Programmierung 1

Aufgabenblatt 4 (Prozeduren/Funktionen)

Prof. Dr.-Ing. Heiko Tapken / Programmier-Team

Wintersemester 2020/21

Abgabe 15.11.2020, 12 Uhr

Bearbeitung KW 45-46, Testat KW 47

Erreichbar: 20 Punkte, Bestehensgrenze: 15 Punkte

Hochschule Osnabrück
University of Applied Sciences

Aufgabe 1 (Alternative) [3 Punkte]

Gegeben sei die folgende Klasse mit neun Methoden (static-Schlüsselwort nicht dargestellt). Berechnen Sie selbst, welche Rückgaben die Funktionen liefern, wenn Sie alle vier möglichen Varianten von Parametern jeweils für neue Objekte durchprobieren. Von welchen der Methoden würden Sie sagen, dass sie äquivalent sind, also bezüglich der gleichen Übergabeparameter immer das gleiche Ergebnis liefern?

```
class IfAnalyseKlasse {

 int x=42;

 int methode0(boolean a, boolean b){
    if(a==true && b==true){
      x=0;
    }
    return x;
  }
 int methode1(boolean a, boolean b){
    if(a==true){
      x=0;
      if(b==true){
        x=0;
      }
    }
    return x;
  }
 int methode2(boolean a, boolean b){
    if(a==true){
      if(b==true){
        x=0;
      }
    }
    return x;
  }
 int methode3(boolean a, boolean b){
    if(a==true){
      x=0;
    } else {
      if(b==true){
        x=0;
      }
    }
    return x;
  }
 int methode4(boolean a, boolean b){
    if(a==true){
      x=0;
    }
    if(b==true){
      x=0;
    }
    return x;
  }

 int methode5(boolean a, boolean b){
    if(a==true){
      if(b==true){
        x=0;
      }
    } else {
        x=0;
    }
    return x;
  }

 int methode6(boolean a, boolean b){
    if(a==true){
      x=0;
    }
    if(b==true){
       x=0;
    } else {
       x=0;
    }
    return x;
  }

 int methode7(boolean a, boolean b){
    if(a==true && b==true){
      x=0;
    } else {
    }
    return x;
  }

 int methode8(boolean a, boolean b){
    if(a==true){
      if(!(b==true)){
      } else {
         x=0;
      }
    }
    return x;
```

Programmierung 1

Aufgabenblatt 4 (Prozeduren/Funktionen)

Prof. Dr.-Ing. Heiko Tapken / Programmier-Team

Wintersemester 2020/21

Abgabe 15.11.2020, 12 Uhr

Bearbeitung KW 45-46, Testat KW 47

Erreichbar: 20 Punkte, Bestehensgrenze: 15 Punkte

Hochschule Osnabrück
University of Applied Sciences

```
}
```

```
}
```

Aufgabe 2 (Schleifen und Funktionen) [5 Punkte]

Schreiben Sie eine neue Klasse `Zahlenanalyse`, mit der Sie das Erstellen von Funktionen einüben sollen. Bei den zu entwickelnden Methoden kann es immer sinnvoll sein, sich den Ablauf zunächst zu überlegen und dabei noch keinen konkreten Programmcode in die Aktionen zu schreiben. Machen Sie solchen Skizzen zunächst per Hand, durchaus mit schwerer zu lesenden Korrekturen, verschwenden Sie keine Zeit mit der Suche nach einem Werkzeug.

a) Schreiben Sie eine Methode `istTeilerVon(int, int)`, die feststellt, ob die erste übergebene Zahl Teiler der zweiten übergebenen Zahl ist. Nutzen Sie die Modulo-Rechnung mit %. Falls die erste Zahl 0 ist, soll das Ergebnis `false` sein.

b) Schreiben Sie eine Methode `alleTeilerVon(int)`, die alle Teiler der übergebenen Zahl ausgibt. Für Zahlen kleiner 1 wird nichts ausgegeben. Überprüfen Sie innerhalb einer Schleife alle möglichen Teiler ab der Zahl 1.

