Informatik Jahrgangsstufe 10

Sequenz mit objektorientiertem Ansatz unter Verwendung visueller Werkzeuge (Programmierung mit DELPHI)

Lernsequenz	Unterrichtsinhalte	Verknüpfung mit der Obligatorik
Einführung in die visuelle und ereignisorientierte Programmierung (mit DELPHI) • In Anlehnung an die wesentlichen Merkmale grafischer Benutzerschnittstellen wird eine erste, vorläufige Charakterisierung von „Programmierung" erarbeitet. „Programmieren" bedeutet, das Aussehen, die Bestandteile und den Ablauf eines Computer Programms festzulegen: die Benutzeroberfläche (Formulare, Buttons, ...), die Auswahl der Ereignisse (Button-Click, Tastendruck,..) und die Reaktionen des Programms (= Befehle für den Computer).	• Begriff „Programmierung" • Bestandteile der DELPHI-Programmierumgebung • Formulare: Eigenschaften von Fenstern und Einstellung von Eigenschaften • Ausführen und Testen kleiner Programme	Benutzerführung untersuchen, Problemstellung eingrenzen
• Hierfür stehen in DELPHI vordefinierte Klassen in einer Klassenbibliothek zur Verfügung. Objekte dieser Klassen werden an die konkrete Problemstellung angepasst, indem ihr äußeres Erscheinungsbild (Position, Größe, Beschriftung, Farbe, ...) durch Festlegen (Zuweisungen) von Eigenschaften (= Properties) gestaltet wird und die Ereignismethoden als Reaktion auf diese Ereignisse erstellt werden.	• Eigenschaften von Formen • Zuweisungen (von Eigenschaften) • Reaktionen auf Ereignisse (Farben ändern) • Aufbau eines Delphi-Programms • Maus-Ereignisse und Ereignismethoden • Komponenten: Dialogelemente im Fenster • Schaltflächen (Buttons): Eigenschaften • Bildelemente (Images): Eigenschaften	Programmierkonzept von DELPHI benutzen, erste Lösungsstrategie entwerfen
• Der Unterricht beschränkt sich hierbei auf wenige, typische „visuelle" Komponenten (z.B. Formulare, Buttons, Images) und Ereignisse (Mausklick-, Tastatur-Ereignis, Timer-Ereignis). Zentrale Objekte sind Formulare, die als Klassen in Modulen realisiert werden. Die Grundelemente der klassischen Algorithmik (z.B. Verzweigungen bzw. Entscheidungen (IF..THEN..ELSE)) werden im Zusammenhang mit der Funktionalität der Ereignismethoden dieser Formulare entwickelt. Schöne Beispiele sind hier etwa Animationen mit Icons.	• Ereignisse für Dialogelemente • Entscheidungen (IF..THEN) • Mehrfachentscheidungen (IF..THEN..ELSE) • Animationen • Zeitgeber (Timer): Eigenschaften • Animationen mit Symbolen • Simulation einer Ampelschaltung • Geometrische Figuren (Shapes)	Reduziertes Modell definieren, problembezogene Objekte spezifizieren

