Technische Universität Braunschweig

IAS | INSTITUTE FOR APPLICATION SECURITY

Programmieren I

Blatt 2

Prof. Dr. Martin Johns, Marius Musch, Malte Wessels, Samuel Sandrock

Melden Sie sich bitte für die Veranstaltung bei Stud.IP an und tragen Sie sich in einer der kleinen Übungen ein.

Verspätete oder nicht hochgeladene Abgaben, Abgaben, die per Mail getätigt werden, sowie Plagiate führen zur Bewertung des gesamten Blatts mit **0 Punkten**.

Insgesamt können Sie für das Lösen der Pflichtaufgabe **13Punkte** erhalten.
Berücksichtigt werden ausschließlich Ergebnisse, die bis zum **21.12.2022 um 23:59** in den bereits für dieses Übungsblatt angelegten Ordner in Ihrem Git Repository auf den master-Branch gepusht wurden.

Nur kompilierende Programme gelten als Lösung! Ein Programm das Teillösungen beinhaltet, aber nicht kompiliert oder aus anderen Gründen nicht ausführbar ist, erhält **0 Punkte**.

Für die meisten Aufgaben geben wir Ihnen Vorgaben, wie die von Ihnen zu erstellenden Klassen zu heißen haben oder in welchen Ordner sie abgelegt werden sollen. Als top-level Ordner gilt immer der nach diesem Blatt benannte Ordner in Ihrem Repository, für dieses Blatt also *blatt2*.
Die Vorgaben sehen wie folgt aus.

Vorgaben

Ordner: pflichtaufgabe0
Klassen: Klasse1, Klasse2, Klasse3

Sie würden also für die Pflichtaufgabe 0 den Ordner *pflichtaufgabe0* im Ordner *blatt2* anlegen und dort mindestens die Klassen *Klasse1.java, Klasse2.java, Klasse3.java* erstellen. Wenn Sie weitere Klassen erstellen wollen um es sich selbst einfacher zu machen, steht Ihnen das natürlich frei.

Contents

Aufgabe 1: switch-case

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 2: Operatoren

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

Aufgabe 3: Fehler im Programm

Das nachfolgende Programmfragment weist Fehler auf. Finden Sie diese. Begründen Sie Ihre Antwort.

```
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);
```

Style - 3 Punkte

Ab diesem Übungsblatt gibt es immer insgesamt 3 Punkte für Style über das gesamte Blatt (alle Pflichtaufgaben) zu erreichen. Davon gibt es je einen Punkt für sinnvolle Variablennamen, ausreichende Kommentierung und korrekte Einrückung.

Beachten Sie also, Ihren gesamten Code spätestens am Ende noch sinnvoll zu überarbeiten und übersichtlich zu halten! Wird eine Aufgabe gar nicht oder nur sehr unzureichend bearbeitet, so gibt es dafür allerdings auch keine Style-Punkte.

Pflichtaufgabe 1: FizzBuzz - 3 Punkte

Vorgaben

Ordner: pflichtaufgabe1
Klassen: FizzBuzz

In dieser Aufgabe sollen Sie das sogenannte Fizz-Buzz-Spiel implementieren, welches der Computer spielen sollte.

Bei dem Spiel geht es darum, von eins aus hochzuzählen; Jede durch drei teilbare Zahl wird dabei durch *"Fizz"* und jede durch fünf teilbare Zahl wird durch *"Buzz"* ersetzt. Eine durch fünf und drei teilbare Zahl (z.B. die 15) soll durch *"Fizz-Buzz"* ersetzt werden.

Lesen sie die Zahl *n* bis zu der der Computer zählen soll, zunächst als Kommandozeilenparameter ein. Erzeugen Sie ein Array, dass zuerst eine Liste aller Zahlen von 1 bis n beinhaltet. Ergänzen Sie dann ihr Programm so, dass an den richtigen Stellen Fizz, Buzz und Fizz-Buzz in das Array eingesetzt werden. Die resultierende Folge soll in der Konsole ausgegeben werden.

Hinweis: Der Aufruf"java FizzBuzz 20" soll schlussendlich "1,2,Fizz,4,Buzz,Fizz,7,8,Fizz,Buzz,11,Fizz,13,14,FizzBuzz,16,17,Fizz,19,Buzz" ausgeben

Bonus (nicht bewertet): Überprüfung

Hinweis: Wenn Sie diese Aufgabe programmieren wollen, machen Sie es bitte in einem anderen Ordner als die Pflichtaufgabe, da dieser Teil nicht gewertet wird.
Programmieren Sie das Fizz-Buzz-Spiel so um, dass nun der Spieler über die Konsole hochzählt. Die Klasse hat nun die Aufgabe, zu Überprüfen, ob die Eingaben des Spielers korrekt sind. Achten Sie hierbei darauf, dass der Spieler korrekt hochzählen muss und die Ausdrücke Fizz, Buzz und Fizz-Buzz korrekt verwenden muss.

