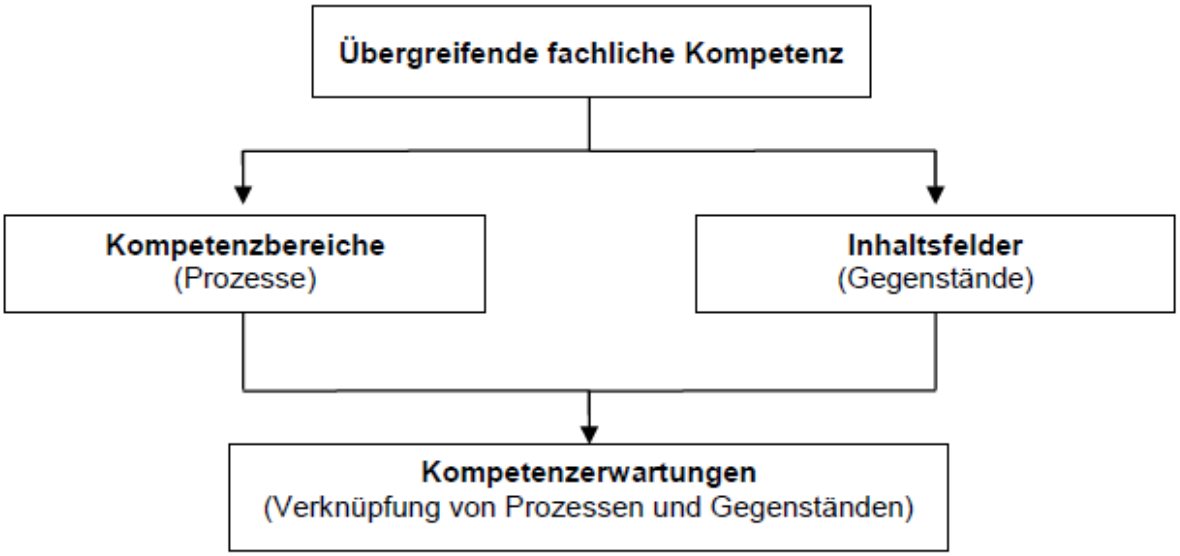
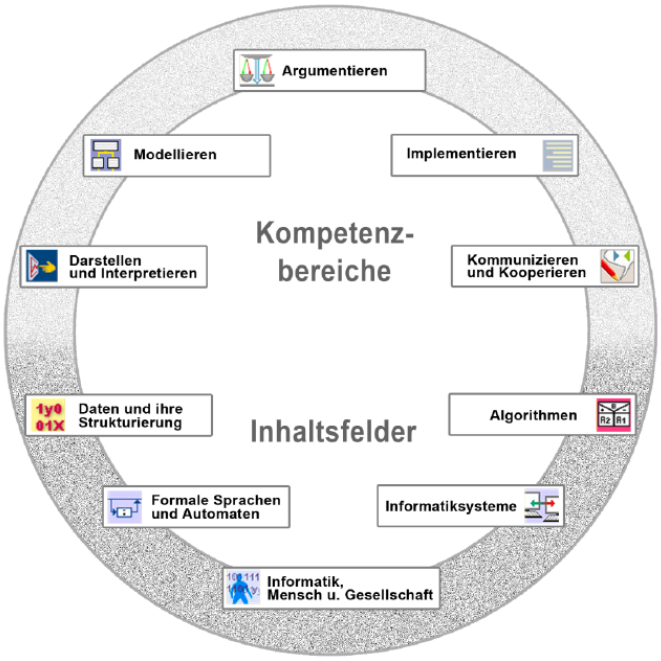
Informatik - Jahrgang Q1/ Q2 **Mai 2015**

Inhaltsfelder und Kompetenzbereiche orientieren sich am ***Kernlehrplan für die Sekundarstufe II Gymnasium /Gesamtschule in Nordrhein-Westfalen (2014)***



Kompetenzbereiche repräsentieren die Grunddimensionen des fachlichen Handelns. Sie dienen dazu, die einzelnen Teiloperationen entlang der fachlichen Kerne zu strukturieren und den Zugriff für die am Lehr-Lernprozess Beteiligten zu verdeutlichen.
Inhaltsfelder systematisieren mit ihren jeweiligen inhaltlichen Schwerpunkten die im Unterricht der gymnasialen Oberstufe verbindlichen und unverzichtbaren Gegenstände und liefern Hinweise für die inhaltliche Ausrichtung des Lehrens und Lernens.
Kompetenzerwartungen führen Prozesse und Gegenstände zusammen und beschreiben die fachlichen Anforderungen und intendierten Lernergebnisse, die auf zwei Stufen bis zum Ende der Sekundarstufe II erreicht werden sollen.



Jahrgang Q1/ Q2

Inhaltsfelder	Kompetenzbereiche und konkretisierte Kompetenzerwartung
Daten und ihre Strukturierung (OOP)	**Die Schülerinnen und Schüler** • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D), • modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M), • verwenden bei der Modellierung geeigneter Problemstellungen Möglichkeiten der Polymorphie (M), • ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M), • stellen die Kommunikation zwischen Objekten grafisch dar (D), • stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen (D), • analysieren und erläutern objektorientierte Modellierungen (A), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I).