Informatik Jahrgangsstufe 10

Lernsequenz	Unterrichtsinhalte	Verknüpfung mit der Obligatorik
• Durch Vererbung wird eine neue Formularklasse erzeugt, die sich von der vorgegebenen Oberklasse durch die verwendeten Komponenten, Ereignisse und Methoden unterscheidet. • Die Problemanalyse ergibt eine Modellierung der jeweiligen Problemkreise in diesen Kategorien. Auf dieser Grundlage wird ein Ablaufschema für die einzelnen Arbeitsschritte entwickelt. Die Beschreibung und Dokumentation der Funktionalität der verwendeten Klassen bewegen sich in diesem Rahmen.		Lösungskonzepte implementieren Denkschema entwickeln, Lösungen dokumentieren
Grafik-Konzept und selbstdefinierte Methoden • Kleinere Projekte mit grafischen Aufgabenstellungen (z.B. Zeichnungen aus zusammengesetzten geometrischen Körpern) können mit Hilfe des Canvas-Konzeptes der DELPHI-Oberfläche bearbeitet werden und lassen leicht eine Binnendifferenzierung zu. • Aus der Sicht des Anwenders visueller Werkzeuge sind die Systemumgebung selbst sowie die verfügbaren kommerziellen Produkte eine Vorlage für die eigenen Konzeptionen. Die Analyse und Bewertung dieser Produkte zeigt die charakteristischen Merkmale auf, die, in reduziertem Umfang, Grundlage des eigenen Projektes (hier: eines eigenen reduzierten Zeichenprogramms) sind. • Die Komponentenklassen besitzen Attribute und Dienste, die ihre Funktionalität beschreiben. Das Prinzip der Datenkapselung verbietet einen direkten Zugriff. Durch Methoden und Eigenschaften ist ein kontrollierter lesender oder schreibender Zugriff möglich (Anfragen oder Aufträge), z.B. Zuweisungen an Stift-Koordinaten, Abfrage der Stift-Position, der Formularbreite usw. . • Erweiterte Aufgabenstellungen (z.B. Reihe von geometrischen Objekten zeichnen) sowie die Grundprinzipien der Modularisierung erfordern die Verwendung eigener, selbst definierter Variablen (Zustandsvariablen oder lokaler Hilfsvariablen) sowie selbst definierter Methoden. Hieran werden weitere Elemente der klassischen Algorithmik (Wiederholschleifen (REPEAT- und WHILE Schleifen)) eingeführt.	• Programmgesteuertes Zeichnen • Canvas-Konzept von Delphi (Übersicht) • Canvas: Hinweise für typische Grafikeinstellungen • Grafikmethoden • Vergleich mit kommerziellen Zeichenprogrammen • Prinzip der Datenkapselung • Selbstdefinierte Methoden • Wiederholung mit Eingangsbedingung (WHILE • Schleife) • Methoden mit Eingabeparameter (call-by-value) • Wiederholung mit Zählvariable (FOR-Schleife) • Wiederholung mit Ausgangsbedingung (REPEAT • Schleife)	Programmierkonzept verstehen Anwendungssoftware analysieren, Software bewerten, Problemlösungen weiterentwickeln Programmierkonzepte verstehen, Strukturen analysieren Programmierkonzepte verstehen
Zeit: ca. bis Ende Januar		

Informatik Jahrgangsstufe 10

Lernsequenz	Unterrichtsinhalte	Verknüpfung mit der Obligatorik
Grundprinzipien der objektorientierten Programmierung • „Objekte reagieren autonom auf Aufträge oder Anfragen (Datenkapselung); Variablen und Methoden sind an die Objekte gebunden. Objekte stehen in Beziehungen zueinander, die analysiert und modelliert werden". Die charakteristischen Merkmale des Sprachkonzeptes werden beschrieben und reflektiert.	• Begriffe: Methoden, Variablen, Objekte, Ereignisse, Eigenschaften, Klassen	Sprachkonzepte charakterisieren und beurteilen
Numerische und alphanumerische Daten • Die Wertebereiche der in den Klassen vorgegebenen Eigenschaften und der Parameterlisten der Methoden vermitteln von Beginn an Kenntnisse über elementare Daten und weitere in der Klassenhierarchie vorgegebene nicht-visuelle Klassen. Die systematische Behandlung numerischer und alphanumerischer Daten knüpft daher an die bisherigen Erfahrungen an. Behandelt werden hier die Typen Integer und Real, die Ein- und Ausgaben über Editier-Felder sowie die erforderlichen Konvertierungen. Anwendungen sind z.B. Rechen-Trainer, Währungs-Umrechner, Glücksspiele mit Hilfe der Random-Funktion,... .	• Ein- und Ausgabefelder • Bezeichnungsfelder: Eigenschaften • Editierfelder: Eigenschaften • Textfelder: Eigenschaften • Umwandlung von Zeichenfolgen in • Zahlen • numerische Datentypen: Integer, Real • Zufallsfunktion (Random) • formatierte Ausgabe • Anwendung: Währungsumrechnung	Standardlösungen kennen lernen und anwenden, Lösungen bewerten, Problemlösungen optimieren
• Zeichenketten (Strings) stellen bereits höhere Datentypen dar, auf denen entsprechende Operationen definiert sind, die anhand kleinerer Aufgaben oder Projekte eingeübt werden (z.B. Reklamebänder, Galgenmännchen, ...)	• Buchstaben und Zeichenketten (Strings) • String-Operationen • Anwendung: Reklamebänder • Anwendung: Galgenmännchen	Standardlösungen kennen lernen und anwenden
• Über elementare Datentypen hinaus und in Anknüpfung an die Zeichenketten wird die Reihung (Array) als ein Mittel eingeführt, um strukturiert mit Daten zu operieren, indem man über eine Indizierung auf einzelne Daten zugreift. Neben Standardalgorithmen der klassischen Algorithmik (z.B. Maximum suchen usw.) können an dieser Stelle weitere Klassen der DELPHIBibliothek eingeführt werden, etwa die ListBox.	• Reihung (Array) • Standardalgorithmen (Suchen, • Minimum, Maximum)	Standardlösungen kennen lernen und anwenden, Lösungen bewerten, Problemlösungen optimieren
• Mit Array lassen sich Standardalgorithmen wie Sortierverfahren simulieren und bezüglich ihres Zeitaufwandes analysieren.	• Sortieralgorithmen (straight insertion, • straight selection, bubble sort) • Zeitaufwands-Analysen	Standardlösungen kennen lernen und anwenden, Lösungen bewerten, Problemlösungen optimieren

