Schulinternes Curriculum

für das Fach

Informatik (Sekundarstufe II)

am

Städt. Albert-Martmöller-Gymnasium Witten

Dieses schulinterne Curriculum stellt eine Konkretisierung der Richtlinien und Lehrpläne für das Fach Informatik dar (Ministerium für Schule und Weiterbildung, Wissenschaft und Forschung des Landes Nordrhein-Westfalen: „Richtlinien und Lehrpläne für die Sekundarstufe II – Gymnasium/Gesamtschule in Nordrhein-Westfalen – Informatik“, Ritterbach Verlag, Frechen 1999). Die Schwerpunktsetzung bei der Auswahl der Unterrichtsinhalte orientiert sich zusätzlich an den für die jeweilige Jahrgangsstufe gültigen „Vorgaben zu den unterrichtlichen Voraussetzungen für die schriftlichen Prüfungen im Abitur in der gymnasialen Oberstufe“, die unter http://www.standardsicherung.schulministerium.nrw.de veröffentlicht werden.

Der Informatikunterricht in der Sekundarstufe II ist keine Fortführung des Unterrichts der Kurse, die im Rahmen des Wahlpflichtbereichs II der Sekundarstufe I angeboten werden. Es sind keine informatischen Vorkenntnisse erforderlich.

Als eingeführte Lehrwerke werden die drei Bände „Informatik mit Java“ genutzt (Bernard Schriek: „Informatik mit Java – Eine Einführung mit BlueJ und der Bibliothek Stifte und Mäuse“, Nili Verlag, Werl 2007). Zusätzliche Unterrichtsmittel sind die in der Schule installierten und auch für Schülerinnen und Schüler kostenlos erhältlichen Softwareprodukte. Eingesetzt werden regelmäßig

- die Programmiersprache „Java“ (im Rahmen des Java Software Development Kits JDK)
- die Klassenbibliothek „Stifte und Mäuse“ (SuM)
- die Entwicklungsumgebung „BlueJ“

Weitere Softwareprodukte werden themenbezogen eingesetzt, beispielsweise Editoren für UML-Diagramme oder Struktogramme sowie die Kryptographie-Software „CrypTool“.

In dem nachfolgenden Curriculum wird die Sequenz realisiert, die im Lehrplan als „objektorientiert visuell“ bezeichnet wird.

Einführungsphase

Inhaltsfeld	Inhalte	Vorschläge für Projekte	Verknüpfung mit der Obligatorik
Einführung in die Objektorientierte Programmierung (OOP)	Menü, Aufruf und Funktionen der Programmierumgebung BlueJ; Interaktives Benutzen von Standardklassen (Bildschirm, Stift); Zeichnen auf dem Bildschirm; Erstellen eigener Klassen; Objekte als Instanzen einer Klasse; Bewegungen von Objekten auf dem Bildschirm darstellen (Animationen)	Zeichnen eines Hauses, Autos oder vergleichbarer Figuren	Benutzerführung, Funktionsumfang und Schnittstellen untersuchen; Informatiksysteme kennen lernen und verstehen; die Notwendigkeit des verantwortungsbewussten Umgangs mit Informationen einschätzen
Java-Klassen	Aufbau einer Java-Klasse; Klassendiagramm Elementare Datentypen Attribute, Bezugsobjekte, Dienste (Aufträge und Anfragen) Selbstdefinierte Klassen: private, protected und public, Konstruktoren, Parameterübergabe Erstellen einer HTML-Dokumentation der Klasse (JavaDoc).	(Einführung schrittweise über verschiedene Projekte)	Denkschema entwickeln; Formale Sprachen und Grammatiken untersuchen; Lösungen dokumentieren; Komponenten der Programmiersprache kennen lernen
Kontrollstrukturen und Struktogramme	Bedingte Anweisungen; Schleifen mit Anfangs- und Endebedingung; Mehrfachverzweigung; Zählschleife	Malprogramme, Pfeilwurf	Formen des Strukturierens einsetzen; Lösungsstrategie entwerfen, implementieren
Beziehungen zwischen Objekten	Eigene Bezugsobjekte einer Klasse (Hat-Beziehung), Bezugnahme auf Objekte einer anderen Klasse (Kennt-Beziehung), Vererbung von Oberklasse auf Unterklasse, Spezialisierung.	Stift / Buntstift, Rotkäppchen und der Wolf, Billard, Minigolf	Strukturen erkennen und beurteilen; problembezogene Objekte und ihre Wechselwirkungen spezifizieren
Verkettete Objekte	Verkettung von Objekten durch Nachfolger-Bezugsobjekt in der Klasse, Garbage Collector, Weiterreichung von Nachrichten (Delegation); Aufgabe als arbeitsteiliges Projekt	Zug, Snake	Anforderungen an ein Modell aufstellen; eine erste Lösungsstrategie entwerfen; Allgemeine Strategien verstehen und anwenden
Generalisierung und Polymorphie	Der Begriff der abstrakten Klasse; Generalisierung der Gemeinsamkeiten mehrerer Klassen in eine Oberklasse.	Zeichenprogramm, Weihnachtsbaum	Allgemeine Strategien und Standardlösungen verstehen und

