

Aufgabe 7.1 (Kommentare) – In der letzten Vorlesung haben wir uns mit Kommentaren beschäftigt. Schauen Sie sich Ihre Lösung für folgende Aufgaben aus den letzten Übungsblättern an:

- Fibonacci iterativ
- Array-Rotation um n
- Rekursiver String.

Überarbeiten Sie Ihren Code so, dass Sie mühelos den gewählten Algorithmus erläutern können. Kommentieren Sie sinnvoll, wählen Sie sinnvolle Bezeichner, machen Sie eine Skizze. Tauschen Sie Ihre Lösungen. Versuchen Sie den jeweils „fremden“ Code zu verstehen.

Aufgabe 7.2 (Konstruktoren) – Schreiben Sie eine Klasse Studierende mit den Instanzvariablen Name, Matrikelnummer, Geburtsjahr und der Klassenvariablen counter, die die Anzahl der Studierenden zählt.

- Schreiben Sie einen Konstruktor, der das Geburtsjahr initialisiert.
- Schreiben Sie einen Konstruktor, der alle Werte initialisiert.

Aufgabe 7.3 (Klassen) – Gegeben ist die nachfolgende Klasse.

```
public class SchachSpieler {  
  
    private String name;  
    private int alter;  
  
    public String getName() {  
        return name;  
    }  
    public void setName(String name) {  
        this.name = name;  
    }  
    public int getAlter() {  
        return alter;  
    }  
    public void setAlter(int alter) {  
        this.alter = alter;  
    }  
}
```

- Erläutern Sie den Aufbau der Klasse grafisch. Erstellen Sie hierfür ein UML-Diagramm.
- Definieren Sie eine Klasse SchachSpielerTest.

Was passiert durch die Anweisungen? Warum benötigt man die zweite Anweisung?

```
01 SchachSpieler mueller;  
02 Mueller = new SchachSpieler();
```

- Was versteht man unter einem Konstruktor?
- Erstellen Sie einen Konstruktor in dem alle Instanzvariablen als Argumente übergeben und initialisiert werden. Was passiert mit der bereits implementierten Anweisung aus Ihrer Testklasse?

```
02 Mueller = new SchachSpieler();
```

- e) Was ist der Unterschied zwischen einer Instanzvariablen und einer Klassenvariablen?
- f) Erweitern Sie Ihre Klasse SchachSpieler um eine Instanzvariable „startNummer“, die jedem Schachspieler eine (eindeutige) Startnummer im Turnier vergibt.

Aufgabe 7.4 (Klassen) – Schreiben Sie eine Klasse Mitarbeiter, die folgende Instanzvariablen enthält:

- Name
- Vorname
- Alter

Weiterhin soll die Anzahl der Mitarbeiter*innen erfasst werden können.

- a) Schreiben Sie einen Konstruktor, der als Parameter das Alter sowie den Vor- und Nachnamen bekommt.
- b) Schreiben Sie - wo es Sinn macht - setter und getter.
- c) Schreiben Sie eine Methode mit folgender Spezifikation
`public boolean istAelter (Mitarbeiter aktuellerMitarbeiter)`
Wenn das Alter des Objekts größer ist als das Alter von „aktuellerMitarbeiter“ ist der Wert true, sonst false.
- d) Schreiben Sie eine Methode toString, die eine Zeichenkette zurückgibt, die den Namen, Vornamen, das Alter sowie die Personalnummer enthält.
- e) Schreiben Sie eine Methode, die die Anzahl der Mitarbeiter zurückgibt.
- f) Schreiben Sie eine Testklasse, die Mitarbeiter-Objekte erzeugt und die einzelnen Methoden testet.