Informatik Jahrgangsstufe 11

Lernsequenz	Unterrichtsinhalte	Verknüpfung mit der Obligatorik
(Vertiefung des objektorientierten Ansatzes) **Erzeugen, Verwalten und Entfernen von Objekten** • Objektorientierte Programmierung ergibt ein System von wieder verwendbaren Klassen und Klassenhierarchien (Tools). Die zahlreichen Beziehungen der Objekte und Klassen untereinander lassen sich auf wenige Grundstrukturen reduzieren, die dem Entwurf neuer, eigenständiger Klassen zugrunde liegen: Hat Beziehung (Zerlegung), Ist-Beziehung (Vererbung), Kennt Beziehung (Verbindung);	• Unterschiedliche Objekt-Beziehungen • Hat-Beziehung (Zerlegung) • Ist-Beziehung (Vererbung) • Kennt-Beziehung (Verbindung)	Konzepte realisieren und optimieren, Programmierkonzepte verstehen und benutzen
• Die Hat-Beziehung konkretisiert die „Besitzverhältnisse" der visuellen Komponenten von DELPHI; Ein Formular verwaltet seine Komponenten, das Hauptformular verwaltet alle zur Laufzeit erzeugten Formulare. (Beispiel: Ausgabe des Ergebnisses einer Umrechnung in einem speziellen Ausgabe-Formular).	• Verwaltung von Formularen • Erstellung von Formularen zur Laufzeit	Formen des Strukturierens einsetzen
• Die Analyse bisher verwendeter Objekte und Klassen liefert Grundschemata für den Entwurf neuer, eigenständiger Formularklassen auf der Grundlage der vorhandenen Komponentenbibliothek. Sie verfügen als Unterklassen über die geerbte Funktionalität. (Beispiel: bunte Buttons als Panels realisiert). Die Begriffe der Vererbung (Ist-Beziehung), überschriebener und virtueller Methoden rücken in den Vordergrund.	• Erzeugen von Komponenten durch Ableitung (Vererbung) • Überschreiben und Erzeugen von Konstruktoren • virtuelle Methoden • Taster und Schalter • Ereignismethoden für Taster und Schalter	Lösungskonzept als Denkschema entwickeln und Wechselwirkungen analysieren
• Das Erzeugen, Verwalten und Entfernen von Objekten zur Laufzeit rückt ins Blickfeld der Betrachtung. Für die Realisierung der Hat Beziehungen müssen eigenständige Schablonen entwickelt werden.	• Einbinden von Schalter-Klassen in die Komponentenbibliothek • Polymorphie • Schutzebenen einer Klasse	Sprachkonzepte verstehen und beurteilen; Lösungsstrategien systematisieren