Daten und ihre Strukturierung (Datenbanken)	**Die Schülerinnen und Schüler** • ermitteln für anwendungsbezogene Problemstellungen Entitäten, zugehörige Attribute, Relationen und Kardinalitäten (M), • stellen Entitäten mit ihren Attributen und die Beziehungen zwischen Entitäten mit Kardinalitäten in einem Entity-Relationship-Diagramm grafisch dar (D), • modifizieren eine Datenbankmodellierung (M), • modellieren zu einem Entity-Relationship-Diagramm ein relationales Datenbankschema (M), • bestimmen Primär- und Sekundärschlüssel (M), • implementieren ein relationales Datenbankschema als Datenbank (I), • analysieren und erläutern eine Datenbankmodellierung (A), • erläutern die Eigenschaften normalisierter Datenbankschemata (A), • überprüfen Datenbankschemata auf vorgegebene Normalisierungseigenschaften (D), • überführen Datenbankschemata in die 1. bis 3. Normalform (M).

Algorithmen	**<u>Analyse, Entwurf und Implementierung einfacher Algorithmen</u>:** **Die Schülerinnen und Schüler** • analysieren und erläutern Algorithmen und Programme (A), • modifizieren Algorithmen und Programme (I), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D), • entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“ und „Teilen und Herrschen“ (M), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I). **<u>Algorithmen in ausgewählten informatorischen Kontexten</u>** **Die Schülerinnen und Schüler** • erläutern Operationen dynamischer (linearer oder nichtlinearer) Datenstrukturen (A), • implementieren und erläutern iterative und rekursive Such- und Sortierverfahren unterschiedlicher Komplexitätsklassen (Speicherbedarf und Laufzeitverhalten) (I), • beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A), • ermitteln Ergebnisse von Datenbankabfragen über mehrere verknüpfte Tabellen (D). • erläutern das Prinzip der Nebenläufigkeit (A), • analysieren und erläutern Algorithmen und Methoden zur Client-Server-Kommunikation (A), • entwickeln und implementieren Algorithmen und Methoden zur Client-Server-Kommunikation (I).
Formale Sprachen und Automaten	**<u>Syntax und Semantik einer Programmiersprache:</u>** **Die Schülerinnen und Schüler** • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A),

	• interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • analysieren und erläutern die Syntax und Semantik einer Datenbankabfrage (A), • verwenden die Syntax und Semantik einer Datenbankabfragesprache, um Informationen aus einen Datenbanksystem zu extrahieren (I). **<u>Endliche Automaten:</u>** **Die Schülerinnen und Schüler** • analysieren und erläutern die Eigenschaften endlicher Automaten einschließlich ihres Verhaltens bei bestimmten Eingaben (A), • ermitteln die Sprache, die ein endlicher Automat akzeptiert oder ein Kellerautomat (D), • entwickeln und modifizieren zu einer Problemstellung endliche Automaten oder Kellerautomaten (M), • stellen endliche Automaten in Tabellen oder Graphen dar und überführen sie in die jeweils andere Darstellungsform (D), • entwickeln zur Grammatik einer regulären Sprache einen zugehörigen endlichen Automaten oder einen Kellerautomaten (M). **<u>Grammatiken regulärer Sprachen</u>** **Die Schülerinnen und Schüler** • 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), • entwickeln zur akzeptierten Sprache eines Automaten eine zugehörige Grammatik (M), • beschreiben an Beispielen den Zusammenhang zwischen Automaten und Grammatiken (D). **<u>Scanner, Parser und Interpreter für eine reguläre Sprache</u>** **Die Schülerinnen und Schüler** • modellieren und implementieren Scanner, Parser und Interpreter zu einer gegebenen regulären Sprache (I).

	Möglichkeiten und Grenzen von Automaten und Formalen Sprachen **Die Schülerinnen und Schüler** • zeigen/erläutern die Grenzen endlicher Automaten und regulärer Grammatiken im Anwendungszusammenhang auf (A). • analysieren NDEA einschließlich ihres Verhaltens und sind in der Lage diese Systeme NDEA - DEA umzuwandeln (A) (D)
Informatiksysteme	**Einzelrechner und Rechnernetzwerke** **Die Schülerinnen und Schüler** • erläutern die Ausführung eines einfachen maschinennahen Programms sowie die Datenspeicherung auf einer „Von-Neumann-Architektur“ (A), • beschreiben und erläutern Netzwerk-Topologien, die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken (A), • analysieren und erläutern Protokolle zur Kommunikation in einem Client-Server-Netzwerk (A), • entwickeln und erweitern Protokolle zur Kommunikation in einem Client-Server-Netzwerk (M). **Nutzung von Informatiksystemen** **Die Schülerinnen und Schüler** • nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zur Erschließung, Aufbereitung und Präsentation fachlicher Inhalte (D), • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Dateien zur Organisation von Arbeitsabläufen sowie zur Verteilung und Zusammenführung von Arbeitsanteilen unter Berücksichtigung der Rechteverwaltung (K), • wenden eine didaktisch orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I), • entwickeln mit didaktisch orientierten Entwicklungsumgebungen einfache Benutzungsoberflächen zur Kommunikation mit einem Informatiksystem (M). **Sicherheit** **Die Schülerinnen und Schüler**