c) Schreiben Sie eine Methode `groessterGemeinsamerTeiler(int, int)`, der die größte Zahl findet, die Teiler beider übergebenen Zahlen ist. Nutzen Sie dabei ein einfaches Verfahren, bei dem Sie z. B. alle Zahlen von 1 bis zur ersten übergebenen Zahl ausprobieren. Sollte ein Parameter kleiner 1 sein, ist das Ergebnis 0.

d) Zu entwickeln ist eine Methode `statistiken()`, die den Nutzer immer wieder zur Eingabe einer Fließkommazahl (`double`) auffordert, die immer zwischen -100.0 und 100.0 liegt (muss geprüft werden, falsche Werte ignorieren) und die nach jeder Eingabe den kleinsten und den größten bisher eingegebenen Wert sowie den Mittelwert (Durchschnitt aller eingegebenen Werte) ausgibt. Überlegen Sie, welche Informationen man in lokalen Variablen speichern muss (man muss sich nicht alle Zahlen merken). Das Programm wird mit der Eingabe der Zahl -1 terminiert. Ein Nutzungsdialog kann wie folgt aussehen.

```
Geben Sie einen Wert ein: 24
Minimum: 24.0  Maximum: 24.0  Schnitt: 24.0
Geben Sie einen Wert ein: 22
Minimum: 22.0  Maximum: 24.0  Schnitt: 23.0
Geben Sie einen Wert ein:
…
```

e) Entwickeln Sie eine Methode `zahlenRaten()` mit der Sie mit dem Computer „Zahlenraten“ spielen können. Dazu soll der Computer eine Zahl zwischen 1 und 100 (jeweils einschließlich) wählen. Danach kann der Spieler seinen Tipp eingeben und der Computer antwortet mit „zu hoch“, „zu tief“ oder „korrekt“. Weiterhin zählt der Computer die Anzahl der Versuche und gibt dies vor jedem Rateversuch aus. Ein Nutzungsdialog kann wie folgt aussehen, Eingaben sind umrandet.

Programmierung 1

Aufgabenblatt 4 (Prozeduren/Funktionen)

Prof. Dr.-Ing. Heiko Tapken / Programmier-Team
Wintersemester 2020/21
Abgabe 15.11.2020, 12 Uhr
Bearbeitung KW 45-46, Testat KW 47
Erreichbar: 20 Punkte, Bestehensgrenze: 15 Punkte

Hochschule Osnabrück
University of Applied Sciences

```
1. Versuch: Zahl raten: 12
zu tief
2. Versuch: Zahl raten: 13
zu tief
3. Versuch: Zahl raten: 20
zu tief
4. Versuch: Zahl raten: 60
zu hoch
5. Versuch: Zahl raten: 50
zu hoch
6. Versuch: Zahl raten: 40
zu hoch
```

```
7. Versuch: Zahl raten: 35
zu hoch
8. Versuch: Zahl raten: 30
zu hoch
9. Versuch: Zahl raten: 25
zu hoch
10. Versuch: Zahl raten: 20
zu tief
11. Versuch: Zahl raten: 22
zu hoch
12. Versuch: Zahl raten: 21
Bravo, getroffen
```

Aufgabe 3: Reaktionsspiel [4 Punkte]

Bei dieser Aufgabe geht es darum, ein Programm zu entwickeln, das es erlaubt, die Reaktionszeit der Nutzer zu messen.

Erstellen Sie zunächst eine Java-Klasse und KOPIEREN die unten bereitgestellten Methoden dort hinein. Es ist nicht nötig, die konkreten Inhalte zu verstehen. Sie müssen nur wissen, wie sie diese Methoden nutzen können und was sie zurückliefern.

