technische universität
dortmund

Fakultät für Informatik
Lehrstuhl 14
Lars Hildebrand

Praktikum zu

Einführung in die Informatik für LogWings, WiMas und MedPhys

Wintersemester 2019/20

Übungsblatt 9

Besprechung:
06.–10.01.2020
(KW 2)

Vorbereitende Aufgaben

Aufgabe 9.1: Bäume

In dieser Aufgabe sollen Sie sich mit Bäumen beschäftigen.

a) Quizfragen:

i) Wie nennt man den Knoten ohne Vorgänger?

ii) Wie nennt man einen Knoten x, der direkter Nachfolger eines Knoten y ist?

iii) Wie nennt man einen Knoten ohne Nachfolger?

b) Gegeben sei ein binärer Baum in Form eines Arrays:

	5	3	7	2	4	9	8

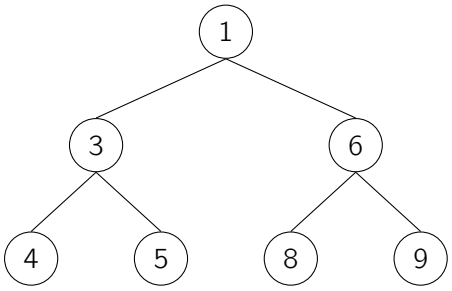
Geben Sie die grafische Repräsentation des Baumes an:

c) Handelt es sich bei diesem Baum um einen Heap? Wenn nein, warum nicht?

Präsenzaufgaben

Aufgabe 9.2: Heaps

Nun wollen wir unsere Kenntnisse über Heaps anwenden, um eine sortierte Folge von Zahlen auszugeben. Leeren Sie den angegebenen Heap vollständig, indem Sie, wie in der Vorlesung (Kapitel 5.2) beschrieben, das jeweils kleinste Element entfernen. Geben Sie eine Baumrepräsentation des Heaps nach jedem Entfernen des Minimums und nach jedem Tausch während der **Heapify**-Operation an:



Aufgabe 9.3: Einführung in die Objektorientierung

Nun wollen wir ein Fahrzeug mit Hilfe objektorientierter Programmierung modellieren.

a) Mit welchem Schlüsselwort können neue Objekte mit ihrem Konstruktor instanziiert werden?

---

b) Gegeben sei die Klasse **Vehicle**. Diese finden Sie auf der Veranstaltungsseite. Die Klasse beschreibt ein beliebiges Fahrzeug. Ergänzen Sie die vermerkten Stellen im Quellcode der Klasse, indem Sie den Inhalt zuerst in eine eigene Quelldatei im Paket **blatt09** übernehmen und dann die Implementierung ergänzen.

```
1  package blatt09;
2
3  public class Vehicle {
4
5      private int wheels;
6      private String fuel;
7
8      public Vehicle(int inWheels, String inFuel) {
9          /* Ergaenzen: Initialisierung der Attribute */
10     }
11
12     public int getWheels() {
13         /* Ergaenzen: Die Anzahl der Reifen zurueckgeben */
14     }
15
16     public void print() {
17         /* Ergaenzen: Den verwendeten Treibstoff und
18          * die Anzahl der Reifen ausgeben */
19     }
20 }
```

Aufgabe 9.4: Klassen selbst definieren

In dieser Aufgabe sollen Sie PKW mit Hilfe objektorientierter Programmierung modellieren. Erstellen Sie eine Klasse namens **Car** im Paket **blatt09**. Diese soll keine **main**-Methode besitzen. Die Klasse repräsentiert ein PKW. Deklarieren Sie anschließend die String-Attribute **manufacturer** und **model** sowie ein Integer-Attribut namens **horsePower**, die Hersteller, Modell und die Pferdestärken des Motors repräsentieren sollen. Deklarieren Sie zudem folgende Konstante: `public static final double WATT_PER_HORSEPOWER = 735.5;`

Schreiben Sie einen öffentlichen Konstruktor, der drei Parameter für die drei Attribute entgegen nimmt und diese auf die übergebenen Werte setzt.

Schreiben Sie eine öffentliche Methode namens **getPower()**, die die Pferdestärken des Autos in Watt umrechnet und als **double** zurückgibt. Schreiben Sie eine Methode namens **print()**, die den Hersteller, das Modell und die Leistung ausgibt.