	• erläutern Eigenschaften, Funktionsweisen und Aufbau von Datenbanksystemen unter dem Aspekt der sicheren Nutzung (A), • analysieren und erläutern Eigenschaften und Einsatzbereiche symmetrischer und asymmetrischer Verschlüsselungsverfahren (A).
Informatik, Mensch und Gesellschaft	**<u>Wirkungen der Automatisierung</u>** **Die Schülerinnen und Schüler** • untersuchen und bewerten anhand von Fallbeispielen Auswirkungen des Einsatzes von Informatiksystemen sowie Aspekte der Sicherheit von Informatiksystemen, des Datenschutzes und des Urheberrechts (A), • untersuchen und bewerten Problemlagen, die sich aus dem Einsatz von Informatiksystemen ergeben, hinsichtlich rechtlicher Vorgaben, ethischer Aspekte und gesellschaftlicher Werte unter Berücksichtigung unterschiedlicher Interessenlagen (A). **<u>Grenzen der Automatisierung</u>** **Die Schülerinnen und Schüler** • untersuchen und beurteilen Grenzen des Problemlösens mit Informatiksystemen (A).
	<u>Kommunizieren und Kooperieren</u> Die folgenden Kompetenzen aus dem Bereich Kommunizieren und Kooperieren werden in allen Unterrichtsvorhaben der Qualifikationsphase vertieft und sollen aus Gründen der Lesbarkeit nicht in jedem Unterrichtsvorhaben separat aufgeführt werden: **Die Schülerinnen und Schüler** • verwenden die Fachsprache bei der Kommunikation über informatische Sachverhalte (K), • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Dateien unter Berücksichtigung der Rechteverwaltung (K), • organisieren und koordinieren kooperatives und eigenverantwortliches Arbeiten (K), • strukturieren den Arbeitsprozess, vereinbaren Schnittstellen und führen Ergebnisse zusammen (K), • beurteilen Arbeitsorganisation, Arbeitsabläufe und Ergebnisse (K), • präsentieren Arbeitsabläufe und -ergebnisse adressatengerecht (K).

Der Leistungskurs zeichnet sich in erster Linie durch eine intensivere und tiefergehende Behandlung einzelner Inhaltsbereiche aus. Die in einem Leistungskurs zusätzlich angestrebten Kompetenzerwartungen wurden farbig markiert.

Beurteilungsbereich „Sonstige Leistungen im Unterricht/ Sonstige Mitarbeit“

Im Beurteilungsbereich „Sonstige Leistungen im Unterricht/ Sonstige Mitarbeit“ können – neben den nachfolgend aufgeführten Überprüfungsformen – vielfältige weitere zum Einsatz kommen, für die kein abschließender Katalog festgesetzt wird. Im Rahmen der Leistungsbewertung gelten auch für diese die oben ausgeführten allgemeinen Ansprüche der Lernerfolgsüberprüfung und Leistungsbewertung. Im Verlauf der gymnasialen Oberstufe ist auch in diesem Beurteilungsbereich sicherzustellen, dass Formen, die im Rahmen der Abiturprüfungen – insbesondere in den mündlichen Prüfungen – von Bedeutung sind, frühzeitig vorbereitet und angewendet werden.

Zu den Bestandteilen der „Sonstigen Leistungen im Unterricht/Sonstigen Mitarbeit“ zählen u. a. unterschiedliche Formen der selbstständigen und kooperativen Aufgabenerfüllung, Beiträge zum Unterricht, von der Lehrkraft abgerufene Leistungsnachweise wie z.B. die schriftliche Übung, von der Schülerin oder dem Schüler vorbereitete, in abgeschlossener Form eingebrachte Elemente zur Unterrichtsarbeit, die z.B. in Form von Präsentationen, Protokollen, Referaten und Portfolios möglich werden. Schülerinnen und Schüler bekommen durch die Verwendung einer Vielzahl von unterschiedlichen Überprüfungsformen vielfältige Möglichkeiten, ihre eigene Kompetenzentwicklung darzustellen und zu dokumentieren.

Der Bewertungsbereich „Sonstige Leistungen im Unterricht/Sonstige Mit-arbeit“ erfasst die im Unterrichtsgeschehen durch mündliche, schriftliche und ggf. praktische Beiträge sichtbare Kompetenzentwicklung der Schülerinnen und Schüler. Der Stand der Kompetenzentwicklung in der „Sonstigen Mitarbeit“ wird sowohl durch Beobachtung während des Schuljahres (Prozess der Kompetenzentwicklung) als auch durch punktuelle Überprüfungen (Stand der Kompetenzentwicklung) festgestellt.