PROGRAMMIER- TECHNIK I ≧ ÜBUNG ≦

7

Aufgabe 7.5 (Heap) – Gegeben ist das nachfolgende Programm. Zeichnen Sie die Referenzen in der Zeile 16 ein. Ziehen Sie hierfür Linien zwischen den Referenzvariablen `hc[0] ... hc[4]` und den entsprechenden Objekten.

```
01 public class HeapChaos {  
02     int id = 0;  
03  
04     public static void main(String[] args) {  
05         HeapChaos [] hc = new HeapChaos[5];  
06         for (int i = 0; i<3; i++) {  
07             hc[i] = new HeapChaos();  
08             hc[i].id = i;  
09         }  
10         hc[3] = hc[1];  
11         hc[4] = hc[1];  
12         hc[3] = null;  
13         hc[4] = hc[0];  
14         hc[0] = hc[3];  
15         hc[3] = hc[2];  
16  
17     }  
18 }
```



Aufgabe 7.6 (Klassen) – Schreiben Sie eine Klasse Cocktail, die folgende Instanzvariablen enthält:

- Name // Name des Cocktails
 - Zutaten // String-Array mit Zutaten
 - IstGeheimrezept // gibt an ob das Rezept „gedruckt“ werden kann oder nicht
1. Schreiben Sie einen Konstruktor, der als Parameter den Namen des Cocktails bekommt.
 2. Schreiben Sie einen Konstruktor, der als Parameter den Namen sowie die Zutatenliste des Cocktails bekommt.
 3. Schreiben Sie getter und setter für die Instanzvariablen
 4. Schreiben Sie eine Methode, die abfragt ob es ein Geheimrezept ist
 5. Schreiben Sie eine Methode, die den Namen und die Zutatenliste ausgibt (wenn es kein Geheimrezept ist, ansonsten pssst...)
 6. Angenommen, Sie experimentieren mit den Cocktail-Rezepten und würden gerne ab und zu die Zutatenliste ergänzen. Welches Problem ergibt sich nach aktuellem Design?

Aufgabe 7.7 (String-Array-Liste) – Schreiben Sie eine Klasse StringArrayList, die ein String-Array (also ein Array aus Strings) bei Bedarf dynamisch vergrößert, **ohne Java-Bibliotheken zu verwenden**.

Folgende Spezifikation ist angegeben:

```
// Erzeugt ein String-Array mit der Standardkapazität 10
public StringArrayList ()

// Erzeugt ein String-Array mit der angegebenen „Kapazität“
public StringArrayList (int capacity)

// Gibt die Anzahl der Elemente im Array zurück
public int getSize ()

// Gibt true zurück wenn das Array leer ist, false sonst
public boolean isEmpty ()

// Gibt true zurück wenn der übergebene String s im String-Array enthalten ist, false sonst
public boolean contains (String s)

// Fügt den übergebenen String zum String-Array hinzu
// Ist das Array „voll“, wird die Größe angepasst
public void add(String aktuellerString)

// Gibt den String an der Position index zurück
public String get(int index)

// Löscht den String an der Position index
public void remove(int index)
```

Tipps (nur bei Bedarf lesen ;-): Gehen Sie dabei folgendermaßen vor, testen Sie dabei jeden Schritt in einer `StringArrayTest`-Klasse

1. Definieren Sie eine Klassen-Konstante, die die default-Größe Ihres String-Arrays speichert. Laut Spezifikation ist das 10.
2. Definieren Sie eine private Instanzvariable vom Typ String-Array. Das ist der „gekapselte“ Speicher für unsere Strings.
3. Definieren Sie eine private Instanzvariable, die die nächste freie Position im Array speichert
4. // Konstruktoren – Sollten klar sein
5. `getSize`: → Gibt die nächste freie Position zurück
6. `isEmpty`: → Gibt `true` zurück, wenn die nächste freie Position 0 ist
7. `contains`: → Durchläuft das Array und gibt `true` zurück, wenn an einer Position im Array der String gespeichert ist
8. `add`: → Fügt den String an die nächste freie Position. Falls das Array keinen Platz mehr hat, wir ein neues Array mit (z.B. doppelter Größe als das vorherige) erzeugt und die Inhalte des alten Arrays hineinkopiert. Die freie Position wird aktualisiert.
9. `get` → sollte klar sein
10. `remove` → kopiert ab dem gelöschten String alle Strings im Array um eine Position nach links und aktualisiert den Zähler für die freie Position