Dr. Annette Bieniusa
Mathias Weber, M. Sc.
Peter Zeller, M. Sc.

TU Kaiserslautern
Fachbereich Informatik
AG Softwaretechnik

Übungsblatt 4: Software-Entwicklung 1 (WS 2016/17)

Ausgabe: 14.11.15
Abgabe: 21.11.15

Wir bieten am Freitag (18.11.) von 15:30 bis 16:30 eine weitere **zusätzliche Fragestunde** im Terminalraum 32-411 an. Dort helfen wir Ihnen gerne, wenn Sie Probleme beim Lösen der Aufgaben haben.

Aufgabe 1 Arrays und Prozeduren (13 Punkte)

Schreiben Sie für jede der folgenden Aufgaben eine Prozedur. Erstellen Sie außerdem eine `main`-Prozedur, welche Ihre Prozeduren mit Beispiel-Werten aufruft und das Ergebnis auf der Konsole ausgibt. Verwenden Sie zum Umwandeln von Arrays in Strings die Bibliotheksfunktion `Arrays.toString(ar)`. Fügen Sie dazu in der Java-Datei am Anfang eine Zeile `import java.util.Arrays;` hinzu. Verwenden Sie für diese Aufgabe sonst keine weiteren Bibliotheksfunktionen.

a) Schreiben Sie eine Prozedur `replaceAll`, welche zwei `int` Werte `x` und `y` sowie ein Array von `int`-Werten nimmt. Die Prozedur soll dann alle Vorkommen von `x` im gegebenen Array durch `y` ersetzen.

   Beispiel:

```
int[] ar = {1, 2, 3, 2, 4};
replaceAll(2, 7, ar);
System.out.println(Arrays.toString(ar)); // Gibt "[1, 7, 3, 7, 4]" aus
```

b) Schreiben Sie zwei weitere Prozeduren wie `replaceAll`. Eine Variante `replaceFirst`, die nur das erste Vorkommen im Array ersetzt und eine andere `replaceLast`, die nur das letzte Vorkommen im Array ersetzt.

c) Schreiben Sie eine Prozedur `substAll`, welche wie `replaceAll` funktioniert, jedoch das Eingabe-Array nicht verändert, sondern ein neues Array mit dem Ergebnis zurückliefert.

   Beispiel:

```
int[] ar = {1, 2, 3, 2, 4};
int[] ar2 = substAll(2, 7, ar);
System.out.println(Arrays.toString(ar)); // Gibt "[1, 2, 3, 2, 4]" aus
System.out.println(Arrays.toString(ar2)); // Gibt "[1, 7, 3, 7, 4]" aus
```

d) Schreiben Sie eine Prozedur `onlyEven`, welche ein Array von `int`-Werten nimmt und ein neues Array zurückgibt, in dem sich nur die geraden Zahlen aus dem ursprünglichen Array befinden.

   Beispiel:

```
int[] a1 = {1, 3, 2, 4, 7, 2};
int[] a2 = onlyEven(a1);
System.out.println(Arrays.toString(a1)); // Gibt "[1, 3, 2, 4, 7, 2]" aus
System.out.println(Arrays.toString(a2)); // Gibt "[2, 4, 2]" aus
```

e) Schreiben Sie eine Prozedur `allHaveZero`, welche ein Array von **`int`**-Arrays nimmt und prüft, ob alle Arrays die Zahl 0 enthalten.

Beispiel:

```
int[][] a1 = {{1,2,0}, {0,7}, {7,0,7}};
int[][] a2 = {{1,2,0}, {4,5,6}};
System.out.println(allHaveZero(a1)); // Gibt "true" aus
System.out.println(allHaveZero(a2)); // Gibt "false" aus
```

Aufgabe 2 StdIn und StdOut (6 Punkte)

Laden Sie sich für diese Aufgabe die Bibliotheken `StdIn.java` und `StdOut.java` von der Vorlesungsseite herunter. Diese Bibliotheken unterstützen das Lesen von der Standardeingabe und das Schreiben von Ausgaben. Die Prozeduren, die Sie für diese Übung brauchen, finden Sie im Script (Kapitel 5, Abschnitt 5 “Eingabe und Ausgabe”).

a) Schreiben Sie ein Programm `SqrtTable`, welches **`double`**-Werte von der Standardeingabe liest und nach jeder gelesenen Zahl die Zahl selbst und die Wurzel der Zahl, jeweils durch Komma getrennt in einer Zeile ausgibt. Verwenden Sie hierzu die Prozeduren `StdIn.isEmpty`, `StdIn.readDouble`, `StdOut.println`, und `Math.sqrt`.

