Schulinterner Lehrplan zum Kernlehrplan für die gymnasiale Oberstufe

Informatik

(Stand: 03.08.2023)

Inhaltsverzeichnis

1 Die Fachgruppe Informatik des Städtischen Gymnasiums Petershagen

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

Derzeit unterrichten rund 70 Lehrerinnen und Lehrer an der Schule. Davon besitzen sechs Lehrkräfte die Fakultas für den Informatikunterricht.

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

Im Informatikunterricht in der Erprobung- und in der Mittelstufe wird in altersstufengerechter Weise unter anderem auf Grundlagen der Algorithmik am Beispiel einer didaktischen Lernumgebung, auf die technische Informatik am Beispiel von Mikrocontrollern und auf Robotik eingegangen. Außerdem werden Grundlagen im Umgang mit informatischen Systemen und insbesondere im Umgang mit persönlichen Daten in vernetzten Systemen vermittelt.

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.

Durch projektartiges Vorgehen, offene Aufgaben und Möglichkeiten, Problemlösungen zu verfeinern oder zu optimieren, entspricht der Informatikunterricht der Oberstufe in besonderem Maße den Erziehungszielen, Leistungsbereitschaft zu fördern, ohne zu überfordern.

Zurzeit stehen für Informatikunterricht drei Computerräume (zwei mit 15 und ein mit 19) zur Verfügung. Alle Arbeitsplätze sind an das schulinterne Rechnernetz angeschlossen, so dass Schüler*innen über einen individuell gestaltbaren Zugang zum zentralen Server der Schule alle Arbeitsplätze der vier Räume zum Zugriff auf ihre eigenen Daten, zur Recherche im Internet oder zur Bearbeitung schulischer Aufgaben verwenden können.

In der Erprobungsstufe wird Informatik im Klassenverbund unterrichtet. Im Wahlpflichtbereich werden in der Regel ein Informatikkurs, in EF zwei bzw. drei Kurse und in der Qualifikationsphase 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 Materialien und Unterrichtsvorhaben, die Evaluation von Lehr- und Lernprozessen sowie die stetige Überprüfung und eventuelle Modifikation des schulinternen Curriculums durch die Fachkonferenz Informatik stellen einen wichtigen Beitrag zur Qualitätssicherung und -entwicklung des Unterrichts dar.

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, Schüler*innen 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

Einführungsphase			
Unterrichtsvorhaben EF-I **Thema**: *Geschichte der digitalen Datenverarbeitung und die Grundlagen des Datenschutzes* *Nutzung von Informatiksystemen* **Zentrale Kompetenzen:** • Argumentieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Informatik, Mensch und Gesellschaft • Informatiksysteme **Inhaltliche Schwerpunkte:** • Geschichte der automatischen Datenverarbeitung • Datenschutz • Aufbau informatischer Systeme (Von-Neumann-Rechner, EVA-Prinzip) • Dateisystem und Internet **Zeitbedarf**: 6 Stunden	*Unterrichtsvorhaben EF-II* **Thema**: *Grundlagen der objektorientierten Analyse, Modellierung und der Programmierung und algorithmischer Grundstrukturen in Java* **Zentrale Kompetenzen:** • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Objektorientierte Modellierung • Syntax und Semantik einer Programmiersprache • Analyse, Entwurf und Implementierung einfacher Algorithmen **Zeitbedarf**: 30 Stunden	*Unterrichtsvorhaben EF-III* **Thema**: *Analyse, Entwurf und Implementierung einfacher Algorithmen mit Hilfe einer grafischen Benutzeroberfläche mit bzw. ohne GUI Builder* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten **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 **Zeitbedarf**: 18 Stunden	*Unterrichtsvorhaben EF-IV* **Thema**: *Modellierung und Implementierung von Klassen- und Objektbeziehungen als Projekt* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Formale Sprachen und Automaten **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Objektorientierte Modellierung • Syntax und Semantik einer Programmiersprache **Zeitbedarf**: 20 Stunden
Summe Einführungsphase: 74 Stunden			

Qualifikationsphase 1 (Grundkurs)				
Unterrichtsvorhaben Q1-I **Thema**: *Wiederholung der objektorientierten Modellierung und Programmierung anhand einer kontextbezogenen Problemstellung* **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**: 8 Stunden	*Unterrichtsvorhaben Q1-II* **Thema:** *Modellierung und Implementierung von Anwendungen mit dynamischen, linearen Datenstrukturen (Stack, Queue und List)* **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**: 20 Stunden	*Unterrichtsvorhaben Q1-III* **Thema**: *Suchen und Sortieren auf linearen Datenstrukturen* **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**: 16 Stunden	*Unterrichtsvorhaben Q1-IV* **Thema**: *Modellierung und Implementierung von Anwendungen mit dynamischen, nichtlinearen Datenstrukturen* **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**: 20 Stunden	*Unterrichtsvorhaben Q1-V* **Thema**: *Sicherheit und Datenschutz in Netzstrukturen. Kryptologie* **Zentrale Kompetenzen:** • Argumentieren • Darstellen und Interpretieren Kommunizieren und Kooperieren **Inhaltsfelder:** • Informatiksysteme Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** • Einzelrechner und Rechnernetzwerke • Kryptologie • Ethik • Nutzung von Informatiksystemen, Wirkungen der Automatisierung **Zeitbedarf**: 10 Stunden
Summe Qualifikationsphase 1 (Grundkurs): 74 Stunden				

