firmen an seine Kunden. Die Kunden sind bei *BTR* registriert (Name, Adresse, Telefon). *BTR* kennt von den Fahrradfirmen den Namen und die Telefonnummer. *Beispiel:* **Buchungssystem** In dem Online-Buchungssystem einer Schule können die Lehrenden Medienräume, Beamer, Laptops, Kameras, usw. für einen bestimmten Zeitpunkt buchen, der durch Datum und die Schulstunde festgelegt ist. Dazu ist die Datenbank zu modellieren, ggf. zu normalisieren und im Datenbanksystem umzusetzen. Weiter sollen sinnvolle Abfragen entwickelt werden. http://mrbs.sourceforge.net (abgerufen: 01. 02. 2016) *Beispiel:* **Schulverwaltung** In einer Software werden die Schulhalbjahre, Jahrgangsstufen, Kurse, Klassen, Schülerinnen und Schüler, Lehrende und Noten einer Schule verwaltet. Man kann dann ablesen, dass z.B. Schüler X von Lehrerin 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 auf das Thema Datenschutz eingegangen werden.

7. Analyse einer lebensweltnahen Problemstellung im Hinblick auf die Entwicklung eines Java-Programms mit Datenbankanbindung d) Entwicklung einer Programmidee e) Analyse des Problembereichs f) Entwicklung eines Anforderungskatalogs für das zu entwickelnde Programm **8. 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 **9. Umsetzung des Softwareprojektes** *d)* Umsetzung der Datenbank in einem Datenbankmanagementsystem *e)* Implementierung der Kontrollklassen mit Anbindung an die Datenbank unter Verwendung didaktischer Klassen *f)* Integration in den Prototypen der Benutzeroberfläche **10. Optional: Evaluation des Projekts**	• modifizieren Algorithmen und Programme (I), • testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I), • 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), • 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), • 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).	*Beispiel*: **Quizspiel** Verwaltung von Quizfragen, Antworten und Ranglisten *Beispiel*: **Navigationssystem** Ermittlung von kürzesten Wegen im deutschen Autobahnnetz *Beispiel*: **Verwaltung der Schülerbücherei** Verwaltungsprogramm für das Einpflegen und Ausleihe von Büchern der Schülerbibliothek *Beispiel*: **Fehlstundenverwaltung** Verwaltungsprogramm für die Fehlstunden von Schülerinnen und Schülern *Beispiel*: **Sportfestverwaltung** Verwaltung von Aufgaben, Sportereignissen und Ergebnissen des Schulsportfestes *Beispiel*: **Materialverwaltung** Materialien für Vertretungsstunden sollen verwaltet werden Materialien: Ergänzungsmaterialien zum Lehrplannavigator (Download LK-Q1-VI.1) (Download LK-Q1-VI.2) (Download LK-Q1-VI.3)

Unterrichtsvorhaben Q2-II (Leistungskurs): *Grundlagen von Automaten und formalen Sprachen sowie die Modellierung und Implementierung eines Scanners, Parsers und Interpreters zu einer formalen Sprache*

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien oder Materialien
1. Endliche Automaten (a) Erarbeitung der formalen Beschreibung eines endlichen Automaten auf der Grundlage von Automaten in bekannten Kontexten (b) Untersuchung, Darstellung und Entwicklung endlicher Automaten (c) Umwandlung nichtdeterministischer endlicher Automaten in deterministische endliche 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), • 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),	*Beispiele:* Cola-Automat, Geldspielautomat, Roboter, Zustandsänderung eines Objekts „Auto“, Akzeptor für bestimmte Zahlen, Akzeptor für Teilwörter in längeren Zeichenketten, Akzeptor für Terme *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator (Download LK-Q2-III.1)
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		*Beispiele:* reguläre Grammatik für Wörter mit ungerader Parität, Grammatik für Wörter, die bestimmte Zahlen repräsentieren, Satzgliederungsgrammatik *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator (Download LK-Q2-III.2)
3. Grenzen endlicher Automaten		*Beispiele:* Klammerausdrücke, $a^n b^n$ im Vergleich zu $(ab)^n$ *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator (Download LK-Q2-III.3)