b) Benutzen Sie Ihr Programm und das Range Programm vom vorherigen Aufgabenblatt und erstellen Sie eine Datei in der die Zahlen 0-99 jeweils mit ihrer Wurzel aufgelistet sind. Verwenden Sie dazu die Techniken zum “Umleiten von Eingaben und Ausgaben”, wie sie im Script beschrieben sind. Geben Sie diese Datei ab. In der Abnahme sollen Sie demonstrieren können, wie Sie die Datei erstellt haben.

c) Schreiben Sie eine Prozedur `String[] readStrings()`, welche mit `StdIn.readString()` `String`-Werte von der Standardeingabe liest, bis die Standardeingabe leer ist und die Werte in einem Array speichert.

Hinweis: Arrays können in Java nach ihrer Erstellung nicht vergrößert oder verkleinert werden. Legen Sie deshalb zu Beginn der Prozedur ein Array der Größe 5 an und erstellen Sie ein neues Array der doppelten Größe, sobald das aktuelle Array zu klein wird. Kopieren Sie dann am Ende der Prozedur die Werte in ein Array der passenden Größe.

Aufgabe 3 Formatierte Ausgabe (Sinnvoll bearbeiten)

Die Prozedur `StdOut.printf` kann zur formatierten Ausgabe verwendet werden. Als ersten Parameter nimmt die Funktion einen String mit Platzhaltern für Werte. Als weitere Parameter folgen dann die Werte, die für die Platzhalter eingesetzt werden. Platzhalter beginnen mit einem Prozentzeichen und enden mit einem Buchstaben, welcher beschreibt, wie der Wert umgewandelt werden soll. So steht zum Beispiel `%d` für die Dezimaldarstellung eines **`int`**-Wertes, `%f` für die Fließkommadarstellung einer Zahl, `%e` für die Darstellung einer Zahl in wissenschaftlicher Darstellung mit Exponent und `%s` für die String-Darstellung eines Wertes.

Beispiel:

```
StdOut.printf("In %s ist es %d Grad warm.", "Kaiserslautern", 3);
```

Weitere Optionen können zwischen dem Prozentzeichen und dem Buchstaben am Ende angegeben werden. Eine Zahl direkt nach dem Prozentzeichen gibt eine minimale Breite an. Ist ein Wort zu kurz wird es links mit Leerzeichen aufgefüllt, bis die minimale Breite erreicht ist. Wenn vor der Zahl noch ein Minus steht, dann werden die Leerzeichen rechts angehängt.

Nach der optionalen minimalen Breite kann auch noch die Präzision festgelegt werden. Dies geschieht nach einem Punkt durch eine weitere Zahl, welche zum Beispiel die Anzahl der Nachkommastellen festlegt.

Es gibt noch einige weitere Funktionen für die formatierte Ausgabe `printf`. Die offizielle Dokumentation finden Sie unter:
http://docs.oracle.com/javase/8/docs/api/java/util/Formatter.html#syntax.

a) Was wird für die folgenden Anweisungen ausgegeben? Geben Sie für diese Teilaufgabe eine Text-Datei `ausgabe.txt` ab, welche nur die Ausgabe enthält.

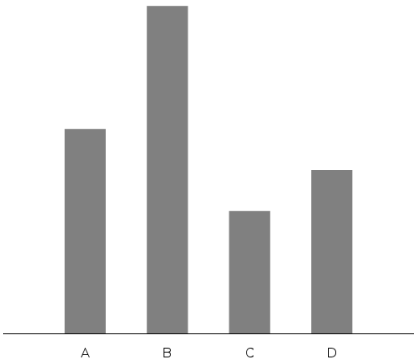
```
StdOut.printf("<%10d>%n",    1234);
StdOut.printf("<%-10d>%n",   1234);
StdOut.printf("<%10.3f>%n", 1234.567891011121314);
StdOut.printf("<%.5f>%n",    1234.567891011121314);
StdOut.printf("<%14.6e>%n", 1234.567891011121314);
StdOut.printf("<%10.5s>%n", "EinsZweiDreiVier");
```

b) Schreiben Sie mit Hilfe von `printf` ein Programm `Table`, welches einen **`int`**-Wert *N* als Programm-Parameter nimmt und **`double`**-Werte von der Standardeingabe liest. Als Ausgabe soll das Programm eine Tabelle der Werte mit *N* Spalten erzeugen. Dabei sollen die Spalten mit "|" voneinander getrennt sein, jede Spalte soll mindestens 12 Zeichen breit sein und die Zahlen sollen alle 2 Nachkommastellen haben, so dass die Ausgabe zum Beispiel wie folgt aussieht:

```
java Range 1 21 1 | java Table 5
       1.00 |        2.00 |        3.00 |        4.00 |        5.00
       6.00 |        7.00 |        8.00 |        9.00 |       10.00
      11.00 |       12.00 |       13.00 |       14.00 |       15.00
      16.00 |       17.00 |       18.00 |       19.00 |       20.00
```