Inhaltsfeld	Inhalte	Vorschläge für Projekte	Verknüpfung mit der Obligatorik
	Abstrakte Dienste, die erst in Unterklassen konkretisiert werden; Dynamische Referenz auf Klassen mit polymorphen Diensten (späte Bindung)		anwenden; Grenzen von Verfahren und Methoden abschätzen
Ereigniskonzept	Ereignisorientierung (Maus, Tastatur); Implementieren eines Ereignisbearbeiters mit Hilfe eines Zustandsdiagramms; Arbeitsweise der Klassen Ereignisanwendung und EBAnwendung	Rechteckknopf, Kreisknopf, Auto, Freihand ereignisorientiert	Allgemeine Strategien verstehen und anwenden; die Entwicklung des Konzepts begreifen und untersuchen
Durchführung eines Softwareprojekts	Strukturierung des Projekts in die Phasen Entwurf, Implementierung und Test; Einbeziehung des MVC-Modells in allen Phasen; Möglichkeiten der Arbeitsteilung evaluieren und bewerten Einsatz des SuM-Programmgenerators	Spielautomat, Kalorienrechner, Knopfjagd	Computergerechte Lösungsvorschläge in Gruppen erarbeiten; Problem analysieren und strukturieren; Lösung implementieren und validieren

Qualifikationsphase

Inhaltsfeld	Inhalte	Vorschläge für Projekte	Verknüpfung mit der Obligatorik
Programmieren mit grafischen Benutzeroberflächen	Wiederholung der wesentlichen Konzepte der ereignisorientierten Programmierung, Erstellen grafischer Benutzeroberflächen mit dem SuM-Programmgenerator	Wechselkurs- oder Temperaturumrechnung	Anforderungen an ein Modell aufstellen; eine erste Lösungsstrategie entwerfen; Benutzerführung, Funktionsumfang und Schnittstellen untersuchen und bewerten
Rekursion als fortgeschrittene Programmstruktur, rekursive Algorithmen	Vor- und Nachteile der Rekursion an ausgewählten Beispielen für mathematische Funktionen (Fakultät, Fibonacci) und Anwendungen	„Türme von Hanoi“, rekursive Zeichenprogramme (Fraktale)	Problemstellungen eingrenzen und Probleme strukturieren; eine erste Lösungsstrategie entwerfen; Formen des Strukturierens einsetzen; allgemeine Strategien und Standardlösungen kennen lernen und anwenden; Programmierkonzepte allgemeiner und spezieller Art verstehen und benutzen; Grenzen von Verfahren und Methoden abschätzen
Statische Datenstrukturen	Das Feld (Array) als statische Datenstruktur: Ein- und zweidimensionale Felder mit primitiven Datentypen und Objekten	Monatskalender, Häufigkeitsanalyse von Buchstaben, Schiffe versenken	Allgemeine Strategien und Standardlösungen kennen lernen und anwenden
Lineare dynamische Datenstrukturen	Liste, Stapel (Stack, auch im Betriebssystem), Schlange (Queue): Anwendung und Implementation der Standardoperationen; Vergleich statischer und dynamischer Datenstrukturen	Wartezimmer-Simulation für eine Arztpraxis mit verschiedenen Warteschlangen,	Lösungskonzepte implementieren und testen; Lösungen dokumentieren; Funktionen und Komponenten von Systemsoftware kennen lernen und beschreiben