Informatik Jahrgangsstufe 11

Lernsequenz	Unterrichtsinhalte	Verknüpfung mit der Obligatorik
(Vertiefung des objektorientierten Ansatzes) Beziehungen zwischen Objekten und Klassen • Das Konzept der Vererbung gestattet es, die Funktionalität einer geeigneten Oberklasse zu übernehmen, zu modifizieren oder zu erweitern. Neue Unterklassen ergeben sich, indem man die benötigte Funktionalität herausarbeitet, eine vorhandene Klasse sucht, die bereits Teile dieser Funktionalität besitzt (Oberklasse) oder indem man die besonderen Merkmale durch weitere Attribute, neue oder überschriebene Methoden realisiert (Unterklasse). Als Dokumentationstechnik für Klassenhierarchien, Beziehungen und Nachrichtenflüsse werden Coad-Yourdon-Diagramme erstellt. • Techniken wie Überschreiben von Methoden, statische und virtuelle Methoden, spätes Binden unterstützen die Entwicklung von visuellen (Komponenten) und nicht-visuellen Unterklassen	• Dokumentation • Coad-Yourdon-Diagramme • Oberklasse, Unterklasse	Allgemeine Strategien verstehen und anwenden, Strukturen kennen und beurteilen Sprachkonzepte verstehen
Maschinennahe Konzepte • Überblick über die Geschichte der Datenverarbeitung. • Abstrahierend von allen technischen Finessen aktueller Systeme wird der von-Neumann-Rechner als elementare konzeptionelle und technische Grundlage der meisten verfügbaren Computer aufgedeckt. Mit kleineren Anleihen an formale Logik und Stellenwertsystemen (binär, hexadezimal) lassen sich der von-Neumann-Zyklus und die basalen Funktionsprinzipien der programmgesteuerten Informationsspeicherung und -transformation transparent machen. • Durch Festlegung auf eine spezielle Realisierung mit elementarem Befehlssatz definiert man eine Zielmaschine, auf die sich die bekannten Programmierkonstrukte moderner Hochsprachen und Entwicklungsumgebungen durch Compilation abbilden lassen. Als Beispiel dient hier die Assemblersprache zu dem Modell-Assembler ALI • Wesentliche Arbeitsschritte von Compilern (Scanner, Parser, Codierer) lassen sich auf überschaubare Methoden „technischer Übersetzung" formal wohldefinierter Sprachen reduzieren. Betrachtet wird hierbei die Transformationskaskade von der Hochsprache (PASCAL) über eine reduzierte PASCAL-Sprache bis hinunter auf die Ebene des Modell-Computers.	• Geschichte der Datenverarbeitung • von-Neumann-Rechner • Zentraleinheit • elementare Hardware-Begriffe • von-Neumann-Zyklus • Dualzahlensystem • Hexadezimalsystem • Zwiebelarchitektur eines Rechners • Compiler, Assembler • Assembler-Programmierung mit ALI • Komponenten eines Compilers: Scanner, Parser und Codierer • lexikalische, syntaktische, semantische Fehler in Programmen • lexikalische Analyse (Scannery • syntaktische Analyse (Parser)	Einblick in Aufbau und Funktionsweise eines Rechners Veranschaulichung informatischer Systeme Syntaxregeln und Beschreibungssysteme