Der Programmablauf sieht folgendermaßen aus:

- Zunächst wird „Achtung: Start“ ausgegeben.
- Dann werden zwischen 5 und 10 Runden absolviert. Die genaue Zahl der Runden wird per Zufall bestimmt. (getRandomNumber(<max>))
- In jeder Runde wird folgendes gemacht:
    - Es wird zunächst zwischen 2 und 5 Sekunden gewartet. Nutzen Sie dazu die Methode wait(int seconds). Die genaue Wartezeit wird per Zufall bestimmt.
    - Dann bestimmt das Programm per Zufall einen Kleinbuchstaben (‚a‘ – ‚z‘) und gibt ihn auf dem Bildschirm aus. (Alternative: Zahl)
    - Der Benutzer muss jetzt versuchen, so schnell wie möglich den Buchstaben einzugeben (inkl. <Enter>). Die Zeit zwischen Ein- und Ausgabe wird gemessen. Die jeweils aktuelle Zeit erhalten durch Aufruf der Methode getMilliSeconds();
    - Falls der ausgegebene Kleinbuchstabe und das eingegebene Zeichen gleich sind, wird die Zeit zwischen Ausgabe des Buchstabens und Eingabe des Nutzers verarbeitet (die Reaktionszeit). Ansonsten merkt sich das Programm einen Fehler.
- Sind alle Runden absolviert, gibt das Programm „Geschafft: Ende“ aus. Weiterhin werden ausgegeben:
    - Anzahl an Fehlversuchen
    - Mittelwert der Reaktionszeiten
    - Langsamster Versuch
    - Schnellster Versuch

Programmierung 1

Aufgabenblatt 4 (Prozeduren/Funktionen)

Prof. Dr.-Ing. Heiko Tapken / Programmier-Team

Wintersemester 2020/21

Abgabe 15.11.2020, 12 Uhr

Bearbeitung KW 45-46, Testat KW 47

Erreichbar: 20 Punkte, Bestehensgrenze: 15 Punkte

Hochschule Osnabrück
University of Applied Sciences

Programmierung 1

Aufgabenblatt 4 (Prozeduren/Funktionen)

Prof. Dr.-Ing. Heiko Tapken / Programmier-Team

Wintersemester 2020/21

Abgabe 15.11.2020, 12 Uhr

Bearbeitung KW 45-46, Testat KW 47

Erreichbar: 20 Punkte, Bestehensgrenze: 15 Punkte

Hochschule Osnabrück
University of Applied Sciences

Beispiel für einen Programmablauf (Benutzereingaben stehen in Klammern (<>)):

```
Achtung: Start!
b
<b>
u
<u>
v
<d>
m
<m>
a
<a>
Geschafft: Ende!
Fehlversuche: 1 von 5
Reaktionszeit-Mittelwert: 2.5748 Sekunden
Langsamster Versuch: 5.813 Sekunden
Schnellster Versuch: 1.218 Sekunden
```

Folgende Funktionen sind dabei vorgegeben:

```
  // liefert eine Zahl, die die aktuelle Zeit in Millisek
repräsentiert
  static long getMilliSeconds() {
    return new java.util.Date().getTime();
  }

  // liefert eine Zufallszahl zwischen 0 und max (einschließlich)
  static int getRandomNumber(int max) {
    return new java.util.Random().nextInt(max + 1);
  }

  // hält das Programm seconds Sekunden an
  static void wait(int seconds) {
    try {
      Thread.sleep(seconds * 1000);
    } catch (Exception exc) {
    }
  }

  // liefert den größt-möglichen long-Wert
  static long getMaxLongNumber() {
    return Long.MAX_VALUE;
  }
```

Programmierung 1

Aufgabenblatt 4 (Prozeduren/Funktionen)

Prof. Dr.-Ing. Heiko Tapken / Programmier-Team
Wintersemester 2020/21
Abgabe 15.11.2020, 12 Uhr
Bearbeitung KW 45-46, Testat KW 47
Erreichbar: 20 Punkte, Bestehensgrenze: 15 Punkte

Hochschule Osnabrück
University of Applied Sciences

Aufgabe 4 (Mathetrainer) [4 Punkte]

Bei dieser Aufgabe sollen Sie einen Mathetrainer zum Üben des Multiplizierens und Dividierens implementieren.

