Klausurvorbereitung Blatt 1 **17./18.01.2024**

1 Begriffe

1.) a) Das Programm, mit dem der Java-Quellcode übersetzt wird, nennt man:

b) Das Programm, mit dem die Java-Klassen ausgeführt werden, nennt man:

c) Eine Ausnahme, die ein Entwickler im Code in jedem Fall behandeln muss, nennt man:

d) Die Definition einer Methode, die in einer Oberklasse bereits definiert ist, nennt man in Java:

e) Die Definition mehrerer Methoden mit gleichem Namen und unterschiedlicher Signatur nennt man:

f) Eine Methode, bei der ausschließlich die Signatur definiert und keine Implementierung angegeben ist, nennt man:

g) Eine Methode, die ohne eine Objektinstanz aufgerufen werden kann, nennt man:

h) Welche Art von Klasse wird im folgenden Code-Ausschnitt erzeugt?

```
Funktion f = new Funktion() {
  public double getY(double x) {
    return x*x;
  }
};
```

i) Zu welcher Art von Klassen gehören die Klassen `Integer`, `Float` und `Character`?

j) Sie wollen einen String aus vielen einzelnen Strings zusammenbauen. Welche Klasse bietet sich aus Geschwindigkeits-Gründen an?

k) Die Anweisung, um eine Schleife aus dem Schleifeninneren heraus abzubrechen, heisst:

l) Die Variable `Bruch b;` kann nicht nur Objekte der Klasse `Bruch` aufnehmen. Welche Objekte kann `b` noch aufnehmen?

m) Welchen Konstruktor fügt Java einer Klasse automatisch zu, wenn kein Konstruktor implementiert wurde?

n) Wie nennt man den Teil in spitzen Klammern im folgenden Ausdruck?

```
ArrayList<String> x = new ArrayList<String>();
```

o) Schreiben Sie den Java-Code auf, den Sie benötigen, um eine Datei *daten.txt* zu öffnen und die erste Zeile der Datei in den String `kopf` einzulesen. Berücksichtigen Sie auch Exceptions.

p) Erläutern Sie anhand der folgenden Programmzeilen kurz die Begriffe

(1) Interface

(2) Anonyme innere Klasse

```
1 Function f = new Function() {
2   public double getY(double x) {
3     return x*x;
4   }
5 };
```

2 Code analysieren

2.) Sie finden in einem Programm die folgenden unkommentierten Funktionen vor:

```
1  public static ArrayList<Integer> getValues(int num) {
2      return getValues(num, 2);
3  }
4
5  private static ArrayList<Integer> getValues(int num, int fac) {
6      if (num == 1) {
7          return new ArrayList<Integer>();
8      } else if (num % fac == 0) {
9          ArrayList<Integer> ret = getValues(num / fac, fac);
10         ret.add(fac);
11         return ret;
12     } else {
13         return getValues(num, fac + 1);
14     }
15 }
```

a) In welchen Zeilen befinden sich rekursive Funktionsaufrufe?

b) In welchen Zeilen findet sich die Abbruchbedingung?

c) Ein Aufruf von `getValues(4)` führt zu folgenden Aufrufen und Rückgabewerten:

```
getValues(4)
getValues(4,2)
getValues(2,2)
getValues(1,2)
return {}
return {2}
return {2,2}
return {2,2}
```

Welche Aufrufe und Rückgabewerte ergeben sich für `getValues(60)`?

d) Was berechnet die Funktion `getValues(int num)`?

e) Der Aufruf von z.B. `getValues(53)` führt zu relativ vielen verschachtelten Rekursionsaufrufen. Wie kann man die Rekursionsaufrufe deutlich früher abbrechen? Beschreiben Sie, an welchen Stellen der Code geändert werden müsste.

3 Funktionen

3.) Ein Text ist zeilenweise in einem Stringfeld abgespeichert. Schreiben Sie eine Methode

```
public static int sucheAB(String[] text)
```

die zählt, wie oft die Zeichenfolge „ab“ im Text vorkommt. Dabei wird auch berücksichtigt, wenn eine Zeile mit dem Zeichen „a“ aufhört und die nächste mit dem Zeichen „b“ anfängt.

4.) Schreiben Sie eine Funktion

```
public static int ggt(int z1, int z2)
```

die den größten gemeinsamen Teiler von z_1 und z_2 zurückgibt. Gehen Sie davon aus, dass z_1 und z_2 beide > 0 sind.

4 Objektorientiertes Programmieren

5.) **Matrix.** Schreiben Sie eine Klasse `QuadratMatrix`, die eine quadratische Matrix aus **`double`**-Werten enthält. Benutzen Sie folgende Vorlage:

```
public class QuadratMatrix {
  private double[][] data;

  public void setData(int x, int y, double val) {
    data[x][y] = val;
  }
  public double getData(int x, int y) {
    return data[x][y];
  }
}
```

a) Fügen Sie eine Methode hinzu, die die Größe der Matrix zurückgibt.

b) Schreiben Sie zwei Konstruktoren

```
// Erzeugt eine Diagonalmatrix
// der Groesse n mal n mit x auf der
// Hauptdiagonalen (und 0 sonst).
public QuadratMatrix(int n, double x)

// Erzeugt eine Quadratmatrix aus dem double-Feld feld
// (Feld wird kopiert).
public QuadratMatrix(double[][] feld)
```

Achten Sie darauf, dass die Konstruktoren bei fehlerhaften Übergabewerten Exceptions werfen.

c) Schreiben Sie eine weitere Methode

```
public boolean istSymmetrisch()
```

die überprüft, ob die Matrix symmetrisch ist oder nicht. Sie gibt **`true`** bei symmetrischen Matrizen zurück und **`false`** bei unsymmetrischen.

5 Interfaces

6.) Schreiben Sie eine Klasse `GramSchmidt` mit einer statischen Methode

```
public static void gramSchmidt(Vektor v1, Vektor v2)
```

Die Methode soll die beiden übergebenen Vektoren nach Gram-Schmidt orthonormieren. Sie hat keinen Rückgabewert, sondern die übergebenen Objekte (die ja als Referenz übergeben werden) werden in der Methode geändert. Beachten Sie, dass die Methode für beliebige Vektoren eines Vektorraums über $\mathbf{R}$ funktionieren soll, also nicht nur für Vektoren des $\mathbf{R}^n$, sondern auch für Vektoren anderer (euklidischer) Vektorräume. Komplexe Vektorräume werden nicht berücksichtigt.

Wählen Sie einen objektorientierten Ansatz. Schreiben Sie dazu eine weitere Klasse `Vektor` als ein Interface oder eine abstrakte Klasse. Eine Implementation für einen konkreten Vektorraum müssen sie **nicht** schreiben.

Tipp: Die Gleichungen für die Orthonormierung nach Gram-Schmidt sind:

$$\begin{aligned} w_1 &= \frac{v_1}{\|v_1\|} \\ r_2 &= v_2 - \langle v_2, w_1 \rangle w_1 \\ w_2 &= \frac{r_2}{\|r_2\|} \end{aligned}$$