Prof. Dr. Christoph Bockisch
MSc. Steffen Dick
Fachbereich Mathematik und Informatik
AG Programmiersprachen und -werkzeuge

Philipps Universität Marburg

Übungen zur Vorlesung
Objektorientierte Programmierung: Wintersemester 2022/2023

Nr. 2, Abgabe bis 7.11.2022

Hinweis: Ab dieser Übung soll nur eine Abgabe pro Gruppe eingereicht werden!

Hinweis zur Abgabe von JShell-Aufgaben:

Mit dem Befehl */ed* können Sie einen einfachen Texteditor über die JSHell öffnen. Mit dem Befehl */save <Dateiname>* können Sie den Zustand Ihrer JShell-Session als Textdatei speichern. Geben Sie für jede Aufgabe eine entsprechende Textdatei ab. Benennen Sie die Dateien nach dem folgenden Muster: *aufgabe-<Aufgabenzettel> <Aufgabennummer>.txt*

Aufgabe 2.1: Simple and Clean — 4 Punkte

Schreiben Sie eine Funktion,

a) `predecessor`, die einen Paramter vom Typ `int` übergeben bekommt und den Vorgänger als Ergebnis liefert. **2**

   Rufen Sie predecessor mit den Werten 1337, 0 und -2147483648 auf.

   Welche Ergebnisse haben Sie bekommen? Welche waren erwartet?

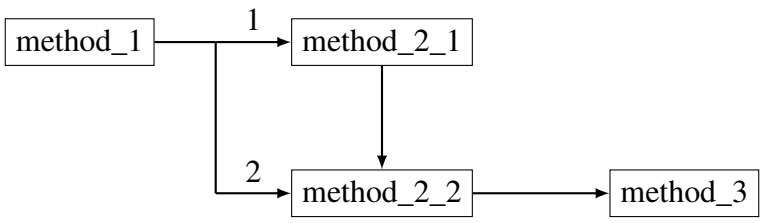
b) `isEqual`, die zwei Parameter vom Typ `String` entgegen nimmt und diese vergleicht. Wenn beide Werte gleich sein sollten, soll die Funktion `true` als Ergebnis liefern, ansonsten `false`. **2**

   Rufen Sie `isEqual` auf und vergleichen Sie die folgenden Werte miteinander: `"Cthulhu"` mit `"Nyarlathotep"`, `"Lovecraft"` mit `"Lovecraft"` und `"Howard"` mit `new String("Howard")`

   Welche Ergebnisse haben Sie bekommen? Welche waren erwartet?

Aufgabe 2.2: One Missed Call | 4 Punkte

Methoden können andere Methoden aufrufen. Bilden Sie die folgende Aufrufhirarchie mit Methoden nach:



Als erste Zeile jeder Methode sollen Sie `Enter <MethodName>` und für jede letzte Zeiler einer Methode `Exit <MethodName>` auf der Konsole ausgeben. Bevor Sie `method_1` aufrufen, schreiben Sie Ihre Erwartungen an eine Reihenfolge auf und vergleichen diese anschließend mit dem echten Ergebnis.

Aufgabe 2.3: Veni! Vidi! Vigor! | 10 Punkte

Letztes Mal haben Sie einen Algorithmus für einen Snack-Automaten geplant. In dieser Aufgabe sollen Sie diesen nun implementieren.

Ein kurzer Rückblick:

Ihre Implementierung sollte einen Wert vom Typen **`int`** entgegennehmen, 72ct abziehen und Ihr Wechselgeld in Münzen auf der Konsole ausgeben. Stellen Sie sicher, dass ihr Wechselgeld immer in den höchstwertigsten Münzen wie möglich ausgegeben wird (also sollte ein Wechselgeld von 1€ als `"1€"` und nicht als `"50ct 50ct"` ausgegeben werden).

Mögliche Münzen:

1ct, 2ct, 5ct, 10ct, 20ct, 50ct, 1€, 2€

Stellen Sie sicher, dass Sie den von Ihnen vorgeschlagenen Weg, um sicherzustellen, dass Ihr Algorithmus funktioniert, ebenfalls implementieren.

Aufgabe 2.4: Landratten! 6 Punkte

Betrachten Sie die folgenden Code-Ausschnitte und geben Sie die Scopes der jeweiligen Variablen an.

Beispiel:

```
1  int x = 6;
2  int y = 7;
3  System.out.println(x * y);
```

Lösung:
x: Zeile 1-3
y: Zeile 2-3

a) Bestimmen Sie die Scopes von `testPassed` und `output` **2**

```
1  int result = pow(2,3);
2  boolean testPassed = result == 6;
3  String output = "";
4  if(testPassed){
5    output = "passed";
6  }
7  else{
8    output = "failed";
9  }
10 System.out.println(output);
```

b) Bestimmen Sie die Scopes von `i` und `output` **2**

```
1  int i = 100;
2  String output = "Testing is important!";
3  while(i > 0){
4    output+= output; //Same as output = output + output;
5    i-=1; //Same as i = i - 1;
6  }
7  System.out.println(output);
```

c) Bestimmen Sie die Scopes von `testPassed` und `output` **2**

```
1  boolean testPassed = pow(2,3) == 6;
2  if(testPassed){
3    String output = "passed";
4    System.out.println(output);
5  }
6  else{
7    String output = "failed";
8    System.out.println(output);
9  }
```