Überprüfungsformen

Die Kompetenzerwartungen des Kernlehrplans ermöglichen eine Vielzahl von Überprüfungsformen. Im Verlauf der gesamten gymnasialen Oberstufe soll – auch mit Blick auf die individuelle Förderung – ein möglichst breites Spektrum der genannten Formen in schriftlichen, mündlichen oder praktischen Kontexten zum Einsatz gebracht werden. Darüber hinaus können weitere Überprüfungsformen nach Entscheidung der Lehrkraft eingesetzt werden. Wichtig für die Nutzung der Überprüfungsformen im Rahmen der Leistungsbewertung ist es, dass sich die Schülerinnen und Schüler zuvor im Rahmen von Anwendungssituationen hinreichend mit diesen vertraut machen konnten. Weitere über die Auflistung hinaus gehende Überprüfungsformen sind möglich.

Überprüfungsform I	Analyse und Eingrenzung einer kontextbezogenen Problemstellung und Entwicklung eines Modells oder Teilmodells mit erläuternden Begründungen der Entwurfsentscheidungen

Überprüfungsform II	Analyse, Erläuterung und Modifikation eines vorgegebenen informatischen Modells sowie die vergleichende Beurteilung unterschiedlicher Entwürfe
Überprüfungsform III	Vollständige oder teilweise Implementation einer bereits modellierten Problemstellung
Überprüfungsform IV	Entwurf und formale Darstellung von Algorithmen zu einer vorgegebenen informatischen Problemstellung
Überprüfungsform V	Analyse und Erläuterung von vorgegebenen Algorithmen oder Programmausschnitten
Überprüfungsform VI	Interpretation gegebener textueller, grafischer oder formaler Darstellungen informatischer Zusammenhänge und deren Überführung in eine andere Darstellungsform
Überprüfungsform VII	Darstellung, Erläuterung und sachgerechte Anwendung von informatischen Begriffen, Verfahren und Lösungsstrategien
Überprüfungsform VIII	Analyse und Beurteilung einer Problemlösung oder eines Informatiksystems nach vorgegebenen oder eigenen Kriterien
Überprüfungsform IX	Analyse und Bewertung des Einsatzes eines Informatiksystems in Bezug auf ethische, rechtliche oder gesellschaftliche Fragestellungen

Inhaltsfelder und Werkzeuge im Detail der Unterrichtsplanungen

Objektorientierte Modellierungen im dreidimensionalen Raum

Die Java-Klassenbibliothek **GLOOP** (Graphics Library for object oriented programming) wurde entwickelt für die Verdeutlichung objektorientierter Zusammenhänge beim Einstieg in die Objektorientierung im Unterricht der gymnasialen Einführungsphase und bietet die Möglichkeit eines didaktisch vereinfachten und intuitiven objektorientierten Zugangs zur dreidimensionalen Grafikprogrammierung mit OpenGL.

In einem virtuellen dreidimensionalen Raum können Objekte positioniert und bewegt werden. Hierfür stehen unter anderem die Klassen "GLKugel", "GLQuader", "GLZylinder", "GLWuerfel", "GLKegel", "GLKegelstumpf", "GLTorus" und "GLPrismoid" zur Verfügung. Sie sind Unterklassen der Klasse "GLObjekt". Durch Setzen einer Textur können die Oberflächen von Objekten gestaltet werden. Um die Objekte sehen zu können, bedarf es allerdings noch einer Kamera und einer Lichtquelle die die Szene ausleuchtet (Objekte der ebenfalls zur Verfügung stehenden Klassen "GLKamera" und "GLLicht"). Für die Programmsteuerung, Benutzereingaben, Ausgaben und Rechnungen stehen die Hilfsklassen "GLMaus", "GLTastatur", "Sys" und "GLVektor" zur Verfügung.

Im Sinne einer Vergleichbarkeit der Kurse sind folgende Beispielprojekte ohne Angabe einer Reihenfolge mit GLOOP zu empfehlen:

//noch frei

Alle benötigten Dateien und Beispiele sind im System vorinstalliert bzw. befinden sich im Fachschaftslaufwerk.

Oberflächenprogrammierung mit Swing und AWT

Bei **Swing** handelt es sich um eine Programmierschnittstelle (API) und Grafikbibliothek zum Programmieren von grafischen Benutzeroberflächen. Das Abstract Window Toolkit (**AWT**) ist Bestandteil der Java Foundation Classes (JFC) und stellt eine Standard-API zur Erzeugung und Darstellung einer plattformunabhängigen grafischen Benutzerschnittstelle (GUI) für Java-Programme dar.

Im Sinne einer Vergleichbarkeit der Kurse sind folgende Beispielprojekte ohne Angabe einer Reihenfolge mit Swing/ AWTzu empfehlen:

//noch frei

Alle benötigten Dateien und Beispiele sind im System vorinstalliert bzw. befinden sich im Fachschaftslaufwerk.

