Universität Stuttgart

FIUS

PSE Vorkurs Tag 2

Heute Abend gibts Karaoke!

Feedback

Bitte füllt den Feedbackbogen aus, damit wir den Vorkurs für die nächsten Jahre verbessern können. Der Bogen ist anonym und dauert nur 2-3 Minuten.
Ihr findet ihn unter: `https://forms.gle/bjVu3k4KAfTsQQL58`

Hilfe suchen und finden

Beim Programmieren sind Google und Dokumentationen der Sprache/Umgebung die ihr nutzt sehr hilfreich. Wenn ihr ein Problem habt, kann es neben Tutoren fragen auch sinnvoll sein einfach mal eine Fehlermeldung oder eine neue Funktion im Internet zu suchen. Die offizielle Dokumentation für Java ist hier: `https://docs.oracle.com/javase/8/docs/api/`.

Musterlösungen

Es wird auf jedem Ausgabenblatt ein Link zu den Musterlösungen des Vortags geben, falls es noch Unklarheiten gibt oder ihr sie mit euren Lösungen vergleichen wollt: Tag 1 Lösungen

Präsentations-Cheatsheet

Booleanoperatoren:

- Boolean: Wahrheitswerte `true` und `false`
- Vergleiche: ==, !=, <, >, <=, >=
- Logische Operatoren: && (UND), || (ODER), ! (NICHT)
- `if`, `else`, `else if` zur Entscheidungslogik

```
if (x > 5 && x < 10) {
  System.out.println("x ist zwischen 5 und 10");
} else {
  System.out.println("x ist außerhalb");
}
```

Schleifen:

- `while`-Schleife: läuft, solange Bedingung `true` ist

```
while (!eingabe.equals("ok")) {
  eingabe = scanner.nextLine();
}
```

- `for`-Schleife: Zählergesteuerte Schleife

```
for (int i = 0; i < 5; i++) {
  System.out.println(i);
}
```

- `break`: beendet Schleife vorzeitig

Aufgabe 1: Boole'sche Ausdrücke und Verzweigungen

1. Schreibe für die UNO diesen Freitag (die Party zum Start des Wintersemesters) ein Programm, bei dem das Alter der Studis geprüft wird und ausgegeben wird, ob die Person auf die Party gelassen wird oder nicht. Frage mit dem Scanner vom Nutzer das Alter ab. Erinnerung an die Scanner-Syntax:
   `https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html`

   ```
   Wie alt bist du: 17
   Sorry, du darfst noch nicht rein :/
   ```

2. Füge dem Programm einen Preiskalkulierer hinzu, der mit dem Scanner einliest ob die Person Ersti ist oder nicht (Tipp: lies einen boolean Wert ein). Bei der UNO zahlen Erstis nämlich nur 3 Euro, alle anderen aber 7 Euro. Vergiss aber nicht, dass ja alle unter 18 leider nicht reindürfen und deswegen keinen Preis ausgegeben bekommen sollten.

   ```
   Bist du ein Ersti (true oder false eingeben): true
   Für dich sinds 3 Euro
   ```

3. Jetzt fehlen nur noch Drinks, frage ab wie viel Geld die Person dabei hat und jeweils wieviele Shots (1 Euro), Softdrinks (2 Euro), und Biere (2 Euro) die Person bestellen will. Gib am Ende aus ob das Geld für die Bestellung ausreicht.

   ```
   Wie viel Geld hast du: 13
   Wie viele Shots hättest du gerne: 1
   Wie viele Softdrinks hättest du gerne: 6
   Wie viele Biere hättest du gerne: 4
   Das sind insgesamt 21 Euro
   Du hast zu wenig Geld dabei
   ```

Aufgabe 2: Schleifen

1. Wir arbeiten weiter mit dem UNO Programm aus Aufgabe 1. Implementiere mithilfe von for-Schleifen, dass nachdem eine Bestellung abgegeben wurde, jedes Getränk nummeriert ausgegeben wird. Das soll natürlich nur passieren wenn die Getränke auch bezahlt werden können:

```
Shot 1 ist eingeschenkt
Shot 2 ist eingeschenkt
Shot 3 ist eingeschenkt
Sofdrink 1 ist fertig
Bier 1 ist geöffnet
Bier 2 ist geöffnet
```

