Universität Rostock,
Fakultät für Informatik und
Elektrotechnik

Algorithmen und Datenstrukturen
Sommersemester 2023
Sonntag, 16.04.2023

Aufgabenblatt 1

Erreichbare Punktzahl: 28 P.

Peter Lustig
Mat.-Nr.: 123456789

Fritz Fuchs
Mat.-Nr.: 987654321

Hermann Paschulke
Mat.-Nr.: 123459876

Aufgabe 1 – Objekt-orientierte Programmierung (OOP) **(18 P.)**

Beantworten Sie die folgenden Fragen zu grundlegenden Konzepten der objekt-orientierten Programmierung in Java.

(a) Was ist ein *Objekt*? Was ist eine *Klasse*? Worin besteht der Unterschied? (2 P.)

Ein Objekt ist häufigerweise eine abstrahierte Repräsentation realer Dinge. Klassen sind eine Kategorie gleichartiger Objekte. Objekte sind Instanzen von Klassen, während Klassen die Baupläne für Objekte darstellen.

(b) Wie wird der Zustand eines Objektes gespeichert? Wie wird sein Verhalten repräsentiert? (2 P.)

Der Zustand eines Objektes wird durch die Werte seine Attribute gespeichert. Das Verhalten wird durch die Methoden der Klasse beschrieben.

(c) Wie werden Objekte in Java erzeugt? Was ist ein *Konstruktor*? Wie und wann werden Objekte in Java wieder zerstört? (3 P.)

Objekte werden mit dem `new`-Keyword generiert. Der Konstruktor initialisiert das durch `new` generierte Objekt. Durch den Garbage-Collector werden Objekte dann erzeugt, wenn sie nicht mehr verwendet werden, d.h. z.B. wenn keine weiteren Referenzen mehr auf das Objekt existieren.

(d) Was ist *Vererbung*? Wie deklariert man in Java eine Klasse, die von einer anderen Klasse erbt? (2 P.)

Vererbung bezeichnet eine hierarchische Relation zwischen Klassen, in der eine Klasse B, die *Unterklasse*, von einer Klasse A, die *Oberklasse*, Attribute, Methoden und Implementierungen erbt. Damit kann B als ein A behandelt werden. In der Klassendefinition kann man mittels dem Keyword `extends` nach dem Namen der Klasse eine andere Klasse angeben, von der sie erbt.

```java
class B extends A {
    ...
}
```

(e) Was sind *abstrakte Klassen* und *Methoden*? (2 P.)

Abstrake Klassen sind Klassen, von denen keine Objekte erzeugt werden können. Abstrakte Methoden sind Methoden ohne Implementierung, die in jeder Unterklasse implementiert werden müssen.

(f) Was sind *Interfaces*? Wozu benötigt man sie in Java? (3 P.)

Interfaces sind ähnlich zu abstrakten Klassen, die ausschließlich aus abstrakten Methoden bestehen. In Java kann eine Klasse nur von maximal einer anderen Klasse erben, jedoch von beliebig vielen Interfaces. Für Mehrfachvererbung sind Interfaces ein Ausweg.

(g) Was sind *Exceptions*? Wie behandelt man sie in Java? (2 P.)

Exceptions sind Fehlerereignisse, die, wenn sie nicht behandelt werden, das Programm stoppen. Sie werden durch `try-catch`-Anweisungen behandelt. Jede `try-catch`-Anweisungen besteht aus einem `try`-Block und beliebig vielen `catch`-Blöcken, die eine bestimmte Exception E behandeln. Im `try`-Block wird der auftretende Fehler abgefangen und der passende `catch`-Fall ausgeführt. Ist kein passender `catch`-Fall vorhanden, wurde sie nicht behandelt und das Programm wird gestoppt.

```java
public static int parseInput(String input) {
    try {
        return Integer.parseInt(input);
    } catch (NumberFormatException e) {
        return -1;
    }
}
```

(h) Was sind *Generics*? Welche Vorteile bieten sie? (2 P.)