Informatik Jahrgangsstufe 11

Lernsequenz	Unterrichtsinhalte	Verknüpfung mit der Obligatorik
Theoretische Informatik • Die in der Reduktionskaskade der sukzessiven Compilation schon sehr genau beleuchtete Ausgangsprogrammiersprache wird zur Wurzel einer Sequenz zur theoretischen Informatik: die Theorie der formalen Sprachen über ihre Symbole, zulässigen Sätze und definierenden Grammatiken. Ausgehend von den wesentlichen Arbeitsschritten eines Compilers (Scanner, Parser und Codierer) werden die Strukturen von Sprache untersucht, welche es gestatten, eine Syntaxanalyse und Übersetzung von Sprache vorzunehmen.	• Analyse von Sprache	Logisch-technische Grundlagen transparent machen
• Ausgangssprache sind einfache arithmetische Terme (nur Ziffern und Plus-Operator), die aus syntaktische Korrektheit untersucht werden sollen. Zu diesem Zweck wird das Modell des endlichen Automaten eingeführt, welcher einen arithmetischen Ausdruck als Eingabe erhält und in einem akzeptierenden Zustand endet oder nicht. Weitere Beispiele, wie etwa ein Fahrkartenautomat, verdeutlichen den informatischen Begriff des Zustandes.	• endlicher Automat • Begriff des Zustandes • Simulation eines Scanners für arithmetische Ausdrücke • Simulation eines Fahrkartenautomaten	Computersimulation, Automatenmodelle und akzeptierte Sprachen, Syntaxregeln und Beschreibungssysteme
• Durch eine Modifizierung der Sprache der arithmetischen Terme, nämlich durch das Zulassen von Klammern, gelangt man zu den Grenzen des Konzeptes vom endlichen Automaten und zu dem Begriff des Kellerautomaten, der hier als eine Art Zwischenspeicher für geöffnete Klammern dient.	• Grenzen endlicher Automaten • Kellerautomaten	Analysieren und Bewerten
• Die betrachteten Sprachen können durch Grammatiken definierend beschrieben werden, wobei sich die Frage erhebt, welcher Typ Sprache von den unterschiedlichen Automaten erkannt werden kann. Dies führt bei den endlichen Automaten zu den regulären Sprachen und bei den Kellerautomaten zu den kontextfreien Sprachen.	• Sprachgrammatiken • formale Sprachen • reguläre Sprachen • kontextfreie Sprachen	Formalsprache und Grammatik, Syntaxregeln und Beschreibungssysteme,
• Ziel ist, die Algorithmisierung, die Sprachkonzepte und Automatenmodelle aus theoretisch fundierter Perspektive heraus beurteilen und analysieren zu können. Der Blick wird dafür geöffnet, mit welchen Einschränkungen die Sätze unserer natürlichen Sprache einem maschinellen Verstehen zugänglich sind.	• Grenzen von Kellerautomaten	Grenzen von Algorithmisierung, Problem der Berechenbarkeit

Informatik Jahrgangsstufe 11

Lernsequenz	Unterrichtsinhalte	Verknüpfung mit der Obligatorik
Höhere Datenstrukturen • Algorithmen mit den zugehörigen Daten werden immer an die verwendeten Klassen in Form von Methoden und Zustandsvariablen gebunden. Unter Verwendung abstrakter Klassen lassen sich auch die Klassen Liste und Baum zu Klassenhierarchien erweitern, in die Strukturen wie Keller, Binärbaum usw. integriert sind. Durch die Behandlung endlicher Automaten und Kellerautomaten gelangt man zu den höheren Datenstrukturen Liste (für das Eingabeband) und Keller (für den Speicher des Kellerautomaten).	• Datenstruktur Liste • Listenoperationen	Problembezogene Objekte und ihre Wechselwirkungen spezifizieren
• Die Datenstruktur Liste lässt sich aus der Klassenbibliothek von DELPHI (Tlist) ableiten bzw. selbst definieren. Entsprechende Operationen lassen sich in Pfeildiagrammen veranschaulichen. Beispiel hierzu ist eine kleine Vokabelverwaltung.	• Listenstrukturen und indizierter Zugriff • Vokabelverwaltung • Darstellung von Listenstrukturen	Sprachkonzepte verstehen
• Die Datenstruktur Keller lässt sich wiederum aus der Datenstruktur Liste als Spezialfall ableiten und entsprechende Methoden werden überschrieben.	• Datenstruktur Keller (Stapel) • Stapelstrukturen	Sprachkonzepte verstehen
• Aus der Klassenhierarchie der Liste lässt sich als weiterer Typ der Datentyp Schlange ableiten. Eine Anwendung hierzu ist z.B. die Warteschlange an einer Tankstelle und dessen .Verwaltung. -	• Datenstruktur Schlange • Simulation und Verwaltung einer Waschstraße	Problembezogene Objekte und ihre Wechselwirkungen spezifizieren