Qualifikationsphase 2 (Grundkurs)			
Unterrichtsvorhaben Q2-I **Thema:** *Modellierung und Nutzung von relationalen Datenbanken in Anwendungskontexten. Sicherheit und Datenschutz in Informatiksystemen* **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, Datenschutz **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* **Zentrale Kompetenzen:** • Argumentieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Informatiksysteme • Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** • Einzelrechner und Rechnernetzwerke **Zeitbedarf**: 6 Stunden	*Unterrichtsvorhaben Q2-IV* **Thema**: *Wiederholung und Vertiefung ausgewählter Kompetenzen und Inhalte des ersten Jahres der Qualifikationsphase* **Zeitbedarf**: 6 Stunden
Summe Qualifikationsphase 2 (Grundkurs): 56 Stunden			

Qualifikationsphase 1 (Leistungskurs)				
Unterrichtsvorhaben Q1-I **Thema**: *Wiederholung der objektorientierten Modellierung und Programmierung anhand einer kontextbezogenen Problemstellung* **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 Anwendungen mit dynamischen, linearen Datenstrukturen (Stack, Queue und List)* **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* **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 sowie Sicherheit in Informatiksystemen und Kryptologie* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren Kommunizieren und Kooperieren **Inhaltsfelder:** • Informatiksysteme • Algorithmen • Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** • Einzelrechner und Rechnernetzwerke • Algorithmen in ausgewählten Kontexten • Kryptologie • Ethik • Nutzung von Informatiksystemen,Wirkungen der Automatisierung **Zeitbedarf**: 35 Stunden
Summe Qualifikationsphase 1 (Leistungskurs): 135 Stunden				

Qualifikationsphase 2 (Leistungskurs)			
Unterrichtsvorhaben Q2-I **Thema:** *Modellierung, Implementierung und Nutzung von relationalen Datenbanken in Anwendungs-kontexten. Projektorientierte Softwareentwicklung am Beispiel einer Anwendung mit Datenbankanbindung mit JAVA. Sicherheit und Datenschutz in Informatiksystemen.* **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:** • Analyse, Entwurf und Implementierung von Algorithmen • Datenbanken • Algorithmen in ausgewählten informatischen Kontexten • Syntax und Semantik einer Programmiersprache • Sicherheit und Datenschutz **Zeitbedarf**: 40 Stunden	*Unterrichtsvorhaben Q2-II* **Thema:** *Endliche Automaten und formale Sprachen und Grenzen der Automatisierbarkeit* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Automaten und formale Sprachen • Informatik, Mensch und Gesellschaft **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 **Zeitbedarf**: 27 Stunden	*Unterrichtsvorhaben Q2-IV* **Thema**: *Prinzipielle Arbeitsweise eines Computers* **Zentrale Kompetenzen:** • Argumentieren • Implementieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Informatiksysteme • Formale Sprachen und Automaten **Inhaltliche Schwerpunkte:** • Einzelrechner und Rechnernetzwerke **Zeitbedarf**: 10 Stunden	*Unterrichtsvorhaben Q2-V* **Thema**: *Wiederholung und Vertiefung ausgewählter Kompetenzen und Inhalte des ersten Jahrs der Qualifikationsphase bzw. Umsetzung eines selbstgewählten Projekts* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren • Syntax und Semantik einer Programmiersprache • Nutzung von Informatiksystemen **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten • Informatiksysteme **Inhaltliche Schwerpunkte:** • Vorgehensmodell zur Softwareentwicklung • Objekte und Klassen • Analyse, Entwurf und Implementierung von Algorithmen • Syntax und Semantik einer Programmiersprache • Nutzung von Informatiksystemen **Zeitbedarf:** 20 Stunden
Summe Qualifikationsphase 2 (Leistungskurs): 97 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 Schüler*innen 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.

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 Schüler*innen

- 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 Schüler*innen	Die Schüler*innen • 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 Schüler*innen (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 Codierung 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: 30 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. **2. Objektorientierte Modellierung** (a) Entwurf der Implementationsdiagramme (b) Beziehungen unter Objekten (Vererbung und Assoziation) (c) Zugriffsrechte und Multiplizitäten **3. Implementierung einfacher Java-Klassen** (a) Grundaufbau einer Java-Klasse (b) Deklaration und Initialisierung von Objekten (c) Konstruktoren (d) Attribute und Methoden (e) Kontrollstrukturen (*do*-, *while*- und *for*-Schleifen, *if*-Verzweigungen)	Die Schüler*innen • ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modellieren Klassen mit ihren Attributen, ihren Methoden und 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 Szenarios *Materialien:* BlueJ, Greenfoot, GLOOP, eclipse, Java-Editor, Java-Hamster, UMLed

Unterrichtsvorhaben EF-III:

Analyse, Entwurf und Implementierung einfacher Algorithmen mit Hilfe einer grafischen Benutzeroberfläche mit bzw. ohne GUI Builder