Generics machen es möglich, Methoden und Klassen für verschiedene Datentypen gleichzeitig zu definieren. Wenn keine besondere Eigenschafft eines Datentyps T benutzt wird, kann man die Klasse bzw. die Methode A mithilfe von Generics auf eine Klasse bzw. Methode *A<T>* generalisieren. Anstelle des Namens des Datentyps kann dann ein beliebiger, in der Methoden- oder Klassendefinition definierter, Platzhalter verwendet werden

```java
import java.util.ArrayList;

public class GenericsExample {
    public static <T> ArrayList<T> addElement(ArrayList<T> a, T b) {
        a.add(b);
        return a;
    }
```

```
8
9       public static void main(String[] args) {
10          ArrayList<Integer> l1 = new ArrayList<Integer>();
11          l1.add(1);
12
13          ArrayList<String> l2 = new ArrayList<String>();
14          l2.add("Hello");
15
16          addElement(l1, 2);
17          addElement(l2, "World");
18      }
19  }
```

Aufgabe 2 – Komplexe Zahlen **(10 P.)**

Sowohl abstrakte Datentypen (ADT) als auch Objekte kapseln Daten zusammen mit den Operationen, die auf diese zugreifen. Es bestehen viele Gemeinsamkeiten, aber auch wichtige Unterschiede.

(a) Spezifizieren Sie den ADT „Complex Number“ zur Repräsentation komplexer Zahlen. Es soll möglich sein, eine komplexe Zahl zu erzeugen, zwei komplexe Zahlen zu addieren, zu subtrahieren, zu multiplizieren und zu dividieren sowie den Real- und Imaginärteil einer komplexen Zahl zu bestimmen! (3 P.)

Ein algebraischer Datentyp $\mathbb{C}$ für „Complex Number“, wobei $Radd$, $Rsub$, $Rdiv$ und $Rmul$ Operanten zur Addition, Subtraktion, Division und Multiplikation von reellen Zahlen sind:

$$[\mathbb{R}]$$

$$complex : \mathbb{R} \times \mathbb{R} \to \mathbb{C}$$
$$add : \mathbb{C} \times \mathbb{C} \to \mathbb{C}$$
$$sub : \mathbb{C} \times \mathbb{C} \to \mathbb{C}$$
$$mul : \mathbb{C} \times \mathbb{C} \to \mathbb{C}$$
$$div : \mathbb{C} \times \mathbb{C} \to \mathbb{C}$$

$$\forall r_1, r_2, i_1, i_2 \in \mathbb{R} \bullet$$

$$add(complex(r_1, i_1), complex(r_2, i_2)) = complex(r_1 + r_2, i_1 + i_2)$$
$$sub(complex(r_1, i_1), complex(r_2, i_2)) = complex(r_1 - r_2, i_1 - i_2)$$
$$mul(complex(r_1, i_1), complex(r_2, i_2)) = complex(r_1 r_2 - i_1 i_2, r_1 i_2 + r_2 i_1)$$
$$div(complex(r_1, i_1), complex(r_2, i_2)) = complex\left(\frac{r_1 r_2 + i_1 i_2}{r_2 r_2 + i_2 i_2}, \frac{i_1 r_2 - r_1 i_2}{r_2 r_2 + i_2 i_2}\right)$$

(b) Wie lassen sich aus der Signatur des ADT Complex die Methodendeklarationen einer objekt-orientierten Implementierung ableiten? Wie gehen Sie vor? (3 P.)

Der ADT ist die Spezifikation einer Klasse. Verwendete Sorten erscheinen als benutzte Datentypen, $\mathbb{R}$ steht hier für `double`, $\mathbb{C}$ ist die Klasse `Complex` selbst. Die Operatoren sind die Signaturen der Methoden für die Klasse, wobei der Definitionsbereich der Abbildung durch die Argumente dargestellt und der Rückgabewert der Methode der Wertebereich ist. Die Regeln sind die Implementierung der jeweiligen Methoden.

(c) Erstellen Sie eine objekt-orientierte Implementierung in Java! Nutzen Sie hierzu die Datei Complex.java! (4 P.)

Siehe auch `src/Complex.java`.

```java
class Complex {
    double r;
    double i;

    public Complex(double r, double i) {
        this.r = r;
        this.i = i;
    }

    public Complex add(Complex c1, Complex c2) {
        return new Complex(c1.r + c2.r, c1.i + c2.i);
    }

    public Complex sub(Complex c1, Complex c2) {
        return new Complex(c1.r - c2.r, c1.i - c2.i);
    }

    public Complex mul(Complex c1, Complex c2) {
        return new Complex(
            c1.r*c2.r - c2.i*c2.i,
            c1.r*c2.i + c2.r*c1.i);
    }

    public Complex div(Complex c1, Complex c2) {
        double n = c2.r*c2.r + c2.i*c2.i;

        return new Complex(
            (c1.r*c2.r + c1.i*c2.i)/n,
            (c1.i*c2.r + c1.r*c2.i)/n);
    }
}
```