3 Qualifikationsphase (Jahrgangsstufe 12)

LK 12.1 Objektorientierte Softwareentwicklung

Begründung

Die objektorientierte Softwareentwicklung baut auf die in der Einführungsphase behandelten Standardalgorithmen auf, vermeidet aber den Nachteil prozeduraler Konzepte, der in der Trennung des Programmablaufs von der Datensicht liegt, was letztlich zu mangelnder Wiederverwendbarkeit und zu hohen Kosten bei Wartung und Pflege führt.

Ziel des objektorientierten Ansatzes ist es, Programmmodule zu gestalten, die

- Realitätsaspekte einheitlich darstellen,
- zu überschaubaren Programmen führen und
- wiederverwendbar sind.

Daten und die darauf erlaubten Operationen werden als Einheit gesehen. Dazu erzeugt man eine Klasse, in der sowohl die Kennzeichen der Daten (Attribute) als auch die dazugehörenden Operationen (Methoden) zusammengefasst werden. Die Daten unterliegen einer Kapselung, d.h. auf sie kann nur über spezifische Methoden dieser Klasse zugegriffen werden.

Die Schülerinnen und Schüler lernen in diesem Kurs, einen Anwendungsfall zu analysieren sowie durch Abstraktion und Reduktion Objekte und Klassen mit ihren relevanten Eigenschaften und Methoden zu bilden. Strukturelle Beziehungen, Datenrepräsentation und zeitliche Abläufe werden modellhaft dargestellt und die geplanten Strukturen des Softwaresystems in ein lauffähiges Programm übersetzt.

Die in der Einführungsphase behandelten Datenstrukturen werden durch abstrakte Datentypen in Form von Klassen modelliert und implementiert. In Attributen werden die Daten gespeichert. Algorithmen erscheinen als Methoden der jeweiligen Klasse.

Verbindliche Unterrichtsinhalte	**Stichworte und Hinweise**
Grundkonzepte der Softwareentwicklung	Problemanalyse, Modellierung, Entwurf, Implementation, Test, Wartung Dokumentationstechniken für Benutzer- und Systemdokumentation
Objektmodell	Objekt als Exemplar einer Klasse Kommunikation über Botschaften Modellierung mit der grafischen Modellierungssprache UML (Anwendungsfall-, Klassen-, Sequenzdiagramm)
Klassen	Attribute als Datenstruktur zur Repräsentierung der Information über ein Objekt Methoden als Schnittstellen für den Zugriff auf Attribute und zum Nachrichtenaustausch Kapselung, Vererbung, Polymorphie Objektbeziehungen
Verwaltung von Objekten	Behälter für Objekte: Feld, verkettete Listen einfache Such- und Sortierverfahren (rekursiv und

	iterativ): – lineares- und binäres Suchen – Sortieren durch Auswahl, Sortieren durch Einfügen Quick-Sort

Fakultative Unterrichtsinhalte	**Stichworte und Hinweise**
binäre Bäume	Binärer Suchbaum, Termbaum
Backtracking	Suchen durch Backtracking
Komplexität von Algorithmen	Zeitkomplexität
Suche in Texten	Volltextsuche, Mustererkennung

Arbeitsmethoden der Schülerinnen und Schüler / Hinweise und Erläuterungen

Objektorientierte Programmierung (OOP) ergibt ein System von wieder verwendbaren Klassen und Klassenhierarchien (Klassenbibliotheken). Die zahlreichen Beziehungen der Objekte und Klassen untereinander lassen sich auf wenige Grundstrukturen reduzieren:

- Hat-Beziehung (Zerlegung, Aggregation),
- Ist-Beziehung (Vererbung) und
- Kennt-Beziehung (Verbindung, Assoziation).

Die objektorientierten Sprachen unterstützen durch das Konzept der Vererbung nur die Realisierung von Ist-Beziehungen. Objekt-Beziehungen lassen sich mittels Referenzen umsetzen. Für die Realisierung der Hat-Beziehungen müssen dagegen u.U. eigenständige Klassen zur Verwaltung von Objekten entwickelt werden. Solche Klassen müssen auch das Suchen nach Objekten mit bestimmten Eigenschaften zulassen.

Es ist naheliegend für die Verwaltung von Objekten zunächst die Datenstruktur „Feld“ heranzuziehen, auf der Operationen wie „Einfügen“, „Suchen“ und „Löschen“ von Objekten definiert sind. Während sich in der Jahrgangsstufe 11 das Feld im Wesentlichen auf die Verwaltung einfacher Datentypen beschränkt, wird nun die Verwaltung von Objekten im Vordergrund stehen. In diesem Zusammenhang sollen auch Sortieralgorithmen problematisiert werden.

Die Datenstruktur „Feld“ erweist sich aber in vielerlei Hinsicht als wenig flexibel, so dass oftmals dynamische Datenstrukturen herangezogen werden müssen.

Methodisch kann dabei zunächst von der Datenstruktur „Stapel“ ausgehend das Prinzip der „verketteten Listen“ als Repräsentant dynamischer Datenstrukturen erarbeitet werden. Da das Einfügen sowie Löschen von Objekten beim Stapel an der gleichen Position erfolgt (LIFO), reduzieren sich die Operationen im Wesentlichen auf zwei Operationen (push, pop). In gleicher Weise kann die Datenstruktur „Schlange“ betrachtet werden.

Schließlich wird das allgemeine Prinzip und die Struktur einer einfach verketteten Liste durch grafische Veranschaulichung der Methoden „Einfügen“ (auch an beliebiger Stelle), „Suchen“ und „Löschen“ verdeutlicht und in einer geeigneten Programmiersprache als Klasse implementiert. Die allgemeine Datenstruktur „Liste“ soll bei der Bearbeitung eines Anwendungsfalls aus der Praxis zum Einsatz kommen.

In Erweiterung dessen kann die allgemeine Liste durch das Prinzip der Vererbung zu einer sortierten Liste angepasst werden („Sortierte Liste“ als Spezialfall der „Liste“ durch Anpassen der Methode „Einfügen“). Eine sortierte Liste kann durch Einfügen an der richtigen Stelle oder

Sortieren einer ungeordneten Liste erfolgen. Auch lassen sich Stapel und Schlange als Spezialisierung der allgemeinen Datenstruktur „Liste“ beschreiben.

Fakultativ kann der Baum als effiziente dynamische Datenstruktur mit hierarchischem Ordnungsprinzip betrachtet werden. Ebenso können Effizienzuntersuchungen für die verschiedenen Datenstrukturen durchgeführt und die Grenzen von Verfahren abgeschätzt werden.

Die Komplexität von Objektmodellen erfordert die Erarbeitung von Analyse- und Entwurfstechniken für Klassenhierarchien, Beziehungen und Nachrichtenflüssen. Hier hat sich die Beschreibungssprache UML (Unified Modelling Language) als Standard herausgebildet. Diese soll bei der objektorientierten Modellierung konsequent eingesetzt und auch zu Dokumentationszwecken herangezogen werden.

Die Schülerinnen und Schüler analysieren einen Anwendungsfall aus der Informationstechnik und erstellen einen objektorientierten Systementwurf in UML-Notation. Mittels einer objektorientierten Programmiersprache setzen die Schülerinnen und Schüler den Entwurf in ein Programm um. Sie testen ihr Programm systematisch und führen Fehlerkorrekturen durch. Sie reflektieren und dokumentieren fortlaufend ihre Arbeitsergebnisse und präsentieren ihre Problemlösung.

Querverweise

Modellbildung: Physik, Chemie