Zeitbedarf: 18 Stunden

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Programmierung einfacher Anwendungen mit Hilfe einer GUI mit bzw. ohne GUI Builder	Die Schüler*innen • 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	*Beispiele*: Zahlenraten, Telefon, Taschenrechner, Memory, etc.
2. Analyse, Entwurf und Implementierung verschiedener Prüfprogramme unter Verwendung linearer Datenstrukturen wie Array		
3. Erarbeitung mehrerer Sortierverfahren (a) Formulierung mehrerer Algorithmen im Pseudocode (Bubble-, Selection-, InsertionSort) (b) Implementierung von BubbleSort (c) Sortierprobleme im Alltag (z.B. Dateisortierung, Tabellenkalkulation, Telefonbuch,usw)		*Beispiele:* (1) Entdecken der Sortierverfahren anhand von Spielkarten (*BubbleSort, SelectionSort, InsertionSort*) (2) Implementieren eines Sortierverfahrens.
4. Lineare Suche (a) Suchaufgaben im Alltag und im Kontext informatischer Systeme (b) Implementierung der linearen Suche		*Beispiele:* (1) Simulationsspiel zur linearen Suche mit Karten. (2) Implementieren des Suchens nach einer Zahl in einem Zahlenarray.

Unterrichtsvorhaben EF-IV: *Modellierung und Implementierung von Klassen- und Objektbeziehungen als Projekt*

Zeitbedarf: 20 Stunden

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Wiederholung der Objektorientierung und Modellierung (a) Entwurf der Implementationsdiagramme (b) Aufstellen der Beziehungen (Vererbung und Assoziation)	Die Schüler*innen • analysieren und modellieren Klassen mit ihren Attributen, ihren Methoden, Vererbung- und Assoziationsbeziehungen (M), • stellen Klassen, Assoziations- und Vererbungsbeziehungen in Diagrammen 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)	*Beispiele*: Modellierung graphischer Objekte mit Java *Materialien: BlueJ, GLOOP, eclipse, Java-Editor, UMLed*
2. Implementierung von Klassen unter Verwendung von Vererbung (a) Analyse und Erläuterung einer Klasse (b) Realisierung der Umsetzung mit und ohne Vererbung (c) Implementierung von Unter- und Oberklassen		*Beispiele:* Implementierung einfacher graphischer Objekte
3. Vertiefung Objektorientierung anhand eines komplexen Projekts (a) Analyse und Modellierung eines komplexen Projektes (b) Implementierung des Projektes (c) Präsentation der Ergebnisse des entwickelten Projektes		*Beispiele:* (1) Implementierung einer komplexen Animation mit graphischen Objekten mit Java (2) Analyse, Entwurf und Implementierung verschiedener Prüfprogramme unter Verwendung linearer Datenstrukturen wie Array und Stack sowie der Methoden der Klassen String und Character

Qualifikationsphase - GRUNDKURS

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 Schüler*innen

- 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 anhand einer kontextbezogenen Problemstellung

Zeitbedarf: 8 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) 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 Schüler*innen • 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:* **Schulverwaltungssystem** • 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. *Beispiel:* ***Medien-Datenbank*** • *Vertiefung Grundprinzipien OOM* • *Dokumentation von Quellcode* • *Objekt- &, Implementationsdiagramme* • *Casten, Subtyping, Polymorphie* • *Abstrakte Klassen und Methoden am Bsp. der Klasse Medium* *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 (Stack, Queue und List)*

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 Schüler*innen • erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), • analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M), • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modifizieren Algorithmen und Programme (I), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme systematisch anhand von Beispielen (I), • stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D).	*Beispiel:* **Erweiterte Warteschlange** Die erweiterte Warteschlange soll von der Queue erben und zusätzliche Funktionen bereitstellen (Daten ausgeben, löschen, zählen usw.) *Beispiel:* **Patientenwarteschlange**
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** 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:* **Rechnungsstapel** In einem Stapel sollen Rechnungen verwaltet 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`.		*Beispiel:* **Erweiterte Liste** Die erweiterte Liste soll von der Liste erben und zusätzliche Funktionen bereitstellen (Daten ausgeben, löschen usw.) *Beispiel*: **Einkaufsliste** *Beispiel*: **Schule** Schüler (Name, Vorname etc.) sollen in einem Kurs (Liste) gespeichert und verwaltet werden.

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

Zeitbedarf: 16 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 Schüler*innen • 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).	Iterative und rekursive Implementierung der binären Suche
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		*Beispiel:* **Karteiverwaltung** Für ein Adressverwaltungsprogramm soll eine Methode zum Suchen einer Adresse geschrieben werden.
3. Untersuchung der Effizienz der verschiedenen Sortierverfahren (a) Grafische Veranschaulichung der Sortierverfahren (b) Untersuchung der Anzahl der Vergleichsoperationen und des Speicherbedarfs bei beiden Sortierverfahren (c) Beurteilung der Effizienz der beiden Sortierverfahren		Effizienz von Algorithmen (Komplexitätsklassen)
4. Rekursion (a) Das Prinzip der Rekursion an Beispielen aus anderen Fachbereichen (Mathematik, Deutsch, Kunst) (b) Anwenden und Darstellen eines rekursiven Funktionsaufrufs		• Fibonacci Zahlen • Fakultät

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

Zeitbedarf: 20 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 Schüler*innen • erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), • analysieren und erläutern Algorithmen und Programme (A), beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • 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 ComparableContent zur Realisierung einer geeigneten Ordnungsrelation (d) Implementierung der Anwendung oder von Teilen der Anwendung inklusive einer sortierten Ausgabe des Baums.		*Beispiel*: **Zeichenzähler** Es soll eine Anwendung entwickelt werden, in der Zeichen gezählt und anschließend deren Anzahl ausgegeben werden. *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. Kryptologie*