Informatik Jahrgangsstufe 12

Lernsequenz	Unterrichtsinhalte	Verknüpfung mit der Obligatorik
Höhere Datenstrukturen • Ausgehend von der Syntax-Analyse arithmetischer Terme stellt sich die Frage danach, wie ein Rechner Ausdrücke unter Beachtung der Regel „Punkt- vor Strichrechnung" auswerten kann und führt auf diese Weise zur Datenstruktur Binärbaum, speziell zur Datenstruktur Termbaum. Binärbäume können mit Hilfe der Methoden der objektorientierten Programmierung wiederum als Klassen implementiert werden.	• Datenstruktur Baum • Baumstrukturen • Binärbäume • Wurzel, Knoten, Blätter, Söhne, Äste • rekursive Baumdurchläufe (inorder, preorder, postorder)	Lösungskonzepte entwickeln und optimieren, Algorithmen beurteilen
Exkurs: Objekte speichern in Delphi • Zur vertiefenden Wiederholung der höheren Datenstrukturen stellt man sich zur Aufgabe, etwa einen Binärbaum in einem sequentiellen Text-File zu speichern.	• Daten speichern und laden • Datenströme	Sprachkonzepte verstehen
Exkurs: Rekursion • Anknüpfend an rekursive Baumdurchläufe werden weitere typische rekursive Anwendungen betrachtet: die Türme von Hanoi, Fibonacci-Zahlen, Pascal'sches Dreieck.	• Rekursion • Rekursionsverankerung und rekursiver Aufruf	Problemlösungen entwickeln und erweitern
Theoretische Informatik • In einem kleineren Projekt wird zur vertiefenden Wiederholung das „Game of life" in seiner Theorie besprochen und auf dem Bildschirm simuliert. Unter anderem stellt sich bei diesem Spiel das Problem der Nicht-Entscheidbarkeit, da der Ausgang der Eingangspopulation nicht vorherzusehen, also nicht berechenbar ist.	• Nicht-Entscheidbarkeit • Spiel-Simulation	Computersimulation

Informatik Jahrgangsstufe 12

Lernsequenz	Unterrichtsinhalte	Verknüpfung mit der Obligatorik
• Anknüpfend an die Unzulänglichkeit von Kellerautomaten erhebt sich das Problem, diesen Automaten weiter zu modifizieren, um etwa die Erkennung korrekt unterstrichener Wörter zu ermöglichen. Neben der Möglichkeit, einen Kellerautomaten mit zwei Kellern auszustatten, gelangt man zum theoretischen Konzept der Turing-Maschine und zum zentralen Begriff der Berechenbarkeit. An kleineren Aufgaben werden Zustandsgraphen entwickelt und Turing -Programme geschrieben und getestet.	• Game of life • Berechenbarkeit • Turing-Maschine • Turing-Programme	Grenzen von Algorithmisierung, Problem der Berechenbarkeit
Netzstrukturen • Auf der Basis von elementarem Datenaustausch zwischen zwei Rechnern lassen sich grundsätzliche Strategien der Kommunikationsorganisation und ihrer technischen Absicherung verdeutlichen. Möglichkeiten zur Fehlererkennung und -beseitigung durch geeignete Codierungen und Nutzung von Redundanz von Informationen werden in Modellen vermittelt. • Im Blickpunkt stehen globale Netzwerke, wobei die Erstellung von Produkten im WWW bei geeigneten Entwicklungswerkzeugen Ausgangspunkt ist, um im Unterricht dahinter liegende informatische Strukturen transparent zu machen. • Zum Unterrichtsgegenstand werden die Fragen nach Zugriffsrechten, Prioritätsregelungen und konfliktfreier Ressourcenfreigabe bei konkurrierenden Anforderungen. • Eine sorgsame Analyse der Auswirkungen der neuen Technologien auf unsere Alltagswelt, auf die Geschäfts- und Berufssphäre und den allgemeinen Wandel der Kommunikationsstrukturen gehört zu den Aufgaben der Analyse und Bewertung durch die Informatik.	• Netzstrukturen • Prinzipien des Datenaustausches • Schichtenmodell • globale Netzwerke • WWW • lokale Netzwerke • Zugriffsrechte • Konflikte • Synchronisation • gesellschaftliche Auswirkungen	Handlungsorientierung Entwicklungen im WWW, anwendungs-, produkt- und handlungsorientiert Aufbau und Struktur lokaler Netzwerke analysieren und beherrschen lernen Auswirkungen neuer Informations- und Kommunikationstechnologien analysieren und bewerten