

Probeklausur

Einführung in die Informatik

Universität: Technische Universität Berlin
Kurs/Modul: Einführung in die Informatik
Bearbeitungszeit: 120 Minuten
Erstellungsdatum: September 19, 2025



Zielorientierte Lerninhalte, kostenlos!
Entdecke zugeschnittene Materialien für deine Kurse:

<https://study.AllWeCanLearn.com>

Einführung in die Informatik

Bearbeitungszeit: 120 Minuten.

Aufgabe 1.

- (a) Entwickeln Sie ausgehend von einer Problemstellung ein abstraktes Modell für ein kleines Bibliotheksverwaltungssystem. Definieren Sie mindestens drei Klassen, nennen Sie zentrale Attribute und zentrale Methoden.
- (b) Skizzieren Sie das textuelle Klassendiagramm der definierten Klassen und geben Sie Beziehungen zueinander an. Bezeichnen Sie Typen der Beziehungen (1:1, 1:n, n:m) und nennen Sie exemplarische Attributwerte.
- (c) Zeigen Sie auf, wie Vererbung in diesem Modell genutzt werden könnte, indem Sie eine Basisklasse „Medium“ ableiten zu „Buch“, „Zeitschrift“ und „DVD“. Nennen Sie Basisklasse und zwei abgeleitete Klassen, listen Sie relevante Felder auf und erläutern Sie die Rolle der Konstruktoren.

Aufgabe 2.

- (a) Beschreiben Sie, wie Zeichenketten in einem typischen Programmiermodell dargestellt werden und welche grundlegenden Operationen sinnvoll sind (Länge, Teilzeichenkette, Vergleich).
- (b) Erläutern Sie, wie Arrays und Listen als Datenstrukturen organisiert sind, insbesondere im Hinblick auf Speicherlayout und Zugriff.
- (c) Vergleichen Sie zwei grundlegende Suchstrategien in Sequenzen: lineare Suche und binäre Suche. Diskutieren Sie Sortieranforderungen, Effizienz und Einsatzszenarien.

Aufgabe 3.

- (a) Geben Sie eine einfache Klassenhierarchie an: Basisklasse **Person** mit Attributen **Name**, **Alter**; Unterklassen **Student** mit **Matrikelnummer** und **Angestellter** mit **Personalnummer**. Diskutieren Sie, welche Felder sinnvoll in der Basisklasse platziert werden und welche in den Unterklassen.
- (b) Erläutern Sie den Begriff der Polymorphie. Nennen Sie ein Beispiel, wie dieselbe Methode in unterschiedlichen Unterklassen verschieden implementiert werden kann und wie dies zur flexibleren Nutzung führt.
- (c) Beschreiben Sie den Zweck einer abstrakten Klasse **Medium** mit einer abstrakten Methode **ausgabe()** und diskutieren Sie Vorteile der Nutzung abstrakter Klassen im Modell.

Aufgabe 4.

- (a) Geben Sie eine kurze Übersicht zum Aufbau eines Computers: CPU, Arbeitsspeicher, Cache, Ein-/Ausgabesystem. Erläutern Sie die Rolle von Bits und Bytes in der Repräsentation von Informationen.
- (b) Skizzieren Sie, wie ein Betriebssystem Prozesse verwaltet und welche Scheduling-Strategien typisch sind.
- (c) Beschreiben Sie zwei gängige Arten, Zahlen im Rechner darzustellen (Ganzzahlen, Fließkommazahlen) und nennen Sie typische Vor- bzw. Nachteile der jeweiligen Darstellung.

Lösungen

Bearbeitungszeit: 120 Minuten.

Aufgabe 1.

(a) Lösungsvorschlag: Abstraktes Modell eines kleinen Bibliotheksverwaltungssystems.