Zeitbedarf: 10 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 Schüler*innen • beschreiben und erläutern Topologien, die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken (A), • analysieren und erläutern Eigenschaften und Einsatzbereiche symmetrischer und asymmetrischer Verschlüsselungsverfahren (A), • untersuchen und bewerten anhand von Fallbeispielen die Auswirkungen des Einsatzes von Informatiksystemen, die Sicherheit von Informatiksystemen sowie die Einhaltung der Datenschutzbestimmungen und des Urheberrechts (A), • untersuchen und bewerten Problemlagen, die sich aus dem Einsatz von Informatiksystemen ergeben, hinsichtlich rechtlicher Vorgaben, ethischer Aspekte und gesellschaftlicher Werte unter Berücksichtigung unterschiedlicher Interessenlagen (A), • nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zum Erschließen, zur Aufbereitung und Präsentation fachlicher Inhalte (D).	*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. Beispiel: **Schulnetz und LAN-Party** • Datenübertragung (Paketübermittlung) • Netzwerktopologien • Protokolle • TCP/IP-Schichtenmodell
2. Symmetrische und asymmetrische kryptografische Verfahren (a) Cäsar-, Vigenère-, RSA-Verfahren (b) Diffie-Hellmann-Schlüsselaustauschverfahren		*Beispiel*: **Implementierung von** Cäsar-, Vigenère-, RSA-Verfahren, Diffie-Hellmann-Schlüsselautauschverfahren
3. Moralische Verantwortung (Ethik) (a) Erarbeitung eines einfachen moralischen Bewertungsmaßstabes (Ethik in der Informatik) (b) Anwendung auf Fallbeispiele		**Ethik** am Beispiel der Verantwortungsethik nach Hans Jonas • Erarbeitung an verschiedenen Fallbeispielen wie z.B. autonomes Fahren • Software für Drohnen • Bewertungssystem für chinesische Bürger

Unterrichtsvorhaben Q2-I (Grundkurs):

Zeitbedarf: 24 Stunden

Modellierung und Nutzung von relationalen Datenbanken in Anwendungskontexten
Sicherheit und Datenschutz in Informatiksystemen

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 Schüler*innen • erläutern die Eigenschaften und den Aufbau von Datenbanksystemen unter dem Aspekt der sicheren Nutzung (A), • analysieren und erläutern die Syntax und Semantik einer Datenbankabfrage (A), • analysieren und erläutern eine Datenbankmodellierung (A), • erläutern die Eigenschaften normalisierter Datenbankschemata (A), • bestimmen Primär- und Sekundärschlüssel (M), • ermitteln für anwendungsbezogene Problemstellungen Entitäten, zugehörige Attribute, Relationen und Kardinalitäten (M), • 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), • modifizieren Algorithmen und Programme (I),	*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. *Materialien*: Sqliteman, SQLiteadmin, eclipse, Java-Editor
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	• 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).	*Beispiel:* **Online-Kursheft**
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 eine GUI		*Beispiel:* **Online-Kursheft**
5. Datenschutz, Sicherheit und Urheberrecht (a) Erarbeitung von Grundideen des Datenschutzes (z.B. personenbezogene Daten, informationelle Selbstbestimmung, Datensparsamkeit) (b) Erarbeitung von Grundideen des Urheberrechts anhand eines verbreiteten Lizenzsystems (c) Einblicke in die Datenschutz-Grundverordnung (d) Erarbeitung von Interesse verschiedener Interessengruppen im Hinblick auf Fallbeispiele (e) Erarbeitung von Stellungnahmen zu Fallbeispielen (f) Darstellung der relevanten Aspekte zum Datenschutz, Urheberrecht und zur moralischen Bewertung		*Beispiele:* • autonomes Fahren • Software für Drohnen • Bewertungssystem für chinesische Bürger • Bewertung eines für verschiedene Zielgruppen einsehbaren Online-Kursheftes • Creative-Commons-Lizenzen URL: http://de.creativecommons.org

