252-0027-00: Einführung in die Programmierung

Übungsblatt 4

Abgabe: 22. Oktober 2019, 10:00

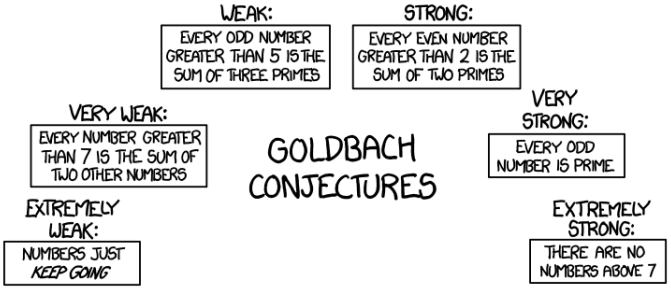
Checken Sie mit Eclipse wie bisher die neue Übungs-Vorlage aus. Beachten Sie, dass Sie mehrere unabhängige Programme im selben Eclipse-Projekt haben werden. Bevor Sie ein Programm starten, achten Sie deshalb darauf, dass Sie die richtige Datei im Package Explorer ausgewählt oder im Editor geöffnet haben. *Vergessen Sie nicht, Ihren Programmcode zu kommentieren!*

Aufgabe 1: Sieb des Eratosthenes

Schreiben Sie ein Programm "Sieb.java", das eine Zahl *limit* einliest und die Anzahl der Primzahlen, die grösser als 1 und kleiner oder gleich dem *limit* sind, ausgibt. Dazu ermitteln Sie in einem ersten Schritt alle Primzahlen, die kleiner oder gleich *limit* sind, und merken sich diese in einem Array. Dieses Teilproblem können Sie mit dem Sieb des Eratosthenes lösen. Danach können Sie die Anzahl der gefundenen Primzahlen anhand dieses Arrays bestimmen.
Beispiel: Für $limit = 13$ sollte Ihr Programm 6 ausgeben (Primzahlen: 2, 3, 5, 7, 11, 13).

Aufgabe 2: Newton-Raphson

Schreiben Sie ein Programm "Newton.java", das eine ganze positive Zahl einliest und eine Approximation der Quadratwurzel nach dem Newton-Raphson Verfahren auf 12 Stellen nach dem Komma berechnet. (Sie sollen selbst entscheiden, welche Methoden Sie definieren.)
Hinweis:



xkcd: Goldbach Conjectures by Randall Munroe (CC BY-NC 2.5)

t	Approximation der Wurzel von c (t und c sind vom Typ `double`)
eps	maximal erlaubter Fehler
$abs(t \cdot t - c) < eps$	falls `true` ist unsere Approximation gut genug, sonst ...
$t' = \frac{c/t + t}{2.0}$	neue Approximation t' ist der Mittelwert von c/t und t

Aufgabe 3: Ohne Sieben (Bonus!)

Achtung: Diese Aufgabe gibt Bonuspunkte (siehe "Leistungskontrolle" im `www.vvz.ethz.ch`). Die Aufgabe muss eigenhändig und alleine gelöst werden. Die Abgabe erfolgt wie gewohnt per Push in Ihres Git-Repository auf dem ETH-Server. Verbindlich ist der letzte Push vor dem Abgabetermin. Bitte lesen Sie zusätzlich die allgemeinen Regeln.

1. Ihre Aufgabe ist eine nicht-negative Zahl $N \geq 0$ in zwei Summanden s_1 und s_2 zu zerlegen, so dass $s_1 \geq s_2$ und $s_1 + s_2 = N$ gilt und die Ziffer 7 weder in s_1 noch in s_2 auftritt (in der Dezimaldarstellung). Zum Beispiel ist für $N = 9743$ eine Zerlegung in $s_1 = 6852$ und $s_2 = 2891$ eine mögliche Lösung.

   Vervollständigen Sie dafür in der Klasse `OhneSieben` die Methode `ohneSieben()`. Die Methode nimmt einen Parameter `zahl`, welcher N entspricht, und gibt einen Array mit Länge 2 zurück, in welchem der Wert beim Index 0 der Zahl s_1 und beim Index 1 der Zahl s_2 entspricht.

2. Committen und pushen Sie Ihre Lösung!

Tipp: Zum Testen können Sie die `main`-Methode verwenden. Wir haben für Sie 3 Tests vorgeschrieben, welche Ergebnisse mit der Methode `richtigesResultat()` prüfen. **Achtung**: Die Methode `richtigesResultat()` überprüft nicht, ob die beiden Summanden die Ziffer 7 nicht enthalten. Wenn Sie wollen, dann können Sie die Methode entsprechend anpassen. Beachten Sie, dass Ihre Lösung mehrere Tests hintereinander bestehen können muss.