Sequenzierung der Unterrichtsvorhaben
Die Konkretisierungen der Unterrichtsvorhaben und der Aufbau der Unterrichtssequenzen orientieren sich am schulinternen Lehrplan der Fachgruppe Informatik des Konrad- Zuse- Gymnasiums Paderborn und Veröffentlichungen der Gesellschaft für Informatik. Die detaillierten Konkretisierungen, verschiedene Beispiele für die Unterrichtspraxis und die Verankerung der Sequenzen im Verlauf der gesamten Oberstufe werden von der Fachgruppe Informatik des Kopernikus Gymnasium Rheine spezifiziert.

Unterrichtsvorhaben Q1-I (8 Stunden)
Zu einer Problemstellung in einem Anwendungskontext soll eine Java-Anwendung entwickelt werden. Die Problemstellung soll so gewählt sein, dass für diese Anwendung die Verwendung einer abstrakten Oberklasse als Generalisierung verschiedener Unterklassen sinnvoll erscheint und eine Klasse durch eine Unterklasse spezialisiert werden kann. Um die Aufgabe einzugrenzen, können (nach der ersten Problemanalyse) einige Teile (Modellierungen oder Teile von Java-Klassen) vorgegeben werden.

Die Schülerinnen und Schülern erläutern und modifizieren den ersten Entwurf und modellieren sowie implementieren weitere Klassen und Methoden für eine entsprechende Anwendung. Klassen und ihre Beziehungen werden in einem Implementationsdiagramm dargestellt. Dabei werden Sichtbarkeitsbereiche zugeordnet. Exemplarisch wird eine Klasse dokumentiert. Der Nachrichtenaustausch zwischen verschiedenen Objekten wird verdeutlicht, indem die Kommunikation zwischen zwei ausgewählten Objekten grafisch dargestellt wird. In diesem Zusammenhang wird das Nachrichtenkonzept der objektorientierten Programmierung wiederholt.

Unterrichtssequenz	Kompetenzerwartung	Beispiel
Wiederholung und Erweiterung der objektorientierten Modellierung und Programmierung durch Analyse und Erweiterung eines kontextbezogenen Beispiels Analyse der Problemstellung Analyse der Modellierung (Implementationsdiagramm) Erweiterung der Modellierung im Implementationsdiagramm (Vererbung, abstrakte	Die Schülerinnen und Schüler • analysieren und erläutern objektorientierte Modellierungen (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M),	Beispiel: Verwaltung von Kundendaten und Produkten

Klasse) Kommunikation zwischen mindestens zwei Objekten (grafische Darstellung) Dokumentation von Klassen Implementierung der Anwendung oder von Teilen der Anwendung	• 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), • wenden eine didaktisch orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen (D), • stellen die Kommunikation zwischen Objekten grafisch dar (D)	

Unterrichtsvorhaben Q1-II (20 Stunden)
Nach Analyse einer Problemstellung in einem geeigneten Anwendungskontext, in dem Daten nach dem First-In-First-Out-Prinzip bzw. nach dem Last-In-First-Out verwaltet werden, werden der Aufbau von Schlangen bzw. Kellerspeichern am Beispiel dargestellt und die Operationen der Klasse Queue bzw. Stack erläutert. Anschließend werden für die Anwendung notwendige Klassen modelliert und implementiert. Anschließend werden die Anwendungen modifiziert, um den Umgang mit den Datenstrukturen zu üben. Anhand unterschiedlicher Anwendungen werden werden Unterschiede zwischen den Datenstrukturen Schlange und Stapel erarbeitet.

Um einfacher an Objekte zu gelangen, die zwischen anderen gespeichert sind, wird die Klasse List eingeführt und in einem Anwendungskontext verwendet. In mindestens einem weiteren Anwendungskontext wird die Verwaltung von Daten in Schlangen, Stapeln oder Listen vertieft. Modellierungen werden dabei in Entwurfs- und Implementationsdiagrammen dargestellt. Die Einführung der Liste kann auch zu Beginn dieses Unterrichtsvorhabens durch den Vergleich zur Datenstruktur des Feldes eingeführt werden.

In diesen Zusammenhängen werden unterschiedliche Varianten und Möglichkeiten der Umsetzung angesprochen, mindestens aber eine Variante (Schülerlösung oder Vorgabe des Landes NRW) im Unterricht implementiert.

Unterrichtssequenz	**Kompetenzerwartung**	**Beispiel**
Die Datenstruktur Schlange im Anwendungskontext unter Nutzung der Klasse Queue Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen Erarbeitung der Funktionalität der Klasse Queue Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse Queue **Die Datenstruktur Stapel im Anwendungskontext unter Nutzung der Klasse Stack**	**Die Schülerinnen und Schüler** • erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), • analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu	Die Klassen Stack, Queue, Liste sowie Beispiele zur Umsetzung befinden sich im Fachschaftslaufwerk

Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen Erarbeitung der Funktionalität der Klasse Stack Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse Stack **Die Datenstruktur lineare Liste im Anwendungskontext unter Nutzung der Klasse List** Erarbeitung der Vorteile der Klasse List im Gegensatz zu den bereits bekannten linearen Strukturen Modellierung und Implementierung einer kontextbezogenen Anwendung unter Verwendung der Klasse List. **Vertiefung - Anwendungen von Listen, Stapeln oder Schlangen in weiteren Kontexten** **Die Datenstrukturen Stack, Queue und Liste werden implementiert.**	(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).	

