technische universität
dortmund

Fakultät für Informatik
Lehrstuhl 14
Lars Hildebrand

Praktikum zu

Einführung in die Informatik für LogWings, WiMas und MedPhys

Wintersemester 2020/21

Übungsblatt 11

Besprechung:
8.–12.02.2021 (KW 5)

Vorbereitende Aufgaben

Aufgabe 11.1: Bäume

Zunächst wollen wir uns mit Binärbäumen und binären Suchbäumen beschäftigen.

a) Welche Eigenschaft besitzt ein binärer Suchbaum?

Ausgehend eines Knoten sind alle Elemente im linken Unterbaum eines Knoten kleiner und alle Elemente im rechten Unterbaum größer als der Knoten selbst

b) Welchen Vorteil besitzt ein binärer Suchbaum im Vergleich zu anderen dynamischen Datenstrukturen?

Suchen in $\mathcal{O}(\log_2(n))$ – wenn balanciert – Schritten bei n Elementen

c) Wie ist die Höhe eines Binärbaums definiert?

$$h(B) = \begin{cases} 0, & \textbf{wenn } B \textbf{ leer} \\ 1 + \max\{h(B_1), h(B_2)\}, & \textbf{sonst} \end{cases}$$

d) Welcher Traversierung für Binärbäume entspricht folgender Algorithmus:

1. aktueller Knoten
2. linker Teilbaum
3. rechter Teilbaum

Preorder

e) Welcher Traversierung für Binärbäume entspricht folgender Algorithmus:

1. linker Teilbaum
2. rechter Teilbaum
3. aktueller Knoten

Postorder

f) Welcher Traversierung für Binärbäume entspricht folgender Algorithmus:

1. linker Teilbaum
2. aktueller Knoten
3. rechter Teilbaum

Inorder

g) Welche Traversierung kennen Sie noch für für Binärbäume?

Breitendurchlauf

Präsenzaufgaben

Aufgabe 11.2: Vererbung: Einstieg

Gegeben sind folgende Klassen:

```java
public class Person {
    private String firstname;
    private String surname;

    public Person(String firstname, String surname) {
        this.firstname = firstname;
        this.surname = surname;
    }

    public String toString() {
        return this.firstname + " " + this.surname;
    }
}
```

```java
public class Student extends Person {
    private int matrnr;

    public Student(String firstname, String surname, int matrnr) {
        super(firstname, surname);
        this.matrnr = matrnr;
    }

    public String toString() {
        String name = super.toString();
        return this.matrnr + " " + name;
    }
}
```

```java
public class Employee extends Person {
    private String chair;
    private double salary;

    public Employee(String firstname, String surname, String chair,
            double salary) {
        super(firstname, surname);
        this.chair = chair;
        this.salary = salary;
    }

    public String toString() {
        String name = super.toString();
        return "Name: " + name + ", Chair: " + this.chair
            + ", Salary: " + this.salary + " Euro per hour";
    }
}
```

Welche Ausgabe hat folgendes Programm? Testen Sie das Programm **nicht**, indem Sie es abtippen!

```java
public class UniTest {
    public static void main(String[] args) {
        Person visitor = new Person("Max", "Mustermann");
        System.out.println(visitor.toString());

        Student junior = new Student("Karl", "Karlson", 123456);
        System.out.println(junior.toString());

        Employee scientist = new Employee("Markus", "Mueller",
                "Software Engineering", 11.0);
        System.out.println(scientist.toString());

        Person senior = new Student("Mark", "Mustermann", 1248);
        System.out.println(senior.toString());

        Person admin = new Employee("Egon", "Schneider", "Databases", 13.5);
        System.out.println(admin.toString());
    }
}
```

Lösung

```
Max Mustermann
123456 Karl Karlson
Name: Markus Mueller, Chair: Software Engineering, Salary: 11.0 Euro per hour
1248 Mark Mustermann
Name: Egon Schneider, Chair: Databases, Salary: 13.5 Euro per hour
```

Aufgabe 11.3: Vererbung: Quizfragen

In dieser Aufgabe sollen Sie sich mit dem Konzept der Vererbung beschäftigen.

a) Mit welchem Schlüsselwort kann man auf das aktuelle Objekt zugreifen? Z. B. um auf dessen Attribute zuzugreifen, wenn sie von einer lokalen Variable überlagert werden.

Mit dem Schlüsselwort this

Genauer: this.attributname oder this.methodenname()

b) Welches Schlüsselwort wird verwendet um in der Klassendeklaration das Erben von einer anderen Klasse zu kennzeichnen?

Mit dem Schlüsselwort extends

Genauer: public class Unterklasse extends Oberklasse ...

c) Welche Methoden und Attribute sind innerhalb einer Unterklasse von der Oberklasse sichtbar?

Elemente ohne Sichtbarkeitsmodifikator oder mit dem Modifikator public oder protected

d) Mit welchem Schlüsselwort können Sie (unter Umständen überschriebene) Methoden der Oberklasse aufrufen?

Mit dem Schlüsselwort super

für Methoden: super.methode(); oder für Konstruktor: super()

e) Eine Klasse **BachelorStudent** erbt von der Klasse **Student**. Ist die Zuweisung `Student max = new BachelorStudent("Max", "Mustermann");` gültig?