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

Zeitbedarf: 20 Stunden

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien oder Materialien
1. Endliche Automaten (a) Von realen Automaten zur formalen Beschreibung eines endlichen Automaten (b) Untersuchung, Darstellung und Entwicklung von Transduktoren, Akzeptoren und nichtdeterministischen endlichen Automaten (c) Umwandlung nichtdeterministischer endlicher Automaten in deterministische endliche Automaten	Die Schüler*innen • analysieren und erläutern die Eigenschaften endlicher Automaten einschließlich ihres Verhaltens auf bestimmte Eingaben (A), • analysieren und erläutern Grammatiken regulärer Sprachen (A), • zeigen die Grenzen endlicher Automaten und regulärer Grammatiken im Anwendungszusammenhang auf (A), • ermitteln die formale Sprache, die durch eine Grammatik erzeugt wird (A), • entwickeln und modifizieren zu einer Problemstellung endliche Automaten (M), • entwickeln und modifizieren zu einer Problemstellung endliche Automaten (M), • entwickeln zur akzeptierten Sprache eines Automaten die zugehörige Grammatik (M), • entwickeln zur Grammatik einer regulären Sprache einen zugehörigen endlichen Automaten (M), • modifizieren Grammatiken regulärer Sprachen (M), • entwickeln zu einer regulären Sprache eine Grammatik, die die Sprache erzeugt (M), • stellen endliche Automaten in Tabellen oder Graphen dar und überführen sie in die 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 *Materialien*: Excorciser (eine Software zur Entwicklung von Automaten)
2. Untersuchung und Entwicklung von Grammatiken regulärer Sprachen (a) Erarbeitung der formalen Darstellung regulärer Grammatiken (b) Untersuchung, Modifikation und Entwicklung von Grammatiken (c) Entwicklung von endlichen Automaten zum Erkennen regulärer Sprachen die durch Grammatiken gegeben werden (d) Entwicklung regulärer Grammatiken zu endlichen Automaten		*Beispiele:* Die deutsche Grammatik (Satzgliederungsgrammatik), reguläre Grammatik für Wörter, die bestimmte Zahlen repräsentieren,
3. Grenzen endlicher Automaten Ausblick: Kellerautomat		*Beispiele:* a^nb^n (im Vergleich zu a^nb^m) ; Palindrome
4. Grenzen der Automatisierbarkeit (a) Untersuchung verschiedener nicht in polynomialer Zeit berechenbarer Probleme der Informatik (b) Vorstellung des Halteproblems (c) Unlösbarkeit des Halteproblems (d) Beurteilung des Einsatzes von Informatiksystemen hinsichtlich prinzipieller Möglichkeiten und prinzipieller Grenzen		*Beispiel:* Travelling Salesman Problem usw., Halteproblem, Affenpuzzle

Unterrichtsvorhaben Q2-III (Grundkurs): *Prinzipielle Arbeitsweise eines Computers*

Zeitbedarf: 6 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 Schüler*innen • erläutern die Ausführung eines einfachen maschinennahen Programms sowie die Datenspeicherung auf einer „Von-Neumann-Architektur“ (A), • untersuchen und beurteilen Grenzen des Problemlösens mit Informatiksystemen (A).	*Beispiel:* Addition von 4 zu einer eingegebenen Zahl mit einem Rechnermodell https://vfhcab.eduloop.de/loop/Computerarchitektur Rollenspiel von Neumann-Architektur aus http://informatik-erleben.aau.at (E-H1 – E-H3)

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 Schüler*innen

- 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*

Zeitbedarf: 15 Stunden

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
2. Wiederholung und Erweiterung der objektorientierten Modellierung und Programmierung durch Analyse und Erweiterung eines kontextbezogenen Beispiels (a) Analyse der Problemstellung (b) 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 Schüler*innen • 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: **Schulverwaltungssystem** • 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. Beispiel: **Medien-Datenbank** • Vertiefung Grundprinzipien OOM • Dokumentation von Quellcode • Objekt- &, Implementationsdiagramme • Casten, Subtyping, Polymorphie • Abstrakte Klassen und Methoden am Bsp. der Klasse Medium 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 (Leistungskurs): *Modellierung und Implementierung von Anwendungen mit dynamischen, linearen Datenstrukturen (Stack, Queue und List)*

Zeitbedarf: 25 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) Implementierung der Klasse Queue (d) Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse Queue	Die Schüler*innen • modellieren Klassen mit ihren Attributen, Methoden und Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), • erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), • analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare und nichtlineare Datensammlungen zu (M), • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modifizieren Algorithmen und Programme (I), • implementieren Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (I), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme anhand von Beispielen und mit Hilfe von Testanwendungen (I), • stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D).	Beispiel: **Erweiterte Warteschlange** (auch mit Vererbung) Die erweiterte Warteschlange soll von der Queue erben und zusätzliche Funktionen bereitstellen (Daten ausgeben, löschen, zählen usw.) Beispiel: **Patientenwarteschlange**
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) Implementierung der Klasse Stack (d) Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse Stack		Beispiel: **Erweiterter Stapel** (auch mit 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: **Rechnungsstapel** In einem Stapel sollen Rechnungen verwaltet 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`		Beispiel: **Erweiterte Liste** (auch mit Vererbung) Die erweiterte Liste soll von der Liste erben und zusätzliche Funktionen bereitstellen (Daten ausgeben, löschen usw.) Beispiel: **Einkaufsliste** Beispiel: **Schule** Schüler (Name, Vorname etc.) sollen in einem Kurs (Liste) gespeichert und verwaltet werden.

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

Zeitbedarf: 20 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 (Speicherbedarf, Anzahl der Vergleiche)	Die Schüler*innen • 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).	Iterative und rekursive Implementierung der binären Suche
2. 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)		*Beispiel:* **Karteiverwaltung** Für ein Adressverwaltungsprogramm soll eine Methode zum Suchen einer Adresse geschrieben werden. *Beispiel:* **Test- und Analyseumgebung für Sortieralgorithmen mit einer GUI**
3. Untersuchung der Effizienz der Sortierverfahren „Sortieren durch direktes Einfügen“ und „Quicksort“ auf linearen Listen (a) Grafische Veranschaulichung der Sortierverfahren (b) Untersuchung der Anzahl der Vergleichsoperationen bei beiden Sortierverfahren (c) Beurteilung der Effizienz der beiden Sortierverfahren, untere Schranke für die Laufzeit von Sortieralgorithmen		Effizienz von Algorithmen (Komplexi-tätsklassen)
4. Rekursion (a) Das Prinzip der Rekursion an Beispielen aus anderen Fachbereichen (Mathematik, Deutsch, Kunst) (b) Anwenden und Darstellen eines rekursiven Funktionsaufrufs		*Fibonacci Zahlen, Fakultät*