Der Mathetrainer generiert dabei jeweils zufällig zwei int-Werte zwischen 0 und 9 sowie zufällig entweder den Multiplikations- oder den Divisionsoperator. Er präsentiert die Aufgabe und fordert den Benutzer auf, das Ergebnis einzugeben. Gibt dieser das falsche Ergebnis ein, teilt ihm der Mathetrainer das korrekte Ergebnis mit. Gibt der Benutzer das korrekte Ergebnis ein, wird er gelobt. In beiden Fällen wird anschließend die Gesamtanzahl der bisherigen korrekten Antworten ausgegeben. Das Ganze wird wiederholt, bis der Benutzer 10 korrekte Antworten gegeben hat.

Beispiel für einen Programmablauf (in <> stehen Benutzereingaben):

```
Start des Mathetrainers
8 / 1 = <8>
Richtig!
Korrekte Antworten: 1
9 / 2 = <4>
Richtig!
Korrekte Antworten: 2
6 * 1 = <4>
Leider falsch! Korrektes Ergebnis ist 6
Korrekte Antworten: 2
0 / 8 = <0>
Richtig!
Korrekte Antworten: 3

...

3 * 2 = <6>
Richtig!
Korrekte Antworten: 9
0 * 5 = <0>
Richtig!
Korrekte Antworten: 10
Ende des Mathetrainers
```

Hinweis: Implementieren und verwenden Sie dabei geeignete Funktionen, um Ihr Programm übersichtlich zu gestalten, z.B.:

- `static boolean frageBearbeiten()` zum Bearbeiten einer einzelnen Frage
- `static int generiereZahl()` zum Generieren einer Zahl zwischen 0 und 9
- `static char generiereOperator()` zum Generieren einer der beiden Operatoren

Programmierung 1

Aufgabenblatt 4 (Prozeduren/Funktionen)

Prof. Dr.-Ing. Heiko Tapken / Programmier-Team

Wintersemester 2020/21

Abgabe 15.11.2020, 12 Uhr

Bearbeitung KW 45-46, Testat KW 47

Erreichbar: 20 Punkte, Bestehensgrenze: 15 Punkte

Hochschule Osnabrück
University of Applied Sciences

Aufgabe 5 „Schnick-Schnack-Schnuck“ [4 Punkte]

Sie sollen das bekannte Spiel „Schnick-Schnack-Schnuck“ so implementieren, dass es der Computer gegen einen Menschen spielen kann.

Regeln: Bei Schnick-Schnack-Schnuck werden mehrere Spielrunden absolviert. In jeder Spielrunde wählen die beiden Spieler gleichzeitig eines der folgenden Symbole: Brunnen, Schere, Stein oder Papier. Dabei gilt:

- Brunnen gewinnt gegen Schere
- Brunnen gewinnt gegen Stein
- Brunnen verliert gegen Papier
- Schere verliert gegen Stein
- Schere gewinnt gegen Papier
- Stein verliert gegen Papier

Der Sieger erhält jeweils einen Punkt. Wählen die beiden Spieler dasselbe Symbol, ist das ein Unentschieden und keiner der beiden Spieler erhält einen Punkt.

Ein Spiel endet, wenn ein Spieler insgesamt 10 Punkte erreicht. Dieser Spieler ist Sieger.

Ablauf: In jeder Spielrunde generiert der Computer per Zufall eines der vier Symbole. Anschließend fordert er den menschlichen Spieler zur Eingabe eines Symbols auf. Er ermittelt das Ergebnis und gibt es auf den Bildschirm aus.

```
Spielrunde 1:
Symbol eingeben (Brunnen, Schere, Stein, Papier): -> Schere
Computer: Brunnen; Mensch: Schere -> Computer gewinnt
Spielstand: Computer = 1; Mensch = 0
…
Spielrunde 21:
Symbol eingeben (Brunnen, Schere, Stein, Papier): -> Stein
Computer: Brunnen; Mensch: Stein -> Computer gewinnt
Spielstand: Computer = 10; Mensch = 5

Spielende: Computer hat gewonnen
```