4. Entwicklung eines Kellerautomaten als Antwort auf die Grenzen endlicher Automaten (a) Erweiterung eines DEA um eine einzelne Speichervariable zum 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)	• 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). • entwickeln und modifizieren zu einer Problemstellung endliche Automaten oder Kellerautomaten (M), • analysieren und erläutern Grammatiken regulärer und kontextfreier Sprachen (A), • modifizieren Grammatiken regulärer und kontextfreier Sprachen (M), • ermitteln die formale Sprache, die durch eine Grammatik erzeugt wird (A), • entwickeln zu einer regulären oder kontextfreien Sprache eine Grammatik, die die Sprache erzeugt (M), • modellieren und implementieren Scanner, Parser und Interpreter zu einer gegebenen regulären Sprache (I), • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Daten, zur Organisation von Arbeitsabläufen sowie zur Verteilung und Zusammenführung von Arbeitsanteilen (K), • untersuchen und beurteilen Grenzen des Problemlösens mit Informatiksystemen (A).	*Materialien:* Ergänzungsmaterialien zum Lehrplannavigator (Download LK-Q2-III.4)
5. Die Schritte eines Compilers (a) Scanner: • endlicher Automat als Grundlage • Vorgabe von Symboltabelle und Tokenliste zur Verwaltung und Erkennung des Quelltextes • Erweiterung des terminalen Alphabets der zu übersetzenden formalen Sprache • Implementierung als endlicher Automat (b) Parser: • reguläre (oder wahlweise kontextfreie) Grammatik als Grundlage • Vorgabe einer Grundversion des Parsers • Erweiterung des Sprachumfangs • Implementierung der Parsermethoden für die Produktionsregeln der kontextfreien Grammatik (c) Interpreter • Vorgabe einer Grundversion des Interpreters • Erweiterung des Sprachumfangs Implementierung		*Beispiel:* Der Scanner-Automat zur Erkennung der einzelnen Symbole wird als endlicher Automat realisiert. Mithilfe der Symboltabelle wird die Vereinfachung des Automaten deutlich gemacht und der vereinfachte Scanner-Automat wird schrittweise erweitert und implementiert. In der zweiten Phase wird die der Pseudoprogrammiersprache zugrunde liegende Grammatik analysiert. Die Überprüfung der syntaktischen Korrektheit wird in Form eines Parsers modelliert und implementiert. Dabei wird der Sprachumfang der Pseudo-Programmiersprache schrittweise erweitert. In der dritten Phase wird ein zu Teilen bereits vorbereiteter Interpreter analysiert und erweitert, welcher Programme der Pseudo-Programmiersprache zur Konstruktion von Zeichnungen in eine Grafik übersetzt. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator (Download LK-Q2-IV.3)

Unterrichtsvorhaben Q2-III (Leistungskurs): *Sicherheit und Datenschutz in Informatiksystemen sowie Grenzen und Auswirkungen der Automatisierung*

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
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	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). • analysieren und erläutern Eigenschaften, Funktionsweisen und Einsatzbereiche symmetrischer und asymmetrischer Verschlüsselungsverfahren (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).	*Beispiel:* ***autonomes Fahren*** 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. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator (Download LK-Q2-I.1)
2. Erarbeitung grundlegender Positionen (ggf. in zieldifferenten Gruppen) 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		*Materialien:* Ergänzungsmaterialien zum Lehrplannavigator (Download LK-Q2-I.2) *Beispiel: Creative-Commons-Lizenzen* URL: http://de.creativecommons.org (Abgerufen am 19.06.2016)
3. Reflexion, Stellungnahme zu einem komplexen Fallbeispiel		*Materialien:* Ergänzungsmaterialien zum Lehrplannavigator (Download LK-Q2-I.3)
4. Sicherheitsziele und der kryptographische Ansatz a) Beschreibung und Gegenüberstellung von Sicherheitszielen anhand von Fallbeispielen b) Problematisierung von Sicherheitszielen in Bezug auf die Kommunikation über offene Kanäle c) Sicherheitsziele der Verschlüsselung		*Beispiel:* Sicherheitsziele *Zugriffskontrolle* und *Authentifikation* anhand des „Keyless Entry-System“ bei Autos *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator (Download LK-Q2.I.4)