Unterrichtsvorhaben Q1-III (24 Stunden)

Anhand kontextbezogener Beispiele werden endliche Automaten entwickelt, untersucht und modifiziert. Dabei werden verschiedene Darstellungsformen für endliche Automaten ineinander überführt und die akzeptierten Sprachen endlicher Automaten ermittelt. An mindestens einem Beispiel wird ein nichtdeterministischer Akzeptor eingeführt als Alternative gegenüber einem entsprechenden deterministischen Akzeptor. Der erkennende Automat und die Überführungsfunktion werden mit entsprechenden Werkzeugen implementiert.

Anhand kontextbezogener Beispiele werden Grammatiken regulärer Sprachen entwickelt, untersucht und modifiziert. Der Zusammenhang zwischen regulären Grammatiken und endlichen Automaten wird verdeutlicht durch die Entwicklung von allgemeinen Verfahren zur Erstellung einer regulären Grammatik für die Sprache eines gegebenen endlichen Automaten bzw. zur Entwicklung eines endlichen Automaten, der genau die Sprache einer gegebenen regulären Grammatik akzeptiert.

Auch andere Grammatiken werden untersucht, entwickelt oder modifiziert. An Beispielen werden die Grenzen endlicher Automaten ausgelotet und der Kellerautomat und die kontextfreie Grammatik entwickelt.

Unterrichtssequenz	**Kompetenzerwartung**	**Beispiel**
Endliche Automaten, Kellerautomaten Vom Automaten in den Schülerinnen und Schülern bekannten Kontexten zur formalen Beschreibung eines endlichen Automaten/ Kellerautomaten Untersuchung, Darstellung und Entwicklung endlicher Automaten/ Kellerautomaten **Untersuchung und Entwicklung von Grammatiken regulärer Sprachen/ kontextfreier Sprachen**	Die Schülerinnen und Schüler • analysieren und erläutern die Eigenschaften endlicher Automaten/ einschließlich ihres Verhaltens auf bestimmte Eingaben (A), • analysieren und erläutern Grammatiken regulärer Sprachen ((A), • zeigen die Grenzen endlicher Automaten und regulärer Grammatiken im Anwendungszusammenhang auf (A), • ermitteln die formale Sprache, die durch	

Erarbeitung der formalen Darstellung regulärer/ kontextfreier Grammatiken Untersuchung, Modifikation und Entwicklung von Grammatiken Entwicklung von endlichen Automaten/ Kellerautomaten zum Erkennen regulärer/ kontextfreier Sprachen die durch Grammatiken gegeben werden Entwicklung regulärer/ kontextfreier Grammatiken zu endlichen Automaten/ Kellerautomaten **Grenzen endlicher Automaten/ Kellerautomaten – Ausblick Turingmaschine/ NDEA**	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). Die Kompetenzerwartung ist auf die Kellerautomaten und die kontextfreien Grammatiken zu übertragen.	

Unterrichtsvorhaben Q2 - I (24 Stunden)
Anhand von Beispielen für Baumstrukturen werden grundlegende Begriffe eingeführt und der rekursive Aufbau binärer Bäume dargestellt. Anschließend werden für eine Problemstellung in einem der Anwendungskontexte Klassen modelliert und implementiert. Dabei werden die Operationen der Datenstruktur Binärbaum thematisiert und die entsprechende Klasse BinaryTree (der Materialien für das Zentralabitur in NRW) der Vorgaben für das Zentralabitur NRW verwendet. Klassen und ihre Beziehungen werden in Entwurfs- und Implementationsdiagrammen dargestellt. Die Funktionsweise von Methoden wird anhand grafischer Darstellungen von Binärbäumen erläutert.
Unter anderem sollen die verschiedenen Baumtraversierungen (Pre-, Post- und Inorder) implementiert werden. Unterschiede bezüglich der Möglichkeit, den Baum anhand der Ausgabe der Bauminhalte via Pre-, In- oder Postorder-Traversierung zu rekonstruieren, werden dabei ebenfalls angesprochen, indem die fehlende Umkehrbarkeit der Zuordnung Binärbaum vs. Inorder-Ausgabe an einem Beispiel verdeutlicht wird.
Eine Tiefensuche wird verwendet, um einen in der Baumstruktur gespeicherten Inhalt zu suchen.
Zu einer Problemstellung in einem entsprechenden Anwendungskontext werden die Operationen der Datenstruktur Suchbaum thematisiert und unter der Verwendung der Klasse BinarySearchTree (der Materialien für das Zentralabitur in NRW) weitere Klassen oder Methoden in diesem Anwendungskontext modelliert und implementiert. Auch in diesem Kontext werden grafische Darstellungen der Bäume verwendet.
Die Verwendung von binären Bäumen und Suchbäumen wird anhand weiterer Problemstellungen oder anderen Kontexten weiter geübt. Es wird ebenfalls auf Ballancierungsmöglichkeiten des AVL- Baumes eingegangen. Bezüge zu Sortierproblemen (Heapsort) sind möglich.