Inhaltsfeld	Inhalte	Vorschläge für Projekte	Verknüpfung mit der Obligatorik
		Tankstelle	
Verzweigte dynamische Datenstrukturen, weitere Datenstrukturen	Struktur von „Bäumen“, Traversierung, Anwendung und Implementation von Standardoperationen; Effizienz-Betrachtungen (Landau-Notation), Ausblick auf effizientere Datenstrukturen (z. B. AVL-Baum), Hashtabelle	Morsecodierung, Huffman-Kompression	Lösungen nach vorgegebenen Kriterien bewerten; Problemlösungen optimieren und weiterentwickeln
Suchen und Sortieren	Suchen in Feldern (linear und binär), Suchen und Sortieren in Feldern und Listen: Standardlösungen (Sortieren durch Einfügen, Sortieren durch Auswahl, Bubblesort, …) und besondere Verfahren (Mergesort, Quicksort)	CD-Sammlung sortieren	Lösungskonzepte implementieren und testen; Effizienzuntersuchungen durchführen
Netzstrukturen / Kommunikation zwischen Computern	Client-Server-Architektur, Schichtenmodelle für Netzwerke (OSI, TCP/IP), Analyse von Protokollen der Anwendungsschicht, IP-Adressen und Routing	Erstellen von Routing-Tabellen	Kommunikations- und Vernetzungsstrukturen einordnen; den Einsatz von Informations- und Kommunikationssystemen in verschiedenen gesellschaftlichen Bereichen untersuchen und bewerten
Netzwerk-programmierung I (Clients)	Implementation von Client-Software auf der Basis einfacher Protokolle durch Nutzung der SuM-Klassen „Verbindung“ und „Client-Verbindung“	Daytime-, QOTD- und ECHO-Client, POP3-Client	Problemstellungen eingrenzen und Probleme strukturieren; Anwendungssoftware klassifizieren
Netzwerk-programmierung II (Server)	Implementation mindestens einer Server-Software für die Protokolle aus „Netzwerkprogrammierung I“	Daytime-, QOTD- oder ECHO-Server	Problembezogene Objekte und ihre Wechselwirkungen spezifizieren
Netzwerk-programmierung III (Client-Server)	Modellierung eines eigenen Protokolls (Befehle für den Client und den Server) und anschließende Implementierung	Chat-Server und Chat-Client mit öffentlichen und privaten Nachrichten, Malen im Netzwerk	Anforderungen an ein Modell aufstellen; ein reduziertes Modell für die Problemstellung definieren; ein Lösungskonzept als Denkschema entwickeln
Kryptologie	Symmetrische und asymmetrische	Verschlüsselung und	Die Notwendigkeit des

Inhaltsfeld	Inhalte	Vorschläge für Projekte	Verknüpfung mit der Obligatorik
	Verschlüsselungsverfahren (Cäsar, Vigenère, RSA), Kryptoanalyse bei primitiven Verfahren (insbesondere Cäsar), hybride Verfahren, E-Mail-Verschlüsselung mit Signaturen, Diffie-Hellmann-Schlüsselaustausch-Verfahren	Entschlüsselung bei klassischen Verfahren	verantwortungsbewussten Umgangs mit Informationen einschätzen; den Einsatz von Informations- und Kommunikationssystemen in verschiedenen gesellschaftlichen Bereichen untersuchen und bewerten
Theoretische Informatik: Endliche Automaten und formale Sprachen	Deterministische endliche Automaten (Transduktoren/Akzeptoren, Mealy-/Moore-Automaten) ; Modellierung einfacher Probleme als Automaten; reguläre Ausdrücke, Sprachen und Grammatiken; die Chomsky-Hierarchie; Ausblick: kontextfreie und kontextsensitive Sprachen und Grammatiken	Erkennen gerader/ungerader Binärzahlen, Erkennen von Teilzeichenketten, Aufzugsteuerung	Formale Sprachen und Grammatiken untersuchen; Syntaxregeln und Beschreibungssysteme beurteilen; Automatenmodelle und akzeptierte Sprachen analysieren und beurteilen; den Algorithmenbegriff und den Begriff der Berechenbarkeit verstehen
Datenbanken	Modellierung relationaler Datenbanken, Entity-Relationship-Modell; Normalformen; SQL-Abfragen ; Datenschutz; Datenbankanbindung in Java-Programmen	Bücherei-Verwaltung, Sportverein	Die Notwendigkeit des verantwortungsbewussten Umgangs mit Informationen einschätzen; den Einsatz von Informations- und Kommunikationssystemen in verschiedenen gesellschaftlichen Bereichen untersuchen und bewerten
Technische Informatik	Zahlsysteme (dual, dezimal, hexadezimal); Von-Neumann-Architektur; Vergleich von Hoch- und Maschinensprachen; Simulation von Assemblerprogrammen; Gatter, Schaltnetze- und Schaltwerke	Einfache Rechenbeispiele in Assembler	Die Informationsdarstellung auf der Maschinenebene analysieren; die Entwicklung von Informatiksystemen kennen lernen und verstehen; den Strukturwandel

Inhaltsfeld	Inhalte	Vorschläge für Projekte	Verknüpfung mit der Obligatorik
			in Industrie und Gesellschaft erkennen und beschreiben; die Struktur und Funktionsweise eines Von-Neumann-Rechners analysieren; alternative Rechnerkonzepte und Maschinenmodelle beschreiben