2. Da Informatiker ja bekanntlich nicht rechnen können gehen wir davon aus, dass es oft passiert das eine Person mehr Shots will als sie sich leisten kann. Nutze eine While Schleife um solange neue Getränkeanfragen einzulesen, bis die Person sich ihre Bestellung leisten kann, nutze `break` um dann aus der Schleife auszubrechen.

```
Wie viel Geld hast du? 13
Wie viele Shots hättest du gerne? 1
Wie viele Softdrinks hättest du gerne? 6
Wie viele Biere hättest du gerne? 4
Das sind insgesamt 21 Euro
Du hast zu wenig Geld, probier eine andere Bestellung
Wie viele Shots hättest du gerne? 1
...
```

Aufgabe 3: Palindrome

Es kann für diese Aufgaben sinnvoll sein Funktionen aus dem `Math` Package zu verwenden, z.B. `Math.log(eineZahl)` für den Logarithmus einer Zahl. Die Dokumentation dafür findet ihr hier: `https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html`

1. Schreibe ein Programm, dass eine Zahl einliest und sie zweimal hintereinander schreibt. Nutze dafür keine String oder Char Variablen.

   ```
   Gib eine Zahl ein: 38429
   Zahl zwei mal hintereinander: 3842938429
   ```

2. Schreibe ein Programm, dass vom Nutzer eine Zahl einliest und prüft, ob diese ein Palindrom ist (Vorwärts und Rückwärts gelesen gleich ist, z.b.: 4165614 oder 829928). Nutze keine String oder Char Variablen.

3. Schreibe ein Programm, dass vom Nutzer eine Zahl einliest und sie Rückwärts geschrieben wieder ausgibt. Nutze natürlich wieder keine String oder Char Variablen.

HIGHPERFORMER-Aufgabe: Formeln und Erfüllbarkeit

In der Highperformer Aufgabe heute geht es um die Erfüllbarkeit von Formeln. Eine Formel ist ein Boole'scher Ausdruck.

Formeln werden in drei Kategorien unterteilt: immer wahr (Tautologien), immer falsch (unerfüllbar) oder je nach den Werten der Variablen manchmal wahr und manchmal falsch (erfüllbar).

Tautologie (ist immer wahr): `(a || !a)`, da `a` entweder wahr oder falsch sein muss, somit ist die oder Verknüpfung von `a` und `!a` immer wahr.
Unerfüllbar (ist immer falsch): `(a && !b && !a && c)`, da `a` zugleich wahr und falsch sein müsste
Erfüllbar (kann wahr oder falsch sein): `((a && c) || b)`, da sie wahr wird wenn z.b. `b = true` ist, jedoch keine Tautologie, da die Gesamtformel auch falsch sein kann, bspw. mit `a = false`, `c = true` und `b = false`

1. Prüfe folgende Boole'sche Ausdrücke mit einem Programm auf ihre Erfüllbarkeit indem du die jeweiligen Ausdrücke nacheinander in deinen Code kopierst und gib dann erfüllbar, unerfüllbar oder Tautologie aus.
   Hinweis: Die Formeln haben bis zu 8 Variablen (a-h)

   (a)
   ```
   (!(a && b) || a) && (!(c && d) || c) && (e || !e)
   ```

   (b)
   ```
   ((a && b && c) && !(a || (b || c))) && d
   ```

   (c)
   ```
   ((a && !b && !c) || (!a && b && d)) && (e || f)
   ```

   (d)
   ```
   ((a && b) || (!a || b) || (!b)) && ((c && d) || (!c || d) || (!d)) &&
   ((e && f) || (!e || f) || (!f)) && ((g && h) || (!g || h) || (!h))
   ```

   (e)
   ```
   ((a && b && !c) && (!a || !b || c)) && ((d && !e && f) && (!d || e ||
   !f)) && ((g && h) && (!g && !h))
   ```

   (f)
   ```
   (!(a && b && c) || (a && (b || c))) || (d || !d)
   ```

   (g)
   ```
   ((a && b && !c && !d && !e) || (!a && c && d && !b && !e) || (!b && !c
   && !d && e && f)) && (!g || h)
   ```

2. Modifiziere nun dein Programm: Wenn eine Formel erfüllbar ist, soll zusätzlich ausgegeben werden, wie viele Belegungen die Formel wahr oder falsch machen und wie groß die Wahrscheinlichkeit ist, dass eine zufällige Belegung wahr die Formel wahr werden lässt.