Aufgabe 4: Schweizerfahne (GUI)

Obwohl die Konsole für viele Programme als Benutzerschnittstelle ausreicht, ist sie für graphische Anwendungen nur begrenzt geeignet. Java enthält deshalb auch Klassen, mit denen man eine *GUI* (graphical user interface) erstellen kann. Da diese Klassen allerdings nicht für Anfänger geeignet sind, benutzen Sie in diesem Kurs eine einfacher zu verwendende Klasse namens `Window`. Das "U04"-Projekt ist bereits so aufgesetzt, dass Sie diese Klasse in Ihren Programmen verwenden können. Ausserdem befinden sich dort vier Beispiel-Programme, welche die `Window`-Klasse verwenden, und welche Sie als Referenz für die korrekte Verwendung der `Window` Klasse nehmen können. In dieser Übung sollen Sie ein erstes, ganz einfaches Programm mit dieser Klasse schreiben, welches die Schweizerfahne in einem Fenster anzeigt.

1. Erstellen Sie ein neues Programm "RealSwissFlag.java", inklusive `main`-Methode.

2. Erstellen Sie eine Instanz der `Window`-Klasse. Sie machen dies analog zur `Scanner`-Klasse:

   ```
   Window window = new Window("Fahne", 400, 400);
   ```

   Das erste Argument, `"Fahne"`, entspricht dem Titel des Fensters und die beiden anderen Argumente, `400` und `400`, entsprechen der Fenstergrösse.

3. Sie bestimmen den Inhalt des Fensters, indem Sie Zeichnungsbefehle auf das `Window`-Objekt aufrufen. Zeichnungsbefehle bestehen meist aus zwei oder mehr Methodenaufrufen: Zuerst wird die Zeichenfarbe festgelegt und dann wird eine Form gezeichnet. Zum Beispiel:

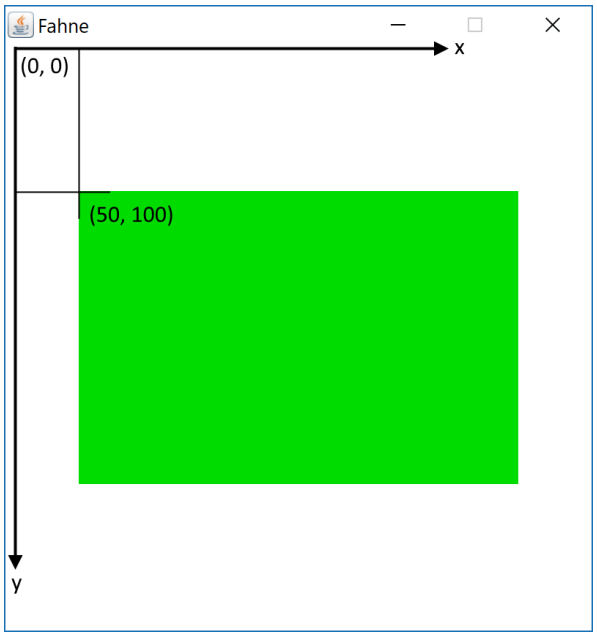
```
window.setColor(0, 220, 0);
window.fillRect(50, 100, 300, 200);
```

Der erste Aufruf setzt die Zeichenfarbe auf einen mittelhellen Grünton und der zweite zeichnet ein ausgefülltes Rechteck mit der linken oberen Ecke bei der Koordinate $(50, 100)$, mit einer Breite von 300 Pixel und einer Höhe von 200 Pixel.

Die Farbe wird als sogenannter **RGB**-Wert angegeben. Das erste Argument entspricht dem **R**ot-Kanal, das zweite dem **G**rün-Kanal und das dritte dem **B**lau-Kanal. Alle Werte sollten zwischen 0 und 255 sein. Beispiele:

`(255,  0,  0)`: rot	`(255, 255,  0)`: gelb	`(  0,  0,  0)`: schwarz
`(  0, 255,  0)`: grün	`(255,  0, 255)`: magenta	`(127, 127, 127)`: grau
`(  0,  0, 255)`: blau	`(  0, 255, 255)`: cyan	`(255, 255, 255)`: weiss

Das Koordinatensystem von GUIs ist ein wenig anders als das, welches Sie von der Mathematik kennen. Statt von unten nach oben verläuft die y-Achse von oben nach unten und der Nullpunkt befindet sich in der linken oberen Ecke des Fensters:



Zeichnen Sie also die Schweizerfahne in das Fenster, indem Sie mehrere Rechtecke in der richtigen Farbe und den Abmessungen zeichnen. Auf Wikipedia finden Sie (teilweise widersprüchliche) Spezifikationen zu Form und Farbe.

4. Um das Fenster anzuzeigen, verwenden Sie als letztes die folgenden beiden Aufrufe:

```
window.open();
window.waitUntilClosed();
```

Der erste Aufruf öffnet das Fenster und der zweite verhindert, dass das Programm sofort wieder beendet und das Fenster dadurch geschlossen wird. Erst wenn der Benutzer das Fenster schliesst, erreicht das Programm das Ende der `main`-Methode und wird beendet.

Optional: Im Allgemeinen ist guter Code ("Good Code") so geschrieben, dass bei kleinen Änderungswünschen möglichst wenige Stellen im Code angepasst werden müssen. Ihr Programm sollte daher möglichst *parametrisiert* geschrieben sein. Ein wichtiger Parameter ist beispielsweise die Grösse der Fahne. Ändern Sie nun Ihr Programm so ab, dass es zuerst von der Konsole die gewünschte Grösse einliest und danach das Fenster und die Fahne in der gewünschten Grösse darstellt.