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

Zeitbedarf: 40 Stunden

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 (Grad, Tiefe, Höhe, Blatt, Inhalt, Teilbaum, Ebene, Vollständigkeit)) (b) Anwendung und Implementierung der Traversierung eines Binärbaums im Pre-, In- und Postorderdurchlauf (c) Modellierung und Implementierung einer Anwendung unter Verwendung der Datenstruktur Binärbaum	Die Schüler*innen • erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), • analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A), • 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.
***2.* Erarbeitung, Implementierung und Verwendung der Datenstruktur binärer Suchbaum im Anwendungskontext** (a) Grafische Darstellung eines binären Suchbaums und Erarbeitung der Struktureigenschaften, Suchen, Einfügen und Löschen von Elementen. (b) Erarbeitung der Attribute und Methoden der generischen Klasse `BinarySearchTree<ContentType>` und des Interfaces `ComparableContent` (c) Modellierung und 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*: **Zeichenzähler** Es soll eine Anwendung entwickelt werden, in der Zeichen gezählt und anschließend deren Anzahl ausgegeben werden. *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.

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** Für das Haus vom Nikolaus existiert ein Eulerweg, aber kein Eulerkreis. 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.
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.

Unterrichtsvorhaben Q1-V (Leistungskurs):

Grundlagen der Netzwerkkommunikation sowie Modellierung und Implementierung von Client-Server-Anwendungen in kontextbezogenen Problemstellungen sowie Sicherheit in Informatiksystemen und Kryptologie. Entwicklung eines Netzwerkspiels

Zeitbedarf: 35 Stunden

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 Protokolls auf der Anwendungsschicht	Die Schüler*innen • 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),	Material: **Aufgabensammlung** Anhand einer Sammlung von Aufgabenblättern erarbeiten die Schüler*innen im Anwendungskontext Grundlagen der Inhaltspunkte „OSI-Referenzmodell“, „Codierung“, „Topologien“, „Routing“ und „Protokolle“ Beispiel: **E-Mail** Betrachtung des TCP/IP-Schichtenmodels Beispiel: **Online-Suchmaschine** Betrachtung des TCP/IP-Schichtenmodels Beispiel: **Schulnetz und LAN-Party** • Datenübertragung (Paketübermittlung) • Netzwerktopologien • Protokolle • TCP/IP-Schichtenmodell
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**

3. Entwicklung eines vollständigen Client-Server-Systems (a) Protokollentwurf, Dialogorientierung (b) Modellierung mittels Entwurfs- und Implementationsdigramm (c) Bedeutung von Nebenläufigkeit (d) Implementierung	• stellen Klassen und ihre Beziehungen grafisch dar (D), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I) • 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). • erläutern Eigenschaften und Aufbau von Informatiksystemen 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: **Messenger-Dienst** (Chat) Versenden von Nachrichten zwischen einzelnen Rechnern, 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**
5. Kryptologie (a) symmetrische und asymmetrische kryptografische Verfahren (Cäsar-, Vigenère-, RSA-Verfahren) als Methoden Daten im Netz verschlüsselt zu übertragen (b) Implementierung symmetrischer Verschlüsselungsverfahren (c) Vertraulichkeit, Integrität, Authentizität in Netzwerken		Beispiel: **Implementierung von** Cäsar-, Vigenère-, RSA-Verfahren Diffie-Hellmann-Schlüsselaustauschverfahren
6. Moralische Verantwortung (Ethik) (a) Erarbeitung eines einfachen moralischen Bewertungsmaßstabes (Ethik in der Informatik) (b) Anwendung auf Fallbeispiele		**Ethik** am Beispiel der Verantwortungsethik nach Hans Jonas • Erarbeitung an verschiedenen Fallbeispielen wie z.B. autonomes Fahren • Software für Drohnen • Bewertungssystem für chinesische Bürger

Unterrichtsvorhaben Q2-I (Leistungskurs):

Modellierung, Implementierung und Nutzung von relationalen Datenbanken in Anwendungskontexten. Projektorientierte Softwareentwicklung am Beispiel einer Anwendung mit Datenbankanbindung mit JAVA. Sicherheit und Datenschutz in Informatiksystemen.

Zeitbedarf: 40 Stunden

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Modellierung von relationalen Datenbanken **a) ER-Diagramm** • Ermittlung von Entitätstypen, Attributen zugehöriger Entitäten, Beziehungstypen und Kardinalitäten in Anwendungssituationen und Modellierung eines Datenbankentwurfs in Form eines ER-Diagramms • Erläuterung und Modifizierung einer Datenbankmodellierung **b) Entwicklung einer Datenbank aus einem Datenbankentwurf** • Modellierung eines relationalen Datenbankschemas 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 • Überprüfung von Datenbankschemata hinsichtlich der 1. bis 3. Normalform und Normalisierung (um Redundanzen zu vermeiden und Konsistenz zu gewährleisten)	Die Schüler*innen • 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 ER-Diagramm grafisch dar (D), • modellieren zu einem ER-Diagramm ein relationales Datenbankschema (M), • bestimmen Primär- und Sekundärschlüssel (M), • 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 verknüpften Tabellen (D), • 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).	*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. Unter http://dokumentation.videocenter.schule.de/old/video/index.html (abgerufen: 01. 02. 2016) *Beispiel*: **Schulbuchausleihe** www.brd.nrw.de/lerntreffs/informatik/structure/material/sek2/datenbanken.php (abgerufen: 01. 02. 2016)
2. 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`)		*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. http://mrbs.sourceforge.net (abgerufen: 01. 02. 2016) *Beispiel:* **Schulverwaltung** In einer Software werden die Schulhalbjahre, Jahrgangsstufen, Kurse, Klassen, Schüler*innen, Lehrende und Noten einer Schule verwaltet.