- Klassen (mindestens drei): - Medium (abstrakt) - Buch (von Medium abgeleitet) - Zeitschrift (von Medium abgeleitet)

- Zentrale Attribute: - Medium: id (string), titel (string), jahr (int) - Buch: autor (string), isbn (string), seiten (int) - Zeitschrift: issn (string), band (int), auflage/ausgabe (int)

- Zentrale Methoden: - Medium: getTitel(), getJahr(), getId(), ausgabe() (abstrakt) - Buch: ausgabe() (overridden), weitere Hilfsmethoden z.B. getAutor(), getIsbn() - Zeitschrift: ausgabe() (overridden), ggf. getIssn(), getBand()

- Begründung der Auswahl: - Die Basisklasse Medium fasst gemeinsame Merkmale zusammen (Titel, Jahr, Identifikator). - Buch und Zeitschrift liefern spezifische Felder, die Medium ergänzen. - Die Abstraktion erleichtert das generische Handling (z. B. in einer Bibliothek) von verschiedenen Medium-Arten.

(b) Textuelles Klassendiagramm (Beziehungen) und exemplarische Attributwerte.

- Klassen und Beziehungen (textuell beschrieben): - Bibliothek –1:N–> Medium - Eine Bibliothek verwaltet mehrere Medien. - Medium besitzt keine direkte Abhängigkeit zu anderen Klassen im Modell (Ausleihe kann separat modelliert werden).

- Typen der Beziehungen: - Bibliothek zu Medium: 1:n (Eine Bibliothek enthält viele Medien) - Innerhalb der abgeleiteten Klassen besteht keine direkte 1:1- oder 1:n-Beziehung, sofern man weitere Klassen für Ausleihe oder Exemplare ergänzt.

- exemplarische Attributwerte (Beispiele): - Medium: id = "M001", titel = "Einführung in die Informatik", jahr = 2020 - Buch: autor = "Musterautor, Max", isbn = "978-3-16-148410-0", seiten = 320 - Zeitschrift: issn = "1234-5678", band = 12, auflage = 3

- Hinweis: - Für ein vollständiges Diagramm ließe sich auch eine UML-Darstellung verwenden (z. B. in PlantUML). Die textuelle Beschreibung reicht hier, um die Beziehungen klar zu machen.

(c) Vererbung: Basisklasse Medium ableiten zu Buch, Zeitschrift und DVD. Basisklasse und zwei abgeleitete Klassen; relevante Felder; Rolle der Konstruktoren.

- Basisklasse: Medium - Attribute: titel (string), jahr (int), id (string) - Rolle der Konstruktoren: Der Konstruktor der Basisklasse initialisiert die gemeinsamen Felder (titel, jahr, id). Er sorgt dafür, dass alle abgeleiteten Klassen diese Felder konsistent setzen können.

- Abgeleitete Klassen (Beispiel zwei davon mit Feldern): - Buch - Zusätzliche Felder: autor (string), isbn (string), seiten (int) - Konstruktor (Beispielidee): Buch(titel, jahr, id, autor, isbn, seiten) und ruft dabei den Basisklassenkonstruktor mit titel, jahr, id auf. - Zweck der Konstruktoren: Sichere Initialisierung der gemeinsamen Felder in der Basisklasse und anschließende Initialisierung der spezifischen Felder in der abgeleiteten Klasse. - DVD - Zusätzliche Felder: regisseur (string), dauerMin (int) - Konstruktor (Beispielidee): DVD(titel, jahr, id, regisseur, dauerMin) und Aufruf des Basisklassenkonstruktors für titel/jahr/id. - (Optional) Zeitschrift als weitere abgeleitete Klasse (falls gewünscht): - Zusätzliche Felder: issn (string), band (int), ausgabe (int)

- Rolle der Konstruktoren (Zusammenfassung): - Der Basisklassenkonstruktor initialisiert die gemeinsamen Felder. - Die abgeleiteten Konstruktoren initialisieren zusätzlich die spezifischen Felder. - Dadurch wird Redundanz vermieden und die Konsistenz der gemeinsamen Attribute gewährleistet. - Falls Polymorphie genutzt wird, ermöglicht die Basisklasse eine einheitliche Behandlung aller Medium-Typen, während die abgeleiteten Klassen jeweils spezialisierte Details

liefern.

Aufgabe 2.

(a) Lösungsvorschlag.

- Darstellung von Zeichenketten: - Typischerweise als Folge von Zeichen gespeichert (in vielen Sprachen als Objekt/String-Typ realisiert). - Speicherrepräsentation kann je nach Sprache variieren (z. B. UTF-8, UTF-16). In vielen Sprachen wird intern ein eigener String-Typ verwendet, der Länge, Kapazität und Inhalte verwaltet.

