

Grundlagen der Programmierung

Marcel Lüthi
Andreas Morel-Forster
HS 23

Universität Basel
Fachbereich Informatik

Übung 11

Voraussetzung

- Ein JDK ist installiert.
- Installierte IDE, Visual Studio Code sowie die Plugins für Java und Gradle
- Wenn Sie die Vorlesung verpasst haben, dann empfehlen wir Ihnen die Unterlagen anzuschauen.
- Die Zip-Datei, die auch dieses Übungsblatt enthält, muss entpackt werden. Es enthält die gesamte Übungsumgebung. Schreiben Sie ihre Lösungen in die dafür vorgesehenen Dateien, wie in der jeweiligen Übungsaufgabe angegeben.

Wichtiger Hinweis

- *Achten Sie auf guten Programmierstil.*

Aufgabe 11.1 (Fehlersuche)

Schauen Sie sich die folgenden Programmstücke an, und finden Sie die Programmierfehler, die sich dort eingeschlichen haben (ein Fehlertyp wird nur einmal gezählt). Erläutern sie kurz was das Problem ist, und wie man dies korrigieren könnte. Schreiben Sie Ihre Antworten in das Textdokument `Fehlersuche.txt`, welches Sie im Übungsordner finden.

Eine Routine, um die Position des ersten Vorkommens eines Buchstabens in einem Text zu bestimmen. Im Falle, dass der Buchstabe nicht vorkommt soll -1 zurückgegeben werden. Die gegebene Implementation enthält 2 Fehler.

```
public static int findChar(String s, char c) {  
    for (int i=0; i<s.length(); ++i)  
        if (s.charAt(i) == c)  
            break;  
    return i;  
}
```

Eine Accountklasse, und die dazugehörige Testroutine, welche als Java-Programm ausführbar sein soll. Die gegebene Implementation enthält 6 Fehler.

```
public class Account {  
    private double value;  
    private String name;  
  
    Account(String name, double value) {  
        name = name;  
        value = value;  
    }  
  
    public String toString() {
```

```

        "Account: " + name + " contains " + value + "chf";
    }

    public static void testAccount() {
        Account accounts[];
        accounts[0] = new Account("Petra", 111);
        accounts[1] = new Account("Klaus", 999);

        for (i=0; i <= accounts.length; ++i)
            System.out.print("account " + i + ": ");
            System.out.println(accounts[i]);
    }
}

```

Aufgabe 11.2 (Exceptions, 2 Punkte)

Sie finden im Verzeichnis `src/main/java/list` die Klasse `LinkedList`. Implementieren Sie die Methode `removeFirst`, welche das erste Element der Liste entfernt und zurückgibt. Schreiben Sie eine Klasse `ListException`. Wenn die Methode `removeFirst` auf einer leeren Liste aufgerufen wird, soll diese eine `ListException` mit der Meldung *“on empty list”* ausgegeben.

Implementieren Sie nun die Methode `removeAll`, welche alle Elemente der Liste entfernt, indem mehrmals die Methode `removeFirst` aufgerufen wird. Die Methode `removeAll` darf keine Exception werfen.

Aufgabe 11.3 (Postleitzahlen)

Im Verzeichnis `src/main/java/zipcode` finden Sie die Klasse `ZipCode`. Diese soll eine Datei der Post einlesen, parsen und für einfache Fragen aufbereiten. Die aufzubereitenden Daten finden Sie in der Datei `src/test/resources/Adressdaten.csv`. Die Daten sind im sogenannten CSV Format gespeichert, wobei die einzelnen Felder durch ein Semikolon getrennt sind. Jede Zeile besteht aus dem Kantonskennzeichen, dem Ortsnamen sowie der Postleitzahl.

Für die Implementation sollen Sie die Datenstrukturen `java.util.LinkedList` und `java.util.HashMap` der Java Standardbibliothek verwenden. Bevor Sie die Aufgabe lösen, lesen Sie in der API Dokumentation von Java, was diese Klassen genau machen. Danach implementieren Sie die fehlenden Methoden. Testen Sie ihre Lösung mit den mitgelieferten Tests.

Aufgabe 11.4 (Funktionale Bilder)

In dieser Aufgabe erzeugen Sie Bilder, indem Sie mathematische Funktionen auswerten. Die Idee stammt ursprünglich von Conal Eliot. Unter folgender URL finden Sie eine Galerie mit vielen Beispielen: <http://conal.net/Pan/Gallery/>.

In dieser Aufgabe beschränken wir uns auf die einfachsten Bilder, nämlich einfache Schwarzweissbilder die auf Euklidischen Koordinaten definiert werden.¹ Ein Bild I ist also eine mathematische Funktion (in Ihrer Implementation entspricht dies einer Methode mit Rückgabewert)

$$I : \mathbb{R}^2 \rightarrow \{0, 1\}$$

die jedem Punkt $p = (x, y) \in \mathbb{R}^2$ den Wert $I(p) \in \{0, 1\}$ zuweist. Ein nur weisses Bild ist zum Beispiel durch die Funktion

$$I : (x, y) \mapsto 0$$

dargestellt. Das Bild, welches den Einheitskreis um den Ursprung zeigt, entspricht der Funktion (nicht verlangt in der Aufgabe)

$$I : (x, y) \mapsto \begin{cases} 1 & \text{falls } \sqrt{x^2 + y^2} < 1 \\ 0 & \text{sonst} \end{cases}.$$

¹Conal Elliot beschreibt auch wie man interessante Bilder via Polarkoordinaten erzeugt, wie man Farbe dazu nimmt und wie man Animationen macht.

