University of Zurich[UZH]

s.e.a.l. software evolution & architecture lab

Programmierung für Mathematik HS11

Übung 8

1 Aufgabe: Codeverständnis (Repetition)

1.1 Lernziele

1. Code verstehen können.
2. Fehler im Code finden und korrigieren können.

1.2 Aufgabenstellung

a) Grundlegendes Code Verständnis

Überlegen Sie sich, was der Output bei folgenden Code Ausschnitten wäre, wenn sie so in einer `main`-Methode stehen würde. (Sie können dazu auch die Java-API verwenden.)

```
1 String dog = "dog";
2 String cat = "cat";
3 cat.toUpperCase();
4 System.out.println(cat);
5 System.out.println(dog);
```

Listing 1: Ausschnitt 1

```
1 double x = 3.54;
2 int z = (int)x;
3 System.out.println(z/2);
```

Listing 2: Ausschnitt 2

```
1 double x = 1.25;
2 int z = 2;
3 System.out.println(z*x);
```

Listing 3: Ausschnitt 3

```
1 String cat = "CAT";
2 cat = cat.toLowerCase();
3 if (cat == "cat"){
4    System.out.println("is equal");
5 } else {
6    System.out.println("isn't equal");
7 }
```

Listing 4: Ausschnitt 4

```
1 for (int i=5; i>2; i--){
2    System.out.println(i+1);
3 }
```

Listing 5: Ausschnitt 5

b) Vererbung

Kompiliert folgender Code und wenn ja, welche Ausgabe wird produziert?

```
1 public class Animal {
2    public void print(){
3       System.out.println("it's an animal.");
4    }
5    public static void main(String[] args){
6       Animal a = new Dog();
7       a.print();
8    }
9 }
```

Listing 6: Die Klasse `Animal`

```
1 public class Dog extends Animal {
2    public void print(){
3       System.out.println("it's a dog.");
4    }
5 }
```

Listing 7: Die Klasse `Dog`

c) Rekursion

Welche Ausgabe wird von folgendem Code produziert?

```
1 public class TestDriver {
2    public static void main(String[] args){
3       System.out.println(A(2,1));
4    }
5    public static long A(long m, long n) {
```

```
6         if (m == 0){
7            return n+1;
8         } else if (n == 0) {
9            return A(m-1, 1);
10        } else {
11           return A(m-1, A(m, n-1));
12        }
13    }
14 }
```

Listing 8: Die Klasse `TestDriver`

d) Fehler finden

Korrigieren Sie folgenden Code so, dass er korrekt kompiliert und auch zu Laufzeit keine Fehler auftreten.

```
1  public class TestDriver2 {
2     public static void main(String[] args) {
3        int[] array = new int(3);
4        array[0] = 3;
5        array[2] = 1;
6        print(array)
7     }
8     public void print(int[] array) {
9        for (int i=0; i<=array.length; i++){
10          System.out.println(array[i]);
11       }
12    }
13 }
```

Listing 9: Die Klasse `TestDriver2`

2 Aufgabe: Euklidischer Algorithmus

2.1 Lernziele

1. Einen rekursiven Algorithmus in einen iterativen Algorithmus umschreiben können.
2. Methoden überladen können.
3. Exceptions sinnvoll behandeln können.

2.2 Aufgabenstellung

a) Euklidischer Algorithmus

Die folgende Methode berechnet rekursiv den grössten gemeinsamen Teiler zweier Zahlen `a` und `b` (vgl. Übungsstunde 6):

```
1 public static int euclidAlgorithm(int a, int b) {
2    if (b == 0) {
3       return a;
4    } else {
5       return euclidAlgorithm(b, a%b);
6    }
7 }
```

Listing 10: Die Methode `euclidAlgorithm`

Schreiben Sie die Methode so um, dass der grösste gemeinsame Teiler iterativ berechnet wird. *Hinweis*: Benutzen Sie eine while-Schleife.

b) Brüche in Java

Schreiben Sie eine Klasse `Fraction`, welche einen Bruch darstellen soll. Der Bruch soll mit einem Konstruktor gesetzt werden können und automatisch in gekürzter Form abgespeichert werden. Falls der Benutzer für den Nenner 0 eingibt, soll eine Exception geworfen werden. Benutzen Sie für das Kürzen der Brüche die Methode `euclidAlgorithm` aus Aufgabe a).

c) Rechnen mit Brüchen

Die Klasse soll Methoden `add` und `multiply` zur Verfügung stellen, womit entweder ein anderer Bruch oder eine ganze Zahl addiert bzw. multipliziert werden können. Das Resultat soll direkt (in gekürzter Form) in den Instanzvariablen der Klasse abgespeichert werden. Prüfen Sie Ihre Klasse in einem TestDriver und behandeln Sie allfällige Exceptions in einem `try-catch`-Block. Ergänzen Sie `Fraction` ausserdem um eine geeignete `toString` Methode.

