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 5

Abgabedatum: 21.11.2021

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 **(6 Punkte)** mit diesem Übungsblatt erreichen.

Aufgabe 1 (Prüfziffern)

1 Punkt

Euro-Banknoten haben (eventuell hatten) eine eindeutige Seriennummer, die aus einem führenden Buchstaben, 10 Ziffern und einer Prüfziffer bestehen. Beispiel: Z 6016220022-6 (Trennstrich zur Übersicht hinzugefügt).

Der führende Buchstabe codiert die nationale Zentralbank (NZB), die den Geldschein in Umlauf gebracht hat. Sie wird NZB-Nummer genannt.

Die Prüfziffer berechnet sich wie folgt:

- Der Buchstabe wird durch seine Positionszahl, d. h. seine Stellung im lateinischen Alphabet ersetzt (also für A = 1 und für Z = 26).
- Es wird die Quersumme der Positionszahl und der 10 Ziffern berechnet (im Beispiel 2 + 6 + 6 + 0 + 1 + 6 + 2 + 2 + 0 + 0 + 2 + 2 = 29).
- Die Zahl wird mit Rest durch 9 geteilt (29 / 9 = 3 Rest 2).
- Der Rest wird von 8 subtrahiert. Das Resultat ist die Prüfziffer (8 - 2 = 6). Es sei denn, es kam 0 dabei heraus, dann ist die Prüfziffer 9.

Schreiben Sie ein Java-Program (`Pruefziffer.java`), das für eine Seriennummer die Prüfziffer berechnet. Überlegen Sie sich eine geeignete Art und Weise die Seriennummer inklusive dem führenden Buchstaben einzulesen und zu speichern.

Aufgabe 2 (Arrays (Theorie))

0,5 Punkte

1. Wie wird ein Array deklariert und initialisiert? Geben Sie ein konkretes Beispiel an!
2. Wie kann der Wert an einem bestimmten Index geändert werden? Geben Sie ein konkretes Beispiel an!

3. Wie kann der Wert an einem bestimmten Index ausgelesen werden? Geben Sie ein konkretes Beispiel an!
4. Warum braucht eine Methode mit der folgenden Signatur:

   `void methode(int[] zahlen)`

   keine Rückgabe, wenn Sie in ihrem Rumpf nur das übergebene Array manipuliert, der Aufrufer aber mit dem geänderten Array weiterarbeiten möchte?

Aufgabe 3 (Funktionen über Arrays) 1 Punkt

1. Schreiben Sie eine Anwendung (`Array.java`), welche eine Reihe von Zahlen einliest. Der Nutzer soll vorher eingeben, wie viele Zahlen er eingeben möchte. Dem entsprechend soll ein von der Größe und vom Datentyp her passendes Array angelegt werden. Nach Eingabe aller Zahlen soll die größte und die kleinste ausgegeben werden. Geben Sie des Weiteren alle Ziffern aus, die mindestens einmal in einer der eingegebenen Zahlen enthalten waren.
2. Schreiben Sie ein Java-Programm (`Wuerfel.java`) um einen Würfel zu simulieren. Schreiben Sie eine Anwendung, welche wiederholt eine Zahl zwischen Eins und Sechs würfelt und sich in einem Array merkt, wie oft jede Zahl vorkam, bis eine Zahl eine Million mal gewürfelt wurde. Anschließend sollen in einer Tabelle die Häufigkeiten ausgegeben werden. Als Tabelle wird dabei eine Formatierung von Ausgaben mittels Tabs `\t` verstanden.

   Zur Wiederholung: die Funktion `Math.random(double a)` erzeugt eine Zufallszahl größer oder gleich Null und kleiner Eins.

Aufgabe 4 (Sieb des Eratosthenes) 1 Punkt

Ein einfaches Verfahren zur Berechnung aller Primzahlen unterhalb einer vorgegebenen Schranke $n \in \mathbb{N}$, ist das nach dem griechischen Mathematiker Eratosthenes von Kyrene benannte Sieb des Eratosthenes.

Das Verfahren wird zutreffend ein Siebverfahren genannt, da aus einer Anfangsmenge M = {2, 3, 4, 5, 6, . . . , n} $\subseteq \mathbb{N}$ nach und nach alle Vielfache der Primzahlen $\leq$ n herausgesiebt werden und nur die Primzahlen übrig bleiben.

