Technische Universität Braunschweig
Institut für Anwendungssicherheit

Prof. Dr. Martin Johns

Programmieren I

02. Übungsblatt

Hinweis: Empfehlung: Nehmen Sie immer an den Rechnerübungen teil.
Auch dann, wenn Sie Ihrem Tutor keine Hausaufgabe vorführen müssen.

Aufgabe 6: Erstellen Sie eine neue Datei und nutzen Sie die Scanner Klasse, um zwei Zahlen in Ihr Programm einzugeben. Nach der Eingabe sollen beide Zahlen addiert werden. Gehen Sie wie folgend vor:

- Fügen Sie in in der ersten Zeile der neuen Datei `import java.util.Scanner;` ein um die Scanner Klasse verfügbar zu machen.
- Erstellen Sie in der Datei nun eine neue Klasse. Der Name der Klasse muss mit dem Dateinamen übereinstimmen. Zum Beispiel: `public class ScannerExample {}`
- Erstellen Sie die *main*-Methode. Prägen Sie sich die Signatur der Methode ein. Sie werden diese immer wieder benötigen.
  ```
  public static void main(String[] args) {}
  ```
- Erstellen Sie einen neuen Scanner mit der folgenden Zeile:
  ```
  Scanner sc = new Scanner(System.in);
  ```
- Erzeugen Sie eine Ausgabe mit der Aufforderung eine neue Zahl einzugeben:
  ```
  System.out.println("Geben Sie eine Zahl ein.");
  ```
- Lesen Sie die nächste eingegebene Zahl und speichern Sie diese in einer int-Variable.
  ```
  int zahl1 = sc.nextInt();
  ```
- Wiederholen Sie die beiden letzten Schritte für die Eingabe einer weiteren Zahl.
- Erzeugen Sie eine Ausgabe mit der Summe der beiden Zahlen: `System.out.println(zahl1 + zahl2);`

Aufgabe 7: Programmieren Sie einen einfachen Taschenrechner. Zuerst sollen wie in Aufgabe 7 zwei Zahlen eingegeben werden. Dann soll nach einer weiteren Zahl zur Bestimmung der Rechenoperation gefragt werden. Wurde eine 1 eingegeben, sollen beide Zahlen addiert, wurde eine 2 eingegeben sollen beide subtrahiert werden. Geben Sie hierfür eine entsprechende Aufforderung an den Nutzer aus.
Verwenden Sie zur Implementierung des Taschenrechners das folgende Konstrukt:

```
if (rechenMethode == 1) {...
} else if (rechenMethode == 2) {...
} else {...}
```

Aufgabe 8: Programmieren Sie den Taschenrechner aus der vorherigen Aufgabe nun mit einer `switch-case` Anweisung anstellte der `if`, `else if` Konstruktion.

Aufgabe 9: Welche Version des Taschenrechners halten Sie für eleganter?

Aufgabe 10: Programmieren Sie ein Programm, welches als Eingabe nun keine Zahl, sondern einen String, also eine Zeichenkette, erhält und jeweils das semantische Gegenteil von einem vordefiniertem Set davon ausgibt. Zum Beispiel:

$$hoch <-> tief$$
$$klein <-> groß$$
$$stark <-> schwach$$

Nutzen Sie dafür die `switch-case`-Anweisung.

Aufgabe 11: Programmieren Sie ein kleines Ratespiel. Hier sollen Sie als erstes eine Zahl zwischen 1 und 100 eingeben. Dann soll der Computer die Zahl erraten.
Nutzen Sie nun zur Eingabe nicht mehr den Scanner, sondern Übergeben Sie die zu erratende Zahl als Programmparameter. Der erste Parameter kann wie folgt ausgelesen werden:
`int zahl = Integer.parseInt(args[0]);` Der Computer soll nun eine Zufallszahl generieren und diese mit der übergebenen Zahl vergleichen. Ist die Zahl korrekt, soll die Anzahl der vom Computer benötigten Versuche ausgegeben werden. Ist die Antwort falsch, soll der Computer es mit der nächsten zufälligen Zahl versuchen. Dies wird solange wiederholt, bis der Computer die richtige Zahl errät.
Eine Zufallszahl zwischen 0.0 (inklusive) und 1.0 (exklusive) kann wie folgt generiert werden: `double zahl = Math.random();`
Überlegen Sie sich, wie Sie das Rateverfahren verbessern können. Testen und evaluieren Sie Ihren Ansatz anhand Ihrer Simulation und der Anzahl der benötigten Versuche.

Aufgabe 12: Werten Sie – jeweils unter Annahme der Deklaration `int x = 7;` – die folgenden Ausdrücke aus. In welchen Fällen wird der Wert von `x` durch die Auswertung verändert? Überlegen Sie zuerst und überprüfen Sie Ihre Antwort anschließend am Rechner.

```
a) x <= 2        e) x & 2         i) x >> 2
b) x -= 2        f) x << 2        j) x += 2
c) x % 2         g) x | 2         k) x =+ 2
d) x / 2         h) x ^ 2         l) x =~ 2
```

Aufgabe 13: Was geben die folgenden Programmstücke aus? Überlegen Sie zuerst und überprüfen Sie Ihre Antwort anschließend am Rechner. (Die Formatierung hält sich in diesem Fall bewusst nicht an die Formatierungsrichtlinien.)