Unterrichtssequenz	**Kompetenzerwartung**	**Beispiel**
Analyse von Baumstrukturen in verschiedenen Kontexten Grundlegende Begriffe (Grad, Tiefe, Höhe, Blatt, Inhalt, Teilbaum, Ebene, Vollständigkeit) Aufbau und Darstellung von binären Bäumen anhand von Baumstrukturen in	Die Schülerinnen und Schüler • erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), • analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A),	*Beispiel:* Termbaum

verschiedenen Kontexten Aufbau und Darstellung von Suchbäumen u. AVL- Bäumen in verschiedenen Kontexten Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen Modellierung eines Entwurfsdiagramms und Entwicklung eines Implementationsdiagramm, grafische Darstellung eines binären Suchbaums und Erarbeitung der Struktureigenschaften Erarbeitung der Klasse BinarySearchTree und Einführung des Interface Item zur Realisierung einer geeigneten Ordnungsrelation Implementierung der Anwendung oder von Teilen der Anwendung inklusive einer sortierten Ausgabe des Baums Übung und Vertiefungen der Verwendung von Binärbäumen oder binären Suchbäumen anhand weiterer Problemstellungen – AVL-Bäume, Heapsort **Ausblick: Graphentheorie**	• ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M), • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M), • verwenden bei der Modellierung geeigneter Problemstellungen die Möglichkeiten der Polymorphie (M), • entwickeln iterative und rekursive Algorithmen unter Nutzung der Konstruktionsstrategien „Modularisierung“ und „Teilen und Herrschen“ (M), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • modifizieren Algorithmen und Programme (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme systematisch anhand von Beispielen (I), • stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D).	*Beispiel*: Ahnenbaum *Beispiel*: Suchbäume zur sortierten Speicherung von Daten *Beispiel*: Entscheidungsbäume *Beispiel*: Codierungsbäume (Bsp.: Morsecode) Die Klassen BinaryTree und BinarySearchTree sowie Beispiele zur Umsetzung befinden sich im Fachschaftslaufwerk

Unterrichtsvorhaben Q2 -II (24 Stunden)

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

In anderen Anwendungskontexten müssen Datenbanken erst noch entwickelt werden, um Daten zu speichern und Informationen für die Beantwortung von möglicherweise auftretenden Fragen zur Verfügung zu stellen. Dafür ermitteln Schülerinnen und Schüler in den Anwendungssituationen Entitäten, zugehörige Attribute, Relationen und Kardinalitäten und stellen diese in Entity-Relationship-Modellen dar. Entity-Relationship-Modelle werden interpretiert und erläutert, modifiziert und in Datenbankschemata überführt. Mit Hilfe von SQL-Anweisungen können anschließend im Kontext relevante Informationen aus der Datenbank extrahiert werden.

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

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

Unterrichtssequenz	**Kompetenzerwartung**	**Beispiel**
Nutzung von relationalen Datenbanken Aufbau von Datenbanken und Grundbegriffe • Entwicklung von Fragestellungen zur vorhandenen Datenbank • Analyse der Struktur der vorgegebenen Datenbank und Erarbeitung der Begriffe Tabelle, Attribut, Datensatz, Datentyp, Primärschlüssel, Fremdschlüssel, Datenbankschema SQL-Abfragen	**Die Schülerinnen und Schüler** • erläutern die Eigenschaften und den Aufbau von Datenbanksystemen unter dem Aspekt der sicheren Nutzung (A), • analysieren und erläutern die Syntax und Semantik einer Datenbankabfrage (A), • analysieren und erläutern eine Datenbankmodellierung (A), • erläutern die Eigenschaften normalisierter Datenbankschemata (A),	Verschiedene Beispiele befinden sich im Fachschaftslaufwerk Beispiel: *Buchungssystem*

