Programmierpraktikum

Wintersemester 2021/22

Prof. Dr. rer. nat. habil. Petra Hofstedt
Ilja Becker M. Sc., Sven Löffler M. Sc.
Sonja Breuß B. Sc., Carlo Bückert B. Sc., Endris Wenzke

Brandenburgische
Technische Universität
Cottbus - Senftenberg

Übungsblatt 4

Abgabedatum: 14.11.2021

Die Abgabe Ihrer Lösungen erfolgt vor Ablauf der Abgabefrist digital über die Moodle-Plattform. Erstellen Sie dazu ein PDF-Dokument, das die Lösungen Ihrer schriftlichen Aufgaben enthält. Laden Sie dieses PDF-Dokument und den erarbeiteten Java-Code (.java-Dateien) mit den in den Aufgaben vorgegebenen Namen bei Moodle hoch. Bitte laden Sie die Dateien einzeln hoch, Dateiarchive (z.B. .zip-Dateien) werden nicht akzeptiert.

Sie können maximal **(7 Punkte)** mit diesem Übungsblatt erreichen.

Aufgabe 1 (Debug)

0,5 Punkte

1. Machen Sie sich mit dem Debugger vertraut. Wozu wird dieser genutzt und welche Möglichkeiten des Fortschreitens (Steps) im Programm ermöglicht dieser?
2. Erstellen Sie eine Klasse `Debug.java` mit folgendem Inhalt.

```
public class Debug {

public static void main(String[] args) {

  for (int i = 0; i < 50; i++) {
    for (int j = 0; j < 50; i++) {
      if (Math.abs(i - 25) + Math.abs(j - 25) > 25) {
        System.out.print(" ");
      } else {
        System.out.print("+");
      }
    }
    System.out.println();
  }
}
}
```

Machen Sie sich vor dem Ausführen des Programmes darüber Gedanken, was dieses bewirkt und führen Sie es dann anschließend aus.

Beenden Sie das Programm durch klicken des Terminate-Buttons (rotes Quadrat). Nutzen Sie die in Eclipse integrierten Debug-Werkzeuge, um die Ursache für das fehlerhafte Verhalten zu bestimmen. Sie erhalten dazu während Ihres Praktikums eine kurze Einführung. Beschreiben Sie den gefundenen Fehler.

Aufgabe 2 (Code Convention) 1 Punkt

Informieren Sie sich zum Beispiel in `http://de.wikipedia.org/wiki/Programmierstil` über Code Conventions.

1. Beschreiben Sie in wenigen kurzen Sätzen, was unter so genannten Code Conventions zu verstehen ist.
2. Nennen Sie einige typische Eigenschaften von Quelltexten, die durch Code Conventions geregelt werden.
3. Nennen Sie Gründe, weshalb sich Programmierer beim Verfassen von Quelltexten an Code Conventions halten sollten.

Aufgabe 3 (Java Code Conventions) 1 Punkt

Informieren Sie sich in `http://www.oracle.com/technetwork/java/codeconvtoc-136057.html` über die Java Code Conventions. Diese werden für die zukünftig im Programmierpraktikum gestellten Programmieraufgaben die verbindlichen Richtlinien zur Gestaltung der Quelltexte sein.

1. Welche Regeln sollen bei Variablendeklarationen beachtet werden?
2. Wie werden die verschiedenen erlaubten Varianten von if-else-Anweisungen formatiert? Warum sollen bei if-Anweisungen die Blockklammern { und } niemals weggelassen werden?
3. Fassen Sie die Regeln zum Einsatz von Leerzeichen (eng. blank spaces) zusammen.
4. Nennen Sie die Konventionen, die bei der Benennung von Klassen (also auch von Quelltextdateien), von Methoden, von Variablen sowie von Konstanten zu berücksichtigen sind.
5. Welche Arten von Kommentaren werden in den Java Code Conventions unterschieden?

Aufgabe 4 (Programmdokumentation mit Javadoc) 1 Punkt

Javadoc ist ein Werkzeug zur automatisierten Erzeugung von Dokumentationsdateien aus Java-Quelltexten. Dazu werden die Quelltexte mit Kommentaren in einer besonderen Syntax versehen. Aus diesen Kommentaren kann automatisch eine Quelltext-Dokumentation im HTML-Format generiert werden. Zum Beispiel wird die Java-API-Dokumentation `http://docs.oracle.com/javase/1.5.0/docs/api/` mit Javadoc erzeugt.

Eine kurze Beschreibung von Javadoc finden Sie bei Wikipedia `https://de.wikipedia.org/wiki/Javadoc`. Eine ausführliche Dokumentation und Hinweise zum Schreiben von Javadoc-Kommentaren finden Sie unter `http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html`.

1. Welche Elemente eines Java-Quelltextes können mit Javadoc-Kommentaren dokumentiert werden?
2. Was sind Javadoc-Tags? Erläutern Sie kurz wofür die folgenden Tags genutzt werden: `@author`, `@version`, `@param`, `@return`.