3. Projektorientierte Softwareentwicklung am Beispiel einer Anwendung mit Datenbankanbindung mit JAVA **a) Analyse einer lebensweltnahen Problemstellung im Hinblick auf die Entwicklung eines Java-Programms mit Datenbankanbindung** • Entwicklung einer Programmidee • Analyse des Problembereichs • Entwicklung eines Anforderungskatalogs für das zu entwickelnde Programm **b) Modellierung einer datenbankgestützten Problemlösung unter Berücksichtigung** • Modellierung der Datenbank (ER-Diagramm, Datenbankschema) • Modellierung der Kontrollklassen und Entwicklung von SQL-Anweisungen entsprechend des Anforderungskatalogs • Modellierung einer GUI **c) Umsetzung des Softwareprojektes** • Umsetzung der Datenbank in einem Datenbankmanagementsystem • Implementierung der Kontrollklassen mit Anbindung an die Datenbank unter Verwendung didaktischer Klassen • Integration in den Prototypen der Benutzeroberfläche **d) Evaluation des Projekts**	• 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 Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M), • stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen (D), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), • ermitteln für anwendungsbezogene Problemstellungen Entitäten, zugehörige Attribute, Relationen und Kardinalitäten (M), • 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), • erläutern Eigenschaften und Aufbau von Datenbanksystemen unter dem Aspekt der sicheren Nutzung (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*: **Sportfestverwaltung** Verwaltung von Aufgaben, Sportereignissen und Ergebnissen des Schulsportfestes *Beispiel*: **Materialverwaltung** Materialien für Vertretungsstunden sollen verwaltet werden *Beispiel:* **Online-Kursheft**
4. Datenschutz, Sicherheit und Urheberrecht (a) Erarbeitung von Grundideen des Datenschutzes (z.B. personenbezogene Daten, informationelle Selbstbestimmung, Datensparsamkeit) (b) Erarbeitung von Grundideen des Urheberrechts anhand eines verbreiteten Lizenzsystems (c) Einblicke in die Datenschutz-Grundverordnung (d) Erarbeitung von Interesse verschiedener Interessengruppen im Hinblick auf Fallbeispiele (e) Erarbeitung von Stellungnahmen zu Fallbeispielen Darstellung der relevanten Aspekte zum Datenschutz, Urheberrecht und zur moralischen Bewertung		*Beispiele:* • autonomes Fahren • Software für Drohnen • Bewertungssystem für chinesische Bürger • Bewertung eines für verschiedene Zielgruppen einsehbaren Online-Kursheftes • Creative-Commons-Lizenzen URL: http://de.creativecommons.org

Unterrichtsvorhaben Q2-II (Leistungskurs): *Endliche Automaten und formale Sprachen und Grenzen der Automatisierbarkeit*

Zeitbedarf: 27 Stunden

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 Schüler*innen • analysieren und erläutern die Eigenschaften endlicher Automaten und Kellerautomaten einschließlich ihres Verhaltens auf bestimmte Eingaben (A), • analysieren und erläutern Grammatiken regulärer und kontextfreier Sprachen (A), • erläutern die Grenzen endlicher Automaten und regulärer Sprachen im Anwendungszusammenhang (A), • ermitteln die formale Sprache, die durch eine Grammatik erzeugt wird (A), • entwickeln und modifizieren zu einer Problemstellung endliche Automaten oder Kellerautomaten (M), • entwickeln zur akzeptierten Sprache eines Automaten die zugehörige Grammatik (M), • entwickeln zur Grammatik einer regulären oder kontextfreien Sprache einen zugehörigen endlichen Automaten oder einen Kellerautomaten (M), • modifizieren Grammatiken regulärer und kontextfreier Sprachen (M), • entwickeln zu einer regulären oder kontextfreien Sprache eine Grammatik, die die Sprache erzeugt (M), • stellen endliche Automaten in Tabellen oder Graphen dar und überführen sie in die jeweils andere Darstellungsform (D), • ermitteln die Sprache, die ein endlicher Automat oder ein Kellerautomat akzeptiert (D),	*Beispiele:* Cola-Automat, Geldspielautomat, Roboter, Zustandsänderung eines Objekts „Auto“, Akzeptor für bestimmte Zahlen, Akzeptor für Teilwörter in längeren Zeichenketten, Akzeptor für Terme
2. Untersuchung und Entwicklung von Grammatiken regulärer Sprachen (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
3. Grenzen endlicher Automaten (a) Erarbeitung von Grenzen endlicher Automaten (b) Entwicklung eines Kellerautomaten (c) Anwendung eines Kellerautomaten zur Syntaxüberprüfung auf Grund-lage von nicht-regulären Grammatiken (d) Implementierung eines Kellerautomaten zur Syntaxüberprüfung		*Beispiele:* Klammerausdrücke, a^nb^n im Vergleich zu $(ab)^n$