• Analyse vorgegebener SQL-Abfragen und Erarbeitung der Sprachelemente von SQL (SELECT (DISTINCT) …FROM, WHERE, AND, OR, NOT) auf einer Tabelle • Analyse und Erarbeitung von SQL-Abfragen auf einer und mehrerer Tabelle zur Beantwortung der Fragestellungen (JOIN, UNION, AS, GROUP BY,ORDER BY, ASC, DESC, COUNT, MAX, MIN, SUM, Arithmetische Operatoren: +, -, *, /, (…), Vergleichsoperatoren: =, <>, >, <, >=, <=, LIKE, BETWEEN, IN, IS NULL) • Vertiefung an einem weiteren Datenbankbeispiel **Modellierung von relationalen Datenbanken** Entity-Relationship-Diagramm • Ermittlung von Entitäten, zugehörigen Attributen, Relationen und Kardinalitäten in Anwendungssituationen und Modellierung eines Datenbankentwurfs in Form eines Entity-Relationship-Diagramms • Erläuterung und Modifizierung einer Datenbankmodellierung • Entwicklung einer Datenbank aus einem Datenbankentwurf • Modellierung eines relationalen Datenbankschematas zu einem Entity-Relationship-Diagramm inklusive der Bestimmung von Primär- und Sekundärschlüsseln 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) **Implementation eines relationalen Datenbankschemas als Datenbank im größeren Umfang mit Kontextbezug**	• bestimmen Primär- und Sekundärschlüssel (M), • ermitteln für anwendungsbezogene Problemstellungen Entitäten, zugehörige Attribute, Relationen und Kardinalitäten (M), • modifizieren eine Datenbankmodellierung (M), • modellieren zu einem Entity-Relationship-Diagramm ein relationales Datenbankschema (M), • bestimmen Primär- und Sekundärschlüssel (M), • überführen Datenbankschemata in vorgegebene Normalformen (M), • verwenden die Syntax und Semantik einer Datenbankabfragesprache, um Informationen aus einen Datenbanksystem zu extrahieren (I), • ermitteln Ergebnisse von Datenbankabfragen über mehrere verknüpfte Tabellen (D), • stellen Entitäten mit ihren Attributen und die Beziehungen zwischen Entitäten in einem Entity-Relationship-Diagramm grafisch dar (D), • überprüfen Datenbankschemata auf vorgegebene Normalisierungseigenschaften (D).	Beispiel: *Schulverwaltung*

Unterrichtsvorhaben Q1 – Folgende Themen werden in die bestehenden Unterrichtsvorhaben der Q1/ Q2I eingebettet

- Das Unterrichtsvorhaben Q1- Zusatz I (16 Stunden) beschäftigt sich mit der Erarbeitung von Such- und Sortieralgorithmen. Der Schwerpunkt des Vorhabens liegt dabei auf der Entwicklung und Implementierung von Sortierverfahren (ebenfalls für lineare Listen und Felder). Hierbei soll auch ein rekursives Sortierverfahren entwickelt werden. Die Implementationen von Quicksort, Insertionsort, Selektionsort, Bubblesort werden analysiert und erläutert.
  Abschließend werden verschiedene Sortierverfahren hinsichtlich der Anzahl der benötigten Vergleichsoperationen und des Speicherbedarfs beurteilt. Es wird mindestens ein spezielles Sortierverfahren mit optimierter Laufzeit von O(n) wie Radixsort oder Bucketsort mit der entsprechenden Laufzeit analysiert. Die untere Laufzeitschranke des allgemeinen Sortierproblems wird hergeleitet. Im Verlauf des Unterrichtsvorhabens Q2-I kann mit Heapsort ebenfalls auf die Thematik eingegangen werden.
- Im Unterrichtsvorhaben Q1/Q2- Zusatz I (10 Stunden) werden die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken erläutert. Eine Umsetzung erfolgt mit der Filius- Umgebung (Beispiele und Unterlagen befinden sich im Fachschaftslaufwerk). Des Weiteren werden Eigenschaften und Einsatzbereiche symmetrischer und asymmetrischer Verschlüsselungsverfahren, Auswirkungen des Einsatzes und die Sicherheit von Informatiksystemen angesprochen. Es werden Problemlagen untersucht und bewertet, die sich aus dem Einsatz von Informatiksystemen ergeben. Dies geschieht hinsichtlich rechtlicher Vorgaben, ethischer Aspekte und gesellschaftlicher Werte unter Berücksichtigung unterschiedlicher Interessenlagen (A). Dieses Unterrichtsvorhaben ist am Ende der Q1 nach der theoretischen Informatik denkbar, kann sich aber auch direkt an das Unterrichtsvorhaben in der Q2 zum Thema Datenbanken anschließen. Die Verschlüsselungsverfahren sind in Anwendungen auch bei der Einführung der Datenstrukturen möglich.
- Im Unterrichtsvorhaben Q1/ Q2- Zusatz II (10 Stunden) werden die prinzipielle Arbeitsweise von Computern sowie die Grenzen der Automatisierbarkeit thematisiert Anhand einer von-Neumann-Architektur und einem maschinennahen Programm wird die prinzipielle Arbeitsweise von Computern verdeutlicht. Ausgehend von den prinzipiellen Grenzen endlicher Automaten liegt die Frage nach den Grenzen von Computern bzw. nach Grenzen der Automatisierbarkeit nahe. Mit Hilfe einer entsprechenden Java-Methode wird plausibel, dass es unmöglich ist, ein Informatiksystem zu entwickeln, dass für jedes beliebige Computerprogramm und jede beliebige Eingabe entscheidet ob das Programm mit der Eingabe terminiert oder nicht (Halteproblem). Anschließend werden Vor- und Nachteile der Grenzen der Automatisierbarkeit angesprochen und der Einsatz von Informatiksystemen hinsichtlich prinzipieller Möglichkeiten und prinzipieller Grenzen beurteilt.

- Das Unterrichtsvorhaben Q2 – Zusatz I (abgängig von der Länge des Schuljahres) bietet die Möglichkeit zur Erstellung und Verarbeitung eigener Internetpräsentationen. Die Schülerinnen und Schüler erstellen direkt mit HTML eigene Inernetseiten in verschiedenen thematischen Zusammenhängen.