Die integrierte Entwicklungsumgebung Eclipse unterstützt die Arbeit mit Javadoc- Kommentaren. So wird z. B. automatisch die Dokumentation von kommentierten Methoden eingeblendet (Mauszeiger über Methodennamen halten).

Aufgabe 5 (Berechnen von π) 1,5 Punkte

1. Die Kreiszahl π kann auf vielerlei Arten berechnet werden. In dieser Aufgabe betrachten wir ein probabilistisches Verfahren, welches auch in einer Weihnachtsbäckerei Anwendung finden könnte. Als Utensilien genügen ein quadratisches Backblech und eine darauf gestellte kreisförmige Kuchenform, deren Durchmesser der Kantenlänge des Blechs gleicht. Nun werden aus einiger Höhe möglichst zufällig Zuckerstreusel auf das Blech verteilt. Danach zählt (oder wiegt) man die Streusel die auf dem Blech in die Kuchenform gefallen sind. Aus dem Verhältnis zur Zahl der Streusel, die insgesamt auf das Blech gefallen sind kann die Kreiszahl π errechnet werden:

$$\frac{\textit{Anz. Streusel im Kreis}}{\textit{Anz. Streusel im Quadrat}} = \frac{\pi * r^2}{(2*r)^2} \quad \text{und somit} \quad \pi = 4 * \frac{\textit{Anz. Streusel im Kreis}}{\textit{Anz. Streusel im Quadrat}}$$

   Schreiben Sie ein Java-Programm (`Streusel.java`), welches das beschriebene Verfahren zur näherungsweisen Berechnung der Kreiszahl π umsetzt.

   Machen Sie sich vorher darüber Gedanken, wie festgestellt werden kann, ob ein Streusel im Kreis gelandet ist oder nicht. Zum zufälligen Verteilen der Streusel kann die `Math.random(double a)`-Methode verwendet werden.

   Gestalten Sie Ihr Programm so, das insgesamt 3.000.000 Streusel fallen und nach jeweils 300.000 gefallenen Streuseln die bisher ermittelte Näherung für π ausgegeben wird.

2. Das Wallis-Produkt ist eine Darstellung der Kreizahl π als Produkt aus unendlich vielen Faktoren. Dies bedeutet, dass die Folge der Teilprodukte mit wachsender Anzahl von Faktoren gegen π konvergiert. Das Wallis-Produkt kann wie folgt notiert werden.

$$\frac{\pi}{2} = \frac{2}{1} * \frac{2}{3} * \frac{4}{3} * \frac{4}{5} * \frac{6}{5} * \frac{6}{7} * \ldots$$

   Schreiben Sie ein Programm (`Wallis.java`), das das Wallis Produkt für 3.000.000 Faktoren berechnet. Geben Sie dabei nach jeweil 300.000 Faktoren das jeweilige angenäherte Ergebnis für π aus.

3. Schreiben Sie ein Programm (`WallisFaktoren.java`), das nach Eingabe einer Genauigkeit $\epsilon > 0$ berechnet, wie viele Faktoren berücksichtigt werden müssen, damit das Teilprodukt näher als ϵ an `Math.PI` liegt, d. h. der Absolutbetrag der Differenz kleiner als ϵ ist. Geben Sie die Anzahl der benötigten Faktoren an für $\epsilon_1 = 0{,}3$, $\epsilon_2 = 0{,}03$, $\epsilon_3 = 0{,}003$, $\epsilon_4 = 0{,}0003$ und $\epsilon_5 = 0{,}00003$ an.

Aufgabe 6 (Bundestagswahl) 2 Punkte

Programmieren Sie eine Anwendung `Bundestagswahl.java`, welche den simulierten Textmodus (Vorgabe `Console.java`) nutzt, um die Ergebnisse einer fiktiven Bundestagswahl als animiertes Balkendiagramm darzustellen. Orientieren Sie Sich dabei am Beispielvideo aus der Vorgabe.

Der Einfachheit halber gehen wir von fünf Antwortmöglichkeiten aus, deren Diagrammbalken mit den dazugehörigen Farben (5 verschiedene) dargestellt werden. Die Wahlergebnisse für alle Antworten sollen in jedem Programmlauf unterschiedlich sein. Generieren Sie dazu Zufallswerte zwischen 0 und 50 für jede Partei. Beachten Sie, dass die Summe der Stimmen wieder 100 ergeben soll.

Das Balkendiagramm soll horizontal ausgerichtet sein, sodass in einer Zeile zunächst die Antwort

steht. Daneben soll der animierte Diagrammbalken erscheinen, an dessen Ende das Ergebnis der entsprechenden Antwort als Zahl dargestellt wird.

Für die Animation sollen die Diagrammbalken und die dargestellten Werte zunächst bei Null starten und dann alle gleichzeitig so lange anwachsen, bis das zu Programmstart gewürfelte Wahlergebnis erreicht wurde. Die numerische Darstellung soll während der Animation ebenfalls hochgezählt werden.

Nach dem Ende der Animation soll unter dem Diagramm die Antwort mit den meisten Stimmen angezeigt werden.