Der vertikale Streifen in der Abbildung unten links wäre demnach durch die Funktion

$$I : (x, y) \mapsto \begin{cases} 1 & \text{falls } |x| < 0.5 \\ 0 & \text{sonst.} \end{cases}$$

gegeben.

Wir können Variationen von Mustern erzeugen, indem wir eine Koordinatentransformation t definieren und damit das Gebiet, auf welchem unser Bild definiert ist, transformieren. Wir erhalten ein neues Bild I' durch die Komposition vom Bild I mit der Koordinatentransformation t :

$$I'(p) = (I \circ t)(p) = I(t(p)).$$

Eine einfaches Beispiel einer Koordinatentransformation ist eine Translation, die wie folgt definiert ist

$$(x, y) \mapsto (x + t_x, y + t_y)$$

wobei t_x, t_y Parameter sind, die die Grösse der Translation definieren.

Sie finden im Verzeichnis `src/main/java/images` die Klasse `FunctionalImage`, welche diese Idee mit Hilfe von boolschen Funktionen (also Funktionen vom Typ

`Function<Point, Boolean>`) umsetzt. Die Methode `render` wertet die repräsentierte Funktion auf der Domain $[-1, 1] \times [-1, 1]$ aus und erzeugt daraus ein Bild.

- Implementieren Sie die Methode `createStrip` welche einen vertikalen Streifen der Breite 1 in der Mitte des Bildes zeichnet (siehe Bild unten links).
- Implementieren Sie die Methode `compose`, welches eine Funktion (Koordinatentransformation) vom Typ `Function<Point, Point>` entgegennimmt und daraus ein neues Bild durch Komposition dieser Funktionen mit dem Bild erzeugt. Für ein gegebenes Bild I und Koordinatentransformation t soll also $I \circ t$ berechnet werden.
- Implementieren Sie dann die Methode `rotate`, welche einen Parameter θ entgegennimmt und eine Funktion vom Type `Function<Point, Point>` zurückgibt. Die Funktion, die von `rotate(θ)` zurückgegeben wird, soll für jeden Punkt eine Rotation um den Nullpunkt um den Winkel θ (in Radians) ausführt. Nutzen Sie dabei folgende Formel um die Rotation zu implementieren:

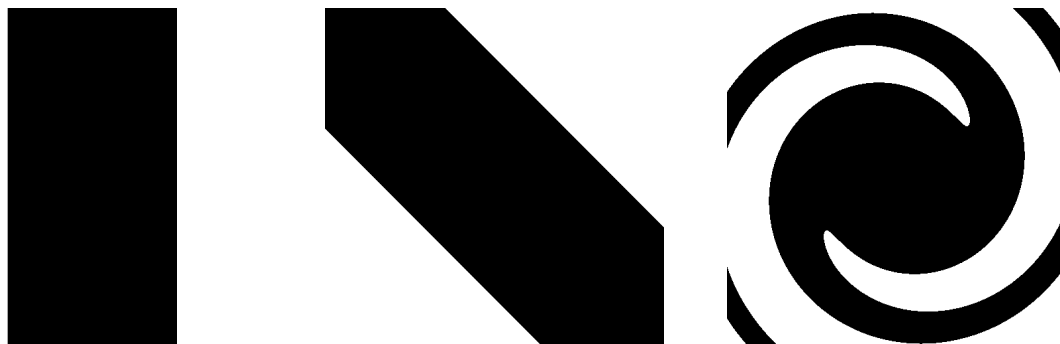
$$(x, y) \mapsto (x \cos(\theta) - y \sin(\theta), y \cos(\theta) + x \sin(\theta))$$

Durch Komposition vom Bild mit `rotate($\pi/4$)` sollten Sie einen um 45° gedrehten Streifen erhalten, wie im Bild unten (Mitte) dargestellt.

- Implementieren Sie die Methode `swirl(r)`, welche einen Punkt nach folgender Formel transformiert:

$$p \mapsto \text{rotate}(\underbrace{\text{dist}(p) \cdot 2 \cdot \pi / r}_{\text{Rotationsparameter}})(p).$$

Dabei bezeichnet $\text{dist}(p)$ die Euklidische Distanz vom Punkt zum Koordinatenursprung und $\text{rotate}(\theta)$ ist die oben implementierte Rotationsmethode und r ist ein Parameter welcher vom Benutzer gewählt werden kann. Wenn Sie nun auch diese Funktion mit $r = 1$ ausführen, sollten Sie ein Bild wie unten rechts erhalten.



Hinweis: Zu dieser Aufgabe gibt es keine automatisierten Tests. Sie erkennen, ob ihr Programm korrekt ist, indem Sie die generierten Bilder anschauen.

Abgabe Erstellen Sie eine Zip-Datei der gesamten Übungsumgebung (also des Verzeichnisses `uebung011`) und laden Sie dieses auf Adam hoch.