5. Verschlüsselungsverfahren und deren Sicherheit a. Monoalphabetische Verschlüsselungsverfahren und Angriffsmöglichkeiten auf diese Verfahren b. Polyalphabetische Verschlüsselungsverfahren und Angriffsmöglichkeiten auf diese Verfahren c. Schlüsseltausch als Grundproblem symmetrischer Verschlüsselungsverfahren (Diffie-Hellman) d. Das Schlüsselpaar und die Einwegfunktion als zentrale Konzepte der asymmetrischen Verschlüsselung e. Probleme der asymmetrischen Verschlüsselung f. Signaturen als Anwendungsgebiet asymmetrischer Kryptographie		*Beispiel:* ***Caesar-Verschlüsselung*** *Beispiel:* ***Vigenère-Verschlüsselung*** *Beispiel:* ***One-Time-Pad*** Der One-Time-Pad als Spezialfall eines sicheren polyalphabetischen Verschlüsselungsverfahrens *Beispiel:* ***RSA-Verfahren*** Das RSA-Verfahren als Beispiel für eine asymmetrische Verschlüsselung mit einer Einwegfunktion *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator (Download LK-Q2.I.5) *Materialien:* URL*: https://www.cryptool.org/de/* *(Abgerufen am 19.06.2016)*

Unterrichtsvorhaben Q2-IV (Leistungskurs): *Prinzipielle Arbeitsweise eines Computers sowie Modellierung und Implementierung eines Scanners, Parsers und Interpreters für eine einfache maschinennahe Programmiersprache*

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
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 Binär-Code, der in einem Register gespeichert werden kann (c) 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), • 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), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), • analysieren und erläutern Algorithmen und Programme (A),	*Materialien:* Ergänzungsmaterialien zum Lehrplannavigator (Download LK-Q2-IV.1)
2. Simulation der Phasen eines Compilers (a) Prozesse beim Compiler: • scannen • parsen • übersetzen/interpretieren (b) Arten von Fehlern: • lexikalischer Fehler • syntaktischer Fehler • semantischer Fehler (c) Einordnung der neuen Begriffe in den Gesamtkontext der formalen Sprachen • Automaten • Grammatiken • Sprachen		*Beispiel:* Scanner, Parser und Interpreter einer Konstruktionssprache für geometrische Figuren Anhand einer einführenden Folienpräsentation werden die Begrifflichkeiten definiert. Ein kleiner Ausschnitt einer Pseudo-Programmiersprache zur Konstruktion von Zeichnungen wird betrachtet. Anfänglich besteht der Sprachumfang aus Programmen mit lediglich einer einzigen Zuweisung. Die grundlegenden Schritte eines Compilers werden am Beispiel dieser Grammatik nachvollzogen. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator (Download LK-Q2-IV.2)

2.2 Grundsätze der fachmethodischen und fachdidaktischen Arbeit

Unter Berücksichtigung des Schulprogramms hat die Fachkonferenz Informatik des Städtischen Gymnasiums Petershagen folgende fachmethodische und fachdidaktischen Grundsätze beschlossen. In diesem Zusammenhang beziehen sich die Grundsätze 1 bis 14 auf fächerübergreifende Aspekte, die auch Gegenstand der Qualitätsanalyse sind, die Grundsätze 15 bis 21 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 Schüler/innen erreichen einen Lernzuwachs.
6) Der Unterricht fördert eine aktive Teilnahme der Schüler/innen.
7) Der Unterricht fördert die Zusammenarbeit zwischen den Schülern/innen und bietet ihnen Möglichkeiten zu eigenen Lösungen.
8) Der Unterricht berücksichtigt die individuellen Lernwege der einzelnen Schüler/innen.
9) Die Schüler/innen erhalten Gelegenheit zu selbstständiger Arbeit und werden dabei unterstützt.
10) Der Unterricht fördert strukturierte und funktionale Partner- bzw. Gruppenarbeit.
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.