Dazu ist es nicht nötig die Primzahlen bereits zu kennen. Stattdessen ist nach Voraussetzung nur die kleinste Zahl der Menge M (die Zahl 2) eine Primzahl. Nach dem Aussieben aller Vielfache dieser kleinsten Zahl entsteht eine Menge M′ = {3, 5, 7, 9, 11, . . . , } $\subseteq$ M. Deren kleinste Zahl (die Zahl 3) ist wiederum eine Primzahl, da diese nicht ausgesiebt wurde und daher nicht Vielfaches einer kleineren Zahl (außer der Eins) sein kann. Durch erneutes Aussieben aller Vielfache dieser neuen kleinsten Zahl entsteht eine Menge M″ = {5, 7, 11, 13, 17,. . . } $\subseteq$ M′. Auf diese Weise werden durch wiederholtes Aussieben der Vielfache der jeweils kleinsten Zahl alle Primzahlen $\leq$ n berechnet.

Schreiben Sie ein Programm (`Sieb.java`), das mit dem Sieb des Eratosthenes alle Primzahlen bis zu einer einzugebenden Schranke berechnet und ausgibt. Die Menge M kann durch ein Array von booleschen Werten repräsentiert werden. Dabei ist eine Zahl i genau dann in der Menge enthalten, wenn im Array an Index i der Wert true gespeichert ist.

Aufgabe 5 (Kaminfeuer) 2,5 Punkte

Durch einfache Mittelwertbildung über Farbwerte lässt sich ein schöner Feuereffekt in Java programmieren. Vervollständigen Sie dazu das Programm `Feuer.java` aus der Vorgabe. Fügen Sie Ihren Code zwischen den Kommentarzeilen `// BEGINN IHRES QUELLCODES` und `// ENDE IHRES QUELLCODES` ein.

In der Vorgabe ist bereits ein zweidimensionales Array `colors` angelegt, welches die Pixel für die grafische Ausgabe im `ImageFrame`-Grafikfenster darstellt. Die Pixel können durch einfaches Zuweisen von Farbwerten an die entsprechende Koordinate im Array gesetzt werden. Vordefinierte Farbwerte sind beispielsweise über die Konstanten `Color.RED`, `Color.YELLOW`, `Color.BLACK` oder `Color.WHITE` verfügbar. Die Vorgabe beinhaltet bereits drei auskommentierte Beispiele, wie Pixel gefärbt werden können. Sie können diese einkommentieren und nachvollziehen bzw. experimentieren. Löschen Sie anschließend die Beispiele.

Damit Flammen entstehen, müssen Sie zunächst in jeder Berechnungsrunde (innerhalb der `while`-Schleife) in der ersten Zeile jeweils 8 Pixel gelb, 16 in rot und 8 in schwarz zufällig einfärben. Diese eingefärbten Pixel dienen als „Keimpunkte“ für die Flammen.

Berechnen Sie anschließend neue Farbwerte für alle Pixel ab der zweiten Zeile, indem Sie zunächst die Farbwerte des aktuellen Pixels a, des Pixels links l und rechts davon r sowie dem Pixel darunter d aus dem Pixel-Array auslesen. Der rote Farbanteil des neuen Farbwertes wird aus dem gewichteten Mittelwert der Rot-Komponenten der zuvor ausgelesenen Pixel gebildet: $neu_{rot} = (a_{rot} * w_0 + l_{rot} * w_1 + r_{rot} * w_2 + d_{rot} * w_3)/(w_0 + w_1 + w_2 + w_3)$ gebildet. Für eine dynamischere Flammendarstellung wird der untere Farbwert höher gewichtet, verwenden Sie dafür den Gewichtsvektor $\{1, 1, 1, 3\}$.

Der Rotanteil kann mit der Methode `getRed()` ausgelesen werden. Verfahren Sie für den grünen und blauen Farbanteil genauso (Methoden `getGreen()` und `getBlue()`). Der neue Farbwert kann anschließen mit `Color neueFarbe = new Color(neuRot, neuGruen, neuBlau);` erzeugt werden und muss anschließend in die aktuelle Position im Array gespeichert werden.

Für eine bessere Veranschaulichung können Sie sich das Video Feuer-Beispiel.mp4 aus der Vorlage angucken.