4. 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	• 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).	*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.
5. Grenzen der Automatisierbarkeit (a) Untersuchung verschiedener nicht in polynomialer Zeit berechenbarer Probleme der Informatik (b) Vorstellung des Halteproblems (c) Unlösbarkeit des Halteproblems (d) Beurteilung des Einsatzes von Informatiksystemen hinsichtlich prinzipieller Möglichkeiten und prinzipieller Grenzen		*Beispiel:* Travelling Salesman Problem usw., Halteproblem, Affenpuzzle

Unterrichtsvorhaben Q2-III (Leistungskurs): *Prinzipielle Arbeitsweise eines Computers*

Zeitbedarf: 10 Stunden

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien oder Materialien
5. 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 Schüler*innen • erläutern die Ausführung eines einfachen maschinennahen Programms sowie die Datenspeicherung auf einer „Von-Neumann-Architektur“ (A), • untersuchen und beurteilen Grenzen des Problemlösens mit Informatiksystemen (A).	*Beispiel:* Addition von 4 zu einer eingegebenen Zahl mit einem Rechnermodell https://vfhcab.eduloop.de/loop/Computerarchitektur Rollenspiel von Neumann-Architektur aus http://informatik-erleben.aau.at (E-H1 – E-H3)

Unterrichtsvorhaben Q2-IV (Leistungskurs):

Zeitbedarf: 20 Stunden

Wiederholung und Vertiefung ausgewählter Kompetenzen und Inhalte des ersten Jahrs der Qualifikationsphase bzw. Umsetzung eines selbstgewählten Projekts (falls in Q1-V oder Q2-I schon nicht durchgeführt)

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 Schüler*innen und bietet ihnen Möglichkeiten zu eigenen Lösungen.
8) Der Unterricht berücksichtigt die individuellen Lernwege der einzelnen Die 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 Schüler*innen 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 Klausur je Halbjahr
  Dauer der Klausur: 90 Minuten
- **Grundkurse Q 1.1**: 2 Klausuren
  Dauer der Klausuren: 90 Minuten
- **Leistungskurse Q 1.1**: 2 Klausuren
  Dauer der Klausuren: 135 Minuten
- **Grundkurse Q 1.2:** 2 Klausuren
  Dauer der Klausuren: 135 Minuten
- **Leistungskurse Q 1.2**: 2 Klausuren
  Dauer der Klausuren: 180 Minuten
- **Grundkurse Q 2.1**: 2 Klausuren
  Dauer der Klausuren: 180 Minuten
- **Leistungskurse Q 2.1**: 2 Klausuren
  Dauer der Klausuren: 225 Minuten
- **Grundkurse Q 2.2**: 1 Klausur unter Abiturbedingungen (225 Minuten)
- **Leistungskurse Q 2.2**: 1 Klausur unter Abiturbedingungen (270 Minuten)
- 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.

2.3.2 Beurteilungsbereich Sonstige Mitarbeit

Den Schüler*innen 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
- Unterstützung von Mitlernenden

Praktische Leistungen am Computer

- Implementierung, Test und Anwendung von Informatiksystemen

Sonstige schriftliche Leistungen

- Lernerfolgsüberprüfung durch kurze schriftliche Übungen
  Schriftliche Übung dauern ca. 20 Minuten und umfassen den Stoff der letzten ca. 4–6 Stunden.
- Dokumentationen zu einem Unterrichtsvorhaben
- Arbeitsmappe und Aufzeichnungen zu einem durchgeführten Unterrichtsvorhaben
- Bearbeitung von schriftlichen Aufgaben im Unterricht
- Bearbeitung und Vortrag der Hausaufgaben

Die Bearbeitung von Aufgaben des Bundeswettbewerbs Informatik als Langzeithausaufgabe kann in die Note der sonstigen Mitarbeit einfließen.

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,
- Anstrengungsbereitschaft und Konzentration auf die Arbeit,

- Umgang mit neuen Problemen, Beteiligung bei der Suche nach neuen Lösungswegen,
- Umgang mit Arbeitsaufträgen (Hausaufgaben, Unterrichtsaufgaben etc.)
- 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 Schüler*innen 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 Schüler*innen,
- durch einen Feedbackbogen,
- 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 die Schüler*innen 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 Facharbeiten vergeben werden, die entweder mit der eigenständigen Entwicklung eines Softwareproduktes oder Bearbeitung eines anspruchsvollen theoretischen Themas verbunden sind. Unter einem Produkt ist dabei aber nicht nur ein Softwareprodukt zu verstehen, sondern eine Facharbeit kann ebenso gut in einem Flyer, einer Ausstellung oder einem Elternabend münden.

Exkursionen
Im Leistungskurs sollte eine fachbezogene Exkursion durchgeführt werden, z.B. ins Heinz-Nixdorf-Forum Paderborn oder zum Campus Minden der Hochschule Bielefeld.

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.

Die Fachschaft Informatik wird fortlaufend die Ergebnisse aus besuchten Fortbildungsveranstaltungen in die Unterrichtsgestaltung einfließen lassen und bei Bedarf das Curriculum daraufhin anpassen.