Fachliche Grundsätze:

15) Der Unterricht unterliegt der Wissenschaftsorientierung und ist dementsprechend eng verzahnt mit seiner Bezugswissenschaft.
16) Der Unterricht ist problemorientiert und soll von realen Problemen ausgehen und sich auf solche rückbeziehen.
17) Der Unterricht folgt dem Prinzip der Exemplarizität und soll ermöglichen, informatische Strukturen und Gesetzmäßigkeiten in den ausgewählten Problemen und Projekten zu erkennen.
18) Der Unterricht ist anschaulich sowie gegenwarts- und zukunftsorientiert und gewinnt dadurch für die Schülerinnen und Schüler an Bedeutsamkeit.
19) Der Unterricht ist handlungsorientiert, d.h. projekt- und produktorientiert angelegt.
20) Im Unterricht werden sowohl für die Schule didaktisch reduzierte als auch reale Informatiksysteme aus der Wissenschafts-, Berufs- und Lebenswelt eingesetzt.
21) Der Unterricht beinhaltet reale Begegnung mit Informatiksystemen.

2.3 Grundsätze der Leistungsbewertung und Leistungsrückmeldung

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

2.3.1 Beurteilungsbereich Klausuren

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

Instrumente:

- **Einführungsphase**: 1. Halbjahr: 1 Klausur
  2. Halbjahr: 2 Klausuren
  Dauer der Klausur: 2 Unterrichtsstunden

- **Grundkurse Q 1**: 2 Klausuren je Halbjahr
  Dauer der Klausuren: 2 Unterrichtsstunden (1. Halbjahr)
  3 Unterrichtsstunden (2. Halbjahr)

- **Grundkurse Q 2.1:** 2 Klausuren
  Dauer der Klausuren: 3 Unterrichtsstunden

- **Grundkurse Q 2.2**: 1 Klausur unter Abiturbedingungen

- Anstelle einer Klausur kann gemäß dem Beschluss der Lehrerkonferenz in Q 1.2 eine Facharbeit geschrieben werden.

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

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

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

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

Klausurdauer im Fach Informatik

Änderung der Klausurdauer für die Sek I und Sek II:

Sek I (Wahlpflichtfach)			
1. Hj		2. Hj	
KL1	Kl2	KL1	Kl2
2 (90)	2 (90)	2 (90)	2 (90)

Einführungsphase	
1. Hj	**2. Hj**
KL1	**KL1**
2 (90)	**2** (90)

Q1 (Grundkurs)				**Q2 (Grundkurs)**		
1. Hj		2. Hj		1. Hj		2. Hj
KL1	Kl2	KL1	Kl2	**KL1**	**Kl2**	**KL1**
2 (90)	2 (90)	3 (135)	3 (135)	**3** (135)	**4** (180)	**5** (225)

Q1 (Leistungskurs)				**Q2 (Leistungskurs)**		
1. Hj		2. Hj		1. Hj		2. Hj
KL1	Kl2	KL1	Kl2	**KL1**	**Kl2**	**KL1**
3 (135)	3 (135)	4 (180)	4 (180)	**5** (225)	**5** (225)	**6** (270)