- Grundlegende Operationen: - Länge: `length()` oder `size()` - Teilzeichenkette/Substring: `substring(i, j)`, `substr(start, length)` - Vergleich: `equals()` bzw. `compareTo()` bzw. Operatoren wie `==`, `<`, `>` je nach Sprache - Konkatination, Suche (`IndexOf`), Umwandlungen (`toLowerCase`, `toUpperCase`) usw.

(b) Lösungsvorschlag.

- Arrays - Speicherlayout: zusammenhängender Speicherblock; feste Größe - Zugriff: indexbasierter Zugriff $O(1)$ - Vorteile: hohes Paging-Verhalten, geringe Overheads - Nachteile: feste Größe, kostenintensive Neuordnungen bei Größenänderungen

- Listen (z. B. verkettete Listen) - Speicherlayout: Knoten mit Wert und Zeigern auf Nachfolger - Zugriff: sequentiell; $O(n)$ im schlechtesten Fall - Vorteile: einfache dynamische Größenerweiterung, kein Realloc wie bei Arrays - Nachteile: schlechtere Cache-Nutzung, iteratorenbasierter Zugriff

- Zusammenfassung: - Arrays eignen sich für festen Speicherbedarf und schnellen Zugriff. - Listen sind flexibel in der Größe und eignen sich, wenn häufige Einfügungen/Entfernungen erfolgen, aber langsamer beim direkten Zugriff.

(c) Lösungsvorschlag.

- Zwei grundlegende Suchstrategien in Sequenzen: - Lineare Suche (Sequential Search): - Funktionsweise: Durchlaufen der Sequenz von Anfang bis Ende. - Sortierabhängigkeit: keine Sortierung nötig. - Komplexität: durchschnittlich $O(n)$; im Worst-Case $O(n)$ - Einsatzszenarien: unsortierte oder regelmäßig veränderte Datenstrukturen, kleine Datensätze.

- Binäre Suche (Binary Search): - Funktionsweise: Nur gültig, wenn die Sequenz sortiert ist; middle-Element vergleichen, dann Halbierung. - Sortierabhängigkeit: Sequenz muss sortiert sein. - Komplexität: $O(\log n)$ - Einsatzszenarien: große, sortierte Datensätze; häufige Suchabfragen.

- Sortieranforderungen: - Für binäre Suche ist eine Sortierung der Daten Voraussetzung. - Falls häufige Suchoperationen stattfinden, kann eine sortierte Struktur (z. B. Baum, Index) sinnvoll sein. - Für kleine oder sich häufig ändernde Datensätze kann lineare Suche bevorzugt werden.

Aufgabe 3.

(a) Lösungsvorschlag.

- Gegebene Klassenhierarchie: - Basisklasse `Person` mit Attributen `Name`, `Alter` - Unterklassen: - `Student` mit `Matrikelnummer` - `Angestellter` mit `Personalnummer`

- Diskussion, welche Felder sinnvoll in der Basisklasse platziert werden: - Sinnvoll in Basis: `Name`, `Alter`, ggf. Kontaktinformationen (E-Mail, Telefonnummer), da diese Merkmale für alle Personen gelten. - Spezifische Felder (`Matrikelnummer`, `Personalnummer`) gehören in die jeweiligen Unterklassen, da sie der jeweiligen Rolle eindeutig zugeordnet sind. - Falls gewünscht, kann man zusätzlich ein gemeinsames Feld `Id` oder `Kundennummer` in der Basisklasse halten, sofern

es für alle gilt.

(b) Lösungsvorschlag.