3 Aufgabe: Implementierung eines News-Dienstes

3.1 Lernziele

1. Problemstellung mit Hilfe von Interfaces sinnvoll implementieren können.
2. ArrayList verwenden können.

3.2 Aufgabenstellung

Bilden Sie folgenden Sachverhalt auf ein Interface, sowie auf Klassen, Attribute und Methoden ab

a) News-Dienst

Abonnenten (Subscriber), können sich bei verschiedenen News-Diensten (Newsletter) anmelden und erhalten dann stets Neuigkeiten zum Thema ihrer Wahl, sobald diese verfügbar werden (Abonnenten sollen erhaltene News einfach auf die Kommandozeile ausgeben). Jeder Dienst bietet News zu einem Thema (Topic) an. News-Dienste können nun wie folgt konfiguriert werden:

- Dienste können sich auf ein sehr spezifisches Thema festlegen, also z.B. ein Dienst auf News zum Thema *Neuerscheinungen DVDs*, ein anderer zum Thema *Neuerscheinungen Bücher*.
- Dienste können sich aber auch bei mehreren anderen Diensten anmelden (d.h. sie können selber auch als Abonnenten auftreten) und senden in dem Fall ihren eigenen Abonnenten die gesammelten News zu einem breiteren Themenkreis. Ein Beispiel wäre ein Dienst mit News zum Thema *Neuerscheinungen Medien*, der sich sowohl bei dem Dienst mit dem Thema *Neuerscheinungen DVDs*, als auch jenem mit dem Thema *Neuerscheinungen Bücher* anmeldet und sämtliche Neuigkeiten zu den beiden Themen an seine Abonnenten weiterleitet.

b) TestDriver

Schreiben Sie einen TestDriver um die Funktionnalität Ihres Programmes zu testen.

4 Aufgabe: Richtig oder falsch

4.1 Lernziele

1. Das Wissen über Java repetieren.

4.2 Aufgabenstellung

Entscheiden Sie für folgende Aussagen, ob sie jeweils wahr oder falsch sind.

1. `1stDigit` ist ein gültiger Bezeichner für eine Variable.
2. `myVariable` und `myvariable` bezeichnen dieselbe Variable.
3. Der Operator `*` bindet in einem Ausdruck stärker als `++`.
4. Eine Division von zwei Integer Zahlen ergibt immer eine Integer Zahl.
5. Das Vergleichen von zwei Strings mittels == verursacht einen Fehler beim Kompilieren.
6. Jede Klasse benötigt mindestens eine main-Methode.
7. Jede `for`-Schleife kann in eine `while`-Schleife umgewandelt werden.
8. Jedes `if`-Statement kann verschiedene `else`-Statements besitzen.
9. Jede `do-while`-Schleife kann in eine `while`-Schleife umgewandelt werden.
10. Eine `public` Methode kann auf `private` Instanzvariablen derselben Klasse zugreifen.
11. Eine statische Methode kann auf `private` Instanzvariablen derselben Klasse zugreifen.
12. Eine statische `private` Methode kann auf `private` Instanzvariablen derselben Klasse zugreifen.
13. Jede `void` Methode muss ein `return` Statement besitzen.
14. Statische Methoden können auf nicht-statische Methoden derselben Klasse zugreifen.
15. Nicht-statische Methoden können auf statische Methoden derselben Klasse zugreifen.
16. Statische Methoden benötigen immer ein `return` Statement.
17. Getter Methoden besitzen ein `return` Statement.
18. Setter Methoden dürfen keine Parameter besitzen.
19. Objekte dürfen nur mit == verglichen werden.
20. In jeder Klasse muss mindestens ein Konstruktor definiert werden.
21. Ein Konstruktor muss mit dem Rückgabewert `void` deklariert werden.
22. Das Schlüsselwort `this` verweist immer auf die eigene Klasse.
23. Eine `private` Instanzvariable vom Typ `int` kann nur durch Methoden der eigenen Klasse verändert werden.
24. Eine `private` Instanzvariable vom Typ `int[]` kann nur durch Methoden der eigenen Klasse verändert werden.
25. Arrays dürfen nur mit == verglichen werden.
26. `myArray[0].length` ist niemals ein gültiger Aufruf.
27. Jede Rekursion kann in eine Iteration umgewandelt werden aber nicht umgekehrt.