Hinweis: Sie können Ihre geschriebene Klasse noch nicht testen. Bearbeiten Sie dafür Aufgabe 9.5.

Aufgabe 9.5: Klassentest

Nun wollen wir Ihre zuvor geschrieben Klasse testen, indem wir eine separate Testklasse schreiben. Schreiben Sie eine Klasse **VehicleTest** im Paket **blatt09** mit einer **main**-Methode. Deklarieren und initialisieren Sie in der **main**-Methode folgende Objekte:

- Drei Fahrzeuge mit den Namen „bike“, „plane“, „spaceship“, der Reifenzahl 2, 3 und 0 sowie der Treibstoffart „feet“, „gasoline“ und „deuterium“
- Drei Autos, zwei davon vom Hersteller „Ford“ und den Modellen „Fiesta“ und „Model T“ und einen „Toyota Aygo“. Denken Sie sich interessante Werte für die Pferdestärke und die Variablenbezeichner aus.

Rufen Sie die **print()**-Methoden der Objekte auf und lassen Sie die Leistung der Motoren in Watt berechnen und geben Sie diese auch aus.

Ergänzende Aufgaben

Aufgabe 9.6: Vergleichende Methoden

In dieser Aufgabe wollen wir Methoden programmieren, die die bisher geschriebenen Objekte miteinander vergleichen können. Ergänzen Sie die beiden Klassen **Vehicle** und **Car** um jeweils eine öffentliche Methoden mit dem Namen **compare** und dem Rückgabetyp **int**. Die Methoden sollen jeweils ein Objekt des gleichen Typs entgegennehmen und das Objekt, auf dem die **compare**-Methode aufgerufen wurde mit dem übergebenen Objekt vergleichen: Das Vergleichskriterium für Vehikel ist ihre Reifenzahl, während das Vergleichskriterium für Autos ihre Motorleistung ist. Die Methoden sollen 1 zurückgeben, wenn das aufrufende Objekt mehr Reifen bzw. Leistung hat, als das andere. Die Methoden sollen -1 zurückgeben, wenn das aufrufende Objekt weniger Reifen bzw. Leistung hat als das andere. Die Methoden sollen 0 zurückgeben, wenn die Objekte gleich viele Reifen bzw. Leistung haben.

Tastatureingabe und Importieren von Bibliotheken

Diese Seite soll Ihnen eine Übersicht über das Einlesen von **Eingaben** über die Tastatur und das **Importieren** von anderen Programmen in Ihr eigenes Programm geben.

Sie können mit der Anweisung **import java.util.Scanner;** ein bereits geschriebenes Programm zur Behandlung von Eingaben aus der **Java-Standard-Bibliothek** in Ihrem Programm verfügbar machen.

Ein **Scanner** muss vor der Verwendung **instanziiert** werden. Dies ist ein Vorgang, mit dem Sie sich im **objektorientierten** Teil der Veranstaltung noch genauer beschäftigen werden. Wenn Sie einen **Scanner** verwenden wollen, müssen Sie eine neue Variable vom Typ **Scanner** anlegen und mit der Anweisung **new Scanner(System.in)** instanziieren. Mit dieser Variablen können Sie anschließend die Standard**eingabe** auslesen.

Ein leerer Klassenrumpf, der in der **main**-Methode einen Scanner verwenden will, würde also folgendermaßen aussehen:

```
1  package blatt09;
2
3  import java.util.Scanner;
4
5  public class Input {
6      public static void main(String[] args) {
7          Scanner input = new Scanner(System.in);
8      }
9  }
```

Sie können anschließend mit folgenden Anweisungen Daten auslesen:

```
int number = input.nextInt();           /* liest einen Integer ein */
long number = input.nextLong();         /* liest einen Long ein */
float number = input.nextFloat();       /* liest einen Float ein */
double number = input.nextDouble();     /* liest einen Double ein */
String text = input.nextLine();         /* liest Text ein */
```

Führt ein Programm eine solche Programmzeile aus, **wartet** es, bis eine Eingabe durch das Betätigen der **Eingabetaste** in der Konsole an das Programm gesendet wird. Je nach Spracheinstellung muss bei **`double`** bzw. **`float`** entweder ein Punkt oder ein Komma als Dezimaltrennzeichen eingegeben werden.

Sie sollten, bevor ein Programm terminiert, mit der Anweisung **input.close();** die Standardeingabe wieder freigeben.