2.3.2 Beurteilungsbereich Sonstige Mitarbeit

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

Leistungsaspekte

Mündliche Leistungen

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

Praktische Leistungen am Computer

- Implementierung, Test und Anwendung von Informatiksystemen

Sonstige schriftliche Leistungen

- Lernerfolgsüberprüfung durch kurze schriftliche Übungen
  In Kursen, in denen höchstens 50% der Kursmitglieder eine Klausur schreiben, sollten schriftliche Übungen mindestens einmal pro Kurshalbjahr stattfinden, in anderen Kursen entscheidet über die Durchführung die Lehrkraft.
  Schriftliche Übung dauern ca. 20 Minuten und umfassen den Stoff der letzten ca. 4–6 Stunden.
- Bearbeitung von schriftlichen Aufgaben im Unterricht
- Bearbeitung und Vortrag der Hausaufgaben

Kriterien

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

Die Bewertungskriterien stützen sich auf

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

Besonderes Augenmerk ist dabei auf

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

Bei Gruppenarbeiten auch auf

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

Bei Projektarbeit darüber hinaus auf

- die Dokumentation des Arbeitsprozesses,
- den Grad der Selbstständigkeit,
- die Reflexion des eigenen Handelns und

- die Aufnahme von Beratung durch die Lehrkraft.

Grundsätze der Leistungsrückmeldung und Beratung

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

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

Die Leistungsrückmeldung kann

- durch ein Gespräch mit der Schülerin oder dem Schüler,
- durch einen Feedbackbogen,
- durch die schriftliche Begründung einer Note oder
- durch eine individuelle Lern-/Förderempfehlung

erfolgen.

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

3 Entscheidungen zu fach- und unterrichtsübergreifenden Fragen

Die Fachkonferenz Informatik hat sich im Rahmen des Schulprogramms für folgende zentrale Schwerpunkte entschieden:

Zusammenarbeit mit anderen Fächern
Im Informatikunterricht werden Kompetenzen anhand informatischer Inhalte in verschiedenen Anwendungskontexten erworben, in denen Schülerinnen und Schülern aus anderen Fächern Kenntnisse mitbringen können. Diese können insbesondere bei der Auswahl und Bearbeitung von Softwareprojekten berücksichtigt werden und in einem hinsichtlich der informatischen Problemstellung angemessenem Maß in den Unterricht Eingang finden. Da im Inhaltsfeld Informatik, Mensch und Gesellschaft auch gesellschaftliche und ethische Fragen im Unterricht angesprochen werden, soll eine mögliche Zusammenarbeit mit den Fächern Sozialwissenschaften und Philosophie in einer gemeinsamen Fachkonferenz ausgelotet werden.

Vorbereitung auf die Erstellung der Facharbeit
Im ersten Halbjahr des ersten Jahres der Qualifikationsphase werden im Unterricht an geeigneten Stellen Hinweise zur Erstellung von Facharbeiten gegeben. Das betrifft u. a. Themenvorschläge, Hinweise zu den Anforderungen und zur Bewertung. Es wird vereinbart, dass nur Facharbeiten vergeben werden, die mit der eigenständigen Entwicklung eines Softwareproduktes verbunden sind.

Exkursionen
Im Leistungskurs sollte eine fachbezogene Exkursion durchgeführt werden, z.B. ins Heinz-Nixdorf-Forum Paderborn oder zur FH Minden.

4 Qualitätssicherung und Evaluation

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

Das schulinterne Curriculum (siehe 2.1) ist zunächst bis 2017 für den ersten Durchgang durch die gymnasiale Oberstufe nach Erlass des Kernlehrplanes verbindlich. Erstmalig nach Ende der Einführungsphase im Sommer 2015, werden in einer Sitzung der Fachkonferenz Erfahrungen ausgetauscht und ggf. Änderungen für den nächsten Durchgang der Einführungsphase beschlossen, um erkannten ungünstigen Entscheidungen schnellstmöglich entgegenwirken zu können.

Nach Abschluss des Abiturs 2017 wir die Fachkonferenz Informatik auf der Grundlage ihrer Unterrichtserfahrungen eine Gesamtsicht des schulinternen Curriculums vornehmen und ggf. eine Beschlussvorlage für die erste Fachkonferenz des folgenden Schuljahres erstellen.