(Unter der Annahme, dass der Konstruktor korrekt aufgerufen wird)

Ja! Ein BachelorStudent kann einer Variablen vom Typ Student zugewiesen werden.

f) Ist entsprechend eine Zuweisung `BachelorStudent maria = new Student("Maria", "Musterfrau");` gültig?

(Unter der Annahme, dass der Konstruktor korrekt aufgerufen wird)

Nein! Ein Student muss kein BachelorStudent sein.

Aufgabe 11.4: Vererbung: Erste Anwendung

In dieser Aufgabe wollen wir eine Unterklasse schreiben und verwenden. Sie benötigen hierzu die Klassen **Vehicle** und **Car** von Blatt 9 oder 10. Diese besitzen eine offensichtliche „*ist ein*"-Eigenschaft zueinander. Kopieren Sie die Quellcode-Dateien der beiden Klassen und ändern Sie die Klasse **Car** so ab, dass sie von **Vehicle** erbt. Der Konstruktor von **Car** ist nun unvollständig. Erweitern Sie den Konstruktor mithilfe des Schlüsselwortes **super** so, dass neu erzeugte Autos Vehikel mit **vier** Reifen sind, die **Benzin** als Treibstoff verwenden.

Aufgabe 11.5: Vererbung: Methoden überschreiben

In dieser Aufgabe wollen wir Methoden überschreiben, um so ihre Funktionalität zu erweitern.

a) Überschreiben Sie die **toString**-Methode der **Vehicle**-Klasse, sodass der Text, den Sie bisher in der **print**-Methode ausgeben, nun als Zeichenkette zurückgegeben wird.

b) Überschreiben Sie die **toString**-Methode der **Car**-Klasse, sodass der Text der **toString**-Methode der Oberklasse, **Vehicle**, um eine neue Zeile (**\n**) und dem Text, den Sie bisher in der **print**-Methode gegeben haben, zurückgegeben wird.

c) Schreiben Sie eine Testklasse mit einer **main**-Methode, in der Sie verschiedene Objekte vom Typ **Vehicle** und **Car** erstellen und direkt an die Funktion **System.out.println()** übergeben.

Komplettlösung für Aufgaben 11.4 und 11.5

```
public class Vehicle {
    private int wheels;
    private String fuel;

    public Vehicle(int inWheels, String inFuel) {
        wheels = inWheels;
        fuel = inFuel;
    }

    public int getWheels() {
        return wheels;
    }

    public String toString() {
        return "Fuel: " + fuel + "\nWheels: " + wheels;
    }

    public void print() {
        System.out.println("Fuel: " + fuel);
        System.out.println("Wheels: " + wheels);
    }
}
```

```
public class Car extends Vehicle{

    private String manufacturer;
    private String model;
    private int horsePower;

    public static final double WATT_PER_HORSEPOWER = 735.5;

    public Car(String inManufacturer, String inModel, int inHorsePower) {
        super(4, "Benzin");
        manufacturer = inManufacturer;
        model = inModel;
        horsePower = inHorsePower;
    }

    public double getPower() {
        return horsePower * WATT_PER_HORSEPOWER;
    }

    public String toString() {
        String result = super.toString();
        result += "\n" + model + " von " + manufacturer + " mit "
```

```
            + (getPower()/1000.0) + " Kilowatt Leistung";
        return result;
    }

    public void print() {
        System.out.println(model + " von " + manufacturer + " mit "
            + (getPower()/1000.0) + " Kilowatt Leistung");
    }

}
```

Aufgabe 11.6: Vererbung: abstrakte Klassen

In dieser Aufgabe wollen wir uns mit dem Konzept abstrakter Klassen beschäftigen:

a) Wenn Sie die Klasse **Vehicle** als Abstrakt deklarieren wollen würden, wie müsste dann die Deklaration der Klasse aussehen?

public abstract class Vehicle { ... }

b) Vehikel sollen nun grundsätzlich eine **getPower()**-Methode, wie die **Car**-Klasse, haben. Die Unterklassen sollen gezwungen werden eine Implementierung anzugeben. Wie sieht die Deklaration dieser Methode aus?

public abstract double getPower();

c) Können Sie nach diesen Änderungen noch Objekte vom Typ **Vehicle** instanziieren?

Nein, aber Objekte vom Typ einer Unterklasse können einer Variablen vom Typ Vehicle zugewiesen werden.

d) Worin läge der Vorteil dieser Änderungen?

Abstrakte Konzepte wie „ein Vehikel“ sollten nur durch tatsächlich existierende Objekte wie Autos repräsentiert werden.

Ergänzende Aufgaben

Aufgabe 11.7: Umsetzung

Setzen Sie die Änderungen der Aufgabe 11.6 in die Praxis um. Ändern Sie entsprechend auch Ihre Testfälle in der Testklasse.

Lösung

```
public abstract class Vehicle { /*ergänzt*/
    private int wheels;
    private String fuel;

    public Vehicle(int inWheels, String inFuel) {
        wheels = inWheels;
        fuel = inFuel;
    }

    public int getWheels() {
        return wheels;
    }

    public abstract double getPower(); /*ergänzt*/

    public String toString() {
        return "Fuel: " + fuel + "\nWheels: " + wheels;
    }

    public void print() {
        System.out.println("Fuel: " + fuel);
        System.out.println("Wheels: " + wheels);
    }
}
```