Gibt der Spieler etwas falsches ein, soll dies in der Konsole ausgegeben werden, zusammen mit der Korrekten *"Zahl"* und den String aller korrekter Zahlen, wie er im ersten Teil ausgegeben wird.

Pflichtaufgabe 2: Mastermind - 7 Punkte

Vorgaben

Ordner: pflichtaufgabe2
Klassen: Mastermind

Im beliebten Spiel *Mastermind* geht es darum, einen 4-stelligen Farbcode mit maximal 10 Versuchen zu erraten. Um dem/der Ratenden zu helfen, wird ihm/ihr nach jedem Versuch mitgeteilt, wie viele richtige Farben an der richtigen Stelle sind und wie viele richtige Farben an der falschen Stelle sind. Da Farben in der Kommandozeile recht aufwändig darzustellen sind, werden wir diese durch die Ziffern 1 bis 8 ersetzen.

Erstellen Sie eine Klasse Mastermind. Lassen Sie den Rechner einen zufälligen 4-stelligen Code erstellen, der nur aus Ziffern von 1 bis 8 besteht (mehrfaches Verwenden einer Ziffer ist nicht erlaubt).

Eine zufällige Zahl erstellen Sie mithilfe der in `java.util` gegeben Klasse Random. Wie Sie dies schön machen, hier noch einmal erläutert: Die Klasse Random muss zunächst importiert werden. Hierfür schreiben Sie `import java.util.Random;` über Ihre Klasse. Nachdem Sie die Klasse nun importiert haben, können Sie mit `Random random = new Random();` eine Instanz dieser erzeugen. Diese Instanz kann nun benutzt werden, um "zufällige" Integer zu erzeugen. Schreiben Sie hierfür `random.nextInt(<bound>);`. Dies generiert einen Integer zwischen 0 (inklusive) und dem angegeben bound, also Höchstwert (exklusive).

Um Beispielsweise zufällige Zahlen auf der Kommandozeile auszugeben, könnte diese Klasse benutzt werden:

```
import java.util.Random;

public class RandomPrint {

    /**
    * Prints random numbers between 0 and 100 (inklusiv)
    */
    public void printRandoms() {
        Random random = new Random();
        System.out.println(random.nextInt(101));
    }
}
```

Nun ist der Spieler am Zug. Erstellen Sie eine Eingabeaufforderung, um dem Spieler mitzuteilen, dass er jetzt raten kann. Lesen Sie die Eingabe des Spielers mit einem Scanner ein. Diesen sollten Sie bereits aus dem letzten Übungsblatt kennen. Hier aber noch einmal das Wichtigste: Um die Java Klasse Scanner nutzen zu können muss sie zunächst mit `import java.util.Scanner;` importiert werden. Die Scanner Instanz kann dann mit der Codezeile
`Scanner input = new Scanner(System.in);` erstellt werden. "input" ist hierbei der Variablenname

und kann frei gewählt werden. Mit der folgenden Zeile können Zeichenketten aus der Kommandozeile eingelesen und beispielsweise in einer Variablen gespeichert werden
`String eingabe = input.next();`.

Schreiben Sie eine Funktion, die die Eingabe des Spielers als Parameter erhält und dann ausgibt, wie viele richtige Ziffern an der richtigen Stelle und wie viele Ziffern zwar richtig, aber an der falschen Stelle sind. Teilen Sie dem Benutzer das Ergebnis seines Versuchs in geeigneter Form mit.

Sie können die Länge eines Strings mit `int len = eingabe.length();` speichern und mit `eingabe.charAt(0)` das erste Zeichen abfragen. Sollte eine Zeichenfolge der falschen Länge eingegeben werden, soll darauf hingewiesen werden und der Spieler eine neue Eigabe machen müssen.

Passen Sie ihr Programm so an, dass maximal 10 Versuche möglich sind. Errät der Spieler den Code innerhalb dieser 10 Versuche, hat der Spieler gewonnen, ansonsten gewinnt das Programm. Erstellen Sie eine passende Ausgabe auf Basis des Ergebnisses.

Bonus (nicht bewertet): Machen wir es Schwerer

Hinweis: Wenn Sie diese Aufgabe programmieren wollen, machen Sie es bitte in einem anderen Ordner als die Pflichtaufgabe, da dieser Teil nicht gewertet wird.
Um den Schwierigkeitsgrad des Spiels anpassbar zu machen, soll die Anzahl der möglichen Ziffern bei Programmstart als Argument übergeben werden. Das Programm mit den Zahlen 1 - 8 könnte mit `java Mastermind 8` gestartet werden.
Wenn Ihnen das immer noch nicht reicht, überlegen Sie sich, wie das Spiel mit Mehrfachnutzung von Zahlen (bspw. 4436) funktioniert.

A Links

- Stud.IP: `https://studip.tu-braunschweig.de`
- Git Dokumentation: `https://git-scm.com/`
- Git des IAS: `https://programming.ias.cs.tu-bs.de`
- Allgemeine Java API Dokumentation:
  `https://docs.oracle.com/en/java/javase/11/docs/api/index.html`
- java.util.Random Dokumentation `https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Random.html`