- Begriff der Polymorphie: - Polymorphie bedeutet, dass Objekte unterschiedlicher Klassen über denselben Basistyp behandelt werden können. - Beispiel: Eine Liste von **Person**-Objekten kann sowohl **Student**- als auch **Angestellter**-Instanzen enthalten. Eine Methode **beschreibe()** kann in jeder Unterklasse verschieden implementiert werden, wird aber über den Basistyp aufgerufen. - Beispiel (mindestens eine Methode mit mehrfacher Umsetzung): - In Java/C++, y) die gleiche Methode **beschreibe()** implementieren: - **Student.beschreibe()** könnte zusätzliche Felder wie Matrikelnummer ausgeben. - **Angestellter.beschreibe()** könnte Personalnummer und Abteilung ausgeben. - Aufruf über Basistyp-Referenzen/Pointer führt zur Auswahl der jeweiligen Implementierung (virtuelle/ überschreibende Methoden).

(c) Lösungsvorschlag.

- Zweck einer abstrakten Klasse **Medium** mit einer abstrakten Methode **ausgabe()**: - Eine abstrakte Klasse kann nicht direkt instanziiert werden; sie definiert ein gemeinsames Interface. - Die abstrakte Methode erzwingt in allen abgeleiteten Klassen die Bereitstellung einer konkreten Implementierung von **ausgabe()**. - Vorteile: - Förderung von Polymorphie; Code, der mit **Medium**-Referenzen arbeitet, kann verschiedene konkrete Typen behandeln. - Klarere Trennung von gemeinsamen und spezifischen Merkmalen. - Reduktion von Duplizierung durch zentrale Definition gemeinsamer Schnittstellen.

Aufgabe 4.

(a) Lösungsvorschlag.

- Aufbau eines Computers (Kurzüberblick): - CPU (Zentraleinheit) - Arbeitsspeicher (RAM) - Cache-Speicher (Zwischenspeicher auf mehreren Ebenen) - Ein-/Ausgabesystem (I/O) - Rolle von Bits und Bytes: - Bits sind die kleinsten informationstragenden Bausteine (0 oder 1). - Bytes (typisch 8 Bits) bilden grundlegende Speichereinheiten zur Darstellung von Zahlen, Zeichen und Adressen. - Alles im Rechner wird letztlich als Bitfolgen interpretiert (z. B. Zahlen, Zeichen, Anweisungen).

(b) Lösungsvorschlag.

- Betriebssystemprozesse und Scheduling: - Prozessverwaltung: Erzeugen, Terminieren, Kontextwechsel, Ressourcenverwaltung. - Scheduling-Strategien (typische Beispiele): - FCFS (First-Come-First-Served): einfache Warteschlange, faire Reihenfolge, aber potenziell lange Wartezeiten. - Round-Robin: Zeit-Scheiben (Quantums), gute Reaktionszeiten, verhindert Long-Running-Tasks. - Shortest-Job-First (SJF) oder Priority-basiert: effektiv, aber ggf. Starvation-Risiko, erfordert Schätzungen der Laufzeiten bzw. Prioritäten. - Ziel des Scheduling: faire Ressourcennutzung, geringe Wartezeiten, hohe Systemdurchsatz.

(c) Lösungsvorschlag.

- Zwei gängige Arten, Zahlen im Rechner darzustellen: - Ganzzahlen (Integer) - Darstellung: meist vorzeichenbehaftet (z. B. Zweierkomplement) oder zusätzlich per Bias, je nach Architektur. - Vorteile: exakte Darstellung, einfache Arithmetik, deterministische Grenzen. - Nachteile: begrenzter Wertebereich, Überlaufgefahr. - Fließkommazahlen (Floating-Point) - Darstellung: IEEE-754-Standard (typisch 32 Bit Einzel- oder 64 Bit Doppelpräzision) - Aufbau: Vorzeichen, Exponent, Mantisse - Vorteile: großer dynamischer Bereich, adäquate Darstellung vieler Werte

mit variabler Genauigkeit - Nachteile: Rundungsfehler, Endlichkeit von Repräsentationen (genaue Ganzzahldarstellung nicht immer möglich), Sonderfälle (NaN, Inf) - Zusammenfassung der Vor- bzw. Nachteile: - Ganzzahlen eignen sich, wenn Genauigkeit und deterministische Wertebereiche wichtig sind (Zähler, Indizes). - Fließkommazahlen eignen sich, wenn Werte mit sehr großem Dynamikumfang benötigt werden (Wissenschaftliche Berechnungen), jedoch muss mit Rundungsfehlern gerechnet werden.