Technische Universität Braunschweig

IAS | INSTITUTE FOR APPLICATION SECURITY

Programmieren I

Blatt 2

Prof. Dr. Martin Johns, Robert Michael, Alexandra Dirksen, Daniel Wolfram, Patrick Glöckner

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 **15 Punkte** erhalten. Bei Erreichen eines Punktes in den Pflichtaufgaben können Sie zusätzlich drei Punkte für das Beachten der Checkstyle Vorgaben erhalten. Hierbei gibt es einen Zusatzpunkt für das Schreiben von Javadoc Kommentaren, einen für das Einhalten der Formatierungsrichtlinien und einen für das sinvolle Benennen der Variablen.

Berücksichtigt werden ausschließlich Ergebnisse, die bis zum **08.12.2021 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);
```

Pflichtaufgabe 1: Der Code des Lebens? - (5 Punkte)

Vorgaben

Ordner: pflichtaufgabe1
Klassen: Fibonacci

Schreiben Sie ein Programm, welches die Zahlen der Fibonacci-Folge bis zu einem bestimmten Grenzwert ausgibt. Der Grenzwert soll hierbei als Kommandozeilenargument eingelesen werden. Eine Zahl der Fibonacci-Folge lässt sich mit der Formel $f_n = f_{n-1} + f_{n-2}$ errechnen. Gestartet wird mit $f_0 = 0$ und $f_1 = 1$. Implementieren Sie den Algorithmus und geben Sie die Fibonacci-Zahlen bis zu dem Grenzwert auf der Kommandozeile aus.

Für die Zahlen kleiner 100:

```
1
1
2
3
5
8
13
21
34
55
89
```

Pflichtaufgabe 2: Das Problem mit dem Virus - (10 Punkte)

Vorgaben

Ordner: pflichtaufgabe2
Klassen: Virus

In dieser Aufgabe wollen wir eine vereinfachte Verbreitung eines Virus simulieren. Jeden Tag kann der Virus sich von einem Wirt auf seine Nachbar übertragen. Um dies zu entscheiden legen wir die Infektiösität des Virus durch eine Wahrscheinlichkeit fest. Durch Ausgaben auf der Kommandozeile ist der Prozess zu beobachten.

Erstellen Sie ein 2D int-Array, welches die Welt simuliert. Der Virus wird zufällig auf einem Feld des 2D Arrays ausgesetzt und beginnt sich langsam auf die Nachbarfelder zu verbreiten. Diagonal

gelegene Felder zählen nicht als Nachbarn. Die Infektiösität Ihres Virus geben Sie als Kommandozeilenargument in Form einer Wahrscheinlichkeit in Prozent an. Beachten Sie, auch Fließkommazahlen wie 31.6 sollen als Eingabe möglich sein. Ob die Eingabe mit oder ohne Prozentzeichen erfolgt ist Ihnen überlassen.

Die Größe der Welt soll ebenfalls als Kommandozeilenargument übergeben werden. Um das Korrigieren zu erleichtern soll ein dritter Parameter, welcher Debug Funktionalitäten aktiviert, als 0 (nicht aktivieren) oder 1 (aktivieren) übergeben werden. Der Programmaufruf lautet somit: java Virus <Infektiösität> <Feldgröße> <Debug>

Wenn eine Zelle infiziert wurde ist sie sofort ansteckend, allerdings nur für eine begrenze Zeit. In unserer Simulation vergehen zwischen 3 und 7 Tagen, bis eine infizierte Zelle nicht mehr infektiös und immun gegen erneute Infektionen ist. Speichern Sie die verbleibenden Tage in dem 2D Array.

Um einen Überblick über das Geschehen zu erhalten, speichern Sie die Infektionen für die letzten sieben Tage. Schreiben Sie eine Funktion, die aus den gespeicherten Daten die 7-Tage-Inzidenz Ihrer Welt errechnet, beispielsweise 23/400. Berechnen und speichern Sie den Faktor, um aus dieser Inzidenz die Inzidenz pro 100.000 zu errechnen. Dies Berechnung des Faktor muss nur einmal zu Beginn geschehen.

Um den Prozess zu visualisieren geben Sie am Ende jedes Tages Ihr 2D Array als Karte, sowie die Neuinfektionen und die Inzidenzen auf der Konsole aus. Hierbei soll "*" für infektiös, "#" als genesen und "-" als noch nicht infiziert ausgegeben werden. Wenn der Debug Modus aktiviert wird, sollen jede Runde eine zweite Karte ausgegeben werden. Diese zeigt anstelle des "*" für infektiös die Tage die die Zelle noch infektiös ist.

Der Prozess soll terminieren, wenn kein weiteres Feld mehr infiziert werden kann. Beachten Sie, dies bedeutet nicht zwingend, dass alle Felder infiziert sind.

Bonus (nicht bewertet): Impfen gegen das Virus und Virusvarianten

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 die Lage unter Kontrolle zu bekommen, wird die Bevölkerung gleichzeitig geimpft. Schaffen Sie eine Möglichkeit, das jeden Tag zufällig eine bestimmte Anzahl an Menschen geimpft wird. Geben Sie zusätzlich die Impfquote auf der Kommandozeile aus.

Ihr Virus ist mutiert und es haben sich verschiedene Varianten gebildet. Diese Varianten sind unterschiedlich infektiös. Um dies zu simulieren benötigen Sie ein weiteres 2D Array (oder ein 3D Array), denn zusätzlich zu der Frage, wie lange die Person noch infektiös ist, müssen Sie nun ebenfalls speichern mit welcher Variante sie infiziert ist.

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/javase/8/docs/api/overview-summary.html`
- java.util.Random Dokumentation `https://docs.oracle.com/javase/8/docs/api/java/util/Random.html`