a)

```java
boolean b = false;
if (b = true) {
    System.out.println(true);
} else {
    System.out.println(false);
}
```

listings/Aufgabe22a.java

b)

```java
int i = 0;
if (i < 1)
    i = i + 1;
    if (i < 1)
        i = i + 1;
else
    i = i - 1;
System.out.println(i);
```

listings/Aufgabe22b.java

c)

```java
int i = 0;
if (i > 0)
    if (i < 1) i = i + 1;
else
    i = i - 1;
System.out.println(i);
```

listings/Aufgabe22c.java

Aufgabe 14: Die nachfolgenden Programmfragmente weisen Fehler auf. Finden Sie diese. Begründen Sie Ihre Antwort.

```java
boolean println, true;
char ab c, de, f^g;
int a = 0xf7, b = 0xg7, c = 12e3, d = 017, 2e;
double s_, t$u, v_$w;
System.out.println("a: " , a);
```

listings/Aufgabe12.java

Pflichtaufgabe 15: In dem Computerspiel Minecraft (die digitale Variante von Lego) wurde die in Abbildung 2 gezeigte Landschaft nachgebaut. Nun hat es im Spiel geregnet und die Landschaft füllt sich mit Wasser, wie in Abbildung 3 ersichtlich.

Ihre Aufgabe ist es, basierend auf der Beschreibung einer solchen Landschaft zu berechnen, wie viele Blöcke Wasser diese aufnehmen kann. Alle freien Felder, die sowohl links als auch rechts von sich einen höheren Block stehen haben, können mit Wasser gefüllt werden. Die Höhe jenseits des betrachteten Bereiches ist immer 0!

Die Beschreibung einer solchen Landschaft hat die Form $[5, 3, 7, 2, 6, 4, 5, 9, 1, 2]$ (Am Beispiel von Abbildung 2). Das Ergebnis wäre (entsprechend der Anzahl der blauen Blöcke in Abbildung 3) 14.
Ihre Applikation soll als Kommandozeilenparameter eine Sequenz von Natürlichen Zahlen (≥ 0), die erste Zahl bestimmt die Breite der Landschaft, die übrigen Zahlen die Höhen der einzelnen Felder.

Der Aufruf für Abbildung 2 wäre also `java -jar mc.jar 10 5 3 7 2 6 4 5 9 1 2`.

Gewünschte Features:

- Behandlung des ersten Kommandozeilen Parameters und Anlegen eines entsprechend großen Arrays für die Landschaft.
- Füllen des Arrays mit der Beschreibung der Landschaft, basierend auf den weiteren Kommandozeilen Parametern
- Simple Fehlerbehandlung (z.B. falsche Anzahl an Parametern sollte zu einer Fehlermeldung und zum Beenden des Programmes führen)
- Ausgabe der initialen Landschaft (z.B. `X` für gefüllte Blöcke)
- Ermitteln, wie viele Blöcke mit Wasser gefüllt werden können.
- Ausgabe der gefüllten Landschaft (z.B. `~` für mit Wasser gefüllte Blöcke) (Achtugn, schwieriger als die vorherige Punkte)

Für die Umsetzung von jedem der obigen Teilaspekte erhalten Sie jeweils einen Punkt. Haben Sie mindestens einen der Teilaspekte korrekt bearbeitet, können Sie einen weiteren Punkt für Einhaltung der Programmierrichtlinien erhalten. Diese sind:

- Verwendung des entsprechend (`Blatt2`) Pflichtübungsordners, in dem der Quellcode abgelegt wird.
- Formatierung des Programms gemäß der Formatierungsrichtlinien.
- Javadoc Kommentare für jede Klasse, jedes Interface und jede public Methode.

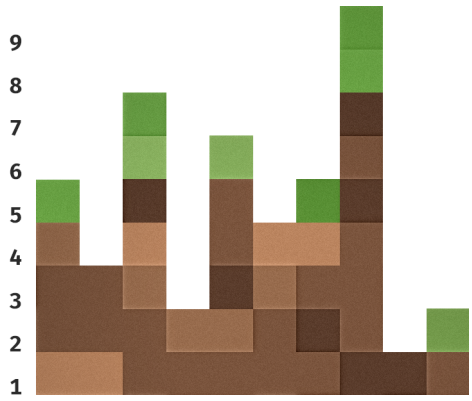

Abbildung 2: Die Ursprüngliche Lanschaft.

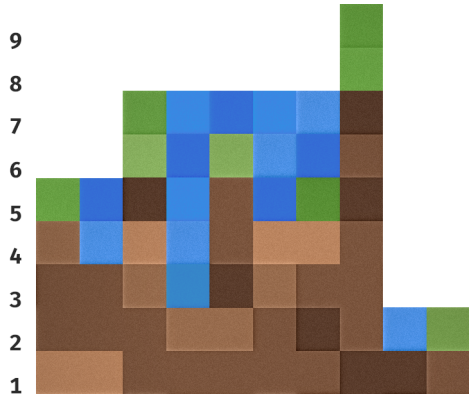

Abbildung 3: Die “gefüllte” Landschaft.

- Keine überflüssigen Dateien (hier primär `.class` und `.jar` Datein, aber auch Sonstiges, das nicht unter Quelltext oder Dokumentation fällt) im Git.

Für das Lösen dieser Pflichtaufgabe können Sie maximal 7 Punkte erhalten.
Berücksichtigt werden ausschließlich Ergebnisse, die bis zum 06.12.2019 um 23:59 in das Git Repository Ihrer Gruppe gepusht wurden.
Sie müssen die Lösung in der Woche vom 09.12. - 13.12. dem Tutor/der Tutorin in Ihrer Übungsgruppe vorstellen.
Nur compilierende Programme gelten als Lösung! Ein Programm das Teillösungen beinhaltet aber nicht compiliert oder aus anderen Gründen nicht ausführbar ist erhält 0 Punkte.