Aufgabe 4 StdDraw (Sinnvoll bearbeiten)

Geben Sie für diese Aufgabe eine einzelne Java-Datei `BarChart.java` ab. Verwenden Sie wenn möglich Prozeduren aus vorherigen Aufgabenteilen um Redundanz zu vermeiden.
Hinweis zur Bearbeitung: Es handelt sich bei dieser Aufgabe um ein etwas größeres Programm. Warten Sie nicht bis zum Ende um Ihr Programm zu testen. Sie sollten einzelne Prozeduren direkt nach dem Schreiben durch Aufrufe in der `main`-Prozedur testen.



Laden Sie sich für diese Aufgabe die Bibliothek `StdDraw.java` von der Vorlesungs-Seite herunter. Diese Bibliothek bietet Prozeduren zum Zeichnen von Bildern.

Das Zeichnen von Bildern basiert auf einem Koordinaten-System, wobei die Werte der x-Achse und y-Achse die Werte 0 bis 1 annehmen können. Der Punkt (0,0) ist die Ecke links unten und der Punkt (1,1) ist die Ecke rechts oben auf der Zeichenfläche.

Für diese Aufgabe sollen Sie die folgenden Prozeduren verwenden:

```
void text(double x, double y, String text)
void line(double x0, double y0, double x1, double y1)
void filledRectangle(double x, double y, double halfWidth, double halfHeight)
```

Um ein Gefühl für das Koordinaten-System zu bekommen, können Sie zuerst das folgende Beispiel verwenden und mit den Werten experimentieren:

```
public class GraphBeispiel {
    public static void main(String[] args) {
        StdDraw.text(0.1, 0.1, "0.1, 0.1");
        StdDraw.text(0.1, 0.9, "0.1, 0.9");
        StdDraw.text(0.9, 0.1, "0.9, 0.1");
        StdDraw.text(0.9, 0.9, "0.9, 0.9");
        StdDraw.line(0.3, 0.8, 0.9, 0.3);
        StdDraw.filledRectangle(0.4, 0.3, 0.3, 0.1);
    }
}
```

Legen Sie dazu die heruntergeladene Datei `StdDraw.java` in den gleichen Ordner und übersetzen Sie dann das Programm mit dem Java-Übersetzer. Anschließend können Sie das Programm ausführen und es sollte ein Fenster mit den gezeichneten Elementen angezeigt werden.

a) Ändern Sie das Beispiel-Programm so ab, dass ein Balken-Diagramm gezeichnet wird, das etwa wie das oben gezeigte aussieht.

   Die Lösung zu Teilaufgabe a) müssen Sie nicht abgeben. Geben Sie am Ende nur eine Datei mit dem kompletten Programm ab, wie es in Aufgabe d) gefordert ist.

b) Schreiben Sie eine Prozedur `void drawBar(double x, double width, String text, double height)`, welche einen einzelnen Balken des Balken-Diagramms zeichnet. Dabei ist der Parameter `text` der Text, der unter dem Balken angezeigt wird, `x` ist die x-Koordinate für die Mitte des Balkens und `width` und `height` geben die Breite und Höhe des Balken an. Schreiben Sie Ihr Programm so um, dass diese Prozedur zum Zeichnen des Diagramms verwendet wird.

c) Schreiben Sie eine Prozedur `void drawBars(String[] labels, double[] heights, double width, double gap)`, welche ein Array von String-Werten und ein Array von `double`-Werten nimmt, die jeweils die Beschriftung und Höhe eines Balken angeben. Außerdem nimmt die Prozedur die Breite der Balken `width` und den Abstand zwischen den einzelnen Balken `gap`. Sie können annehmen, dass es für jede Höhe ein Label gibt.

   Verwenden Sie zur Implementierung die Prozedur `drawBar`.

d) Schreiben Sie eine `main`-Prozedur für Ihr Programm.

   Das Programm soll 2 Programm-Parameter nehmen:

   1. Die Breite der Balken
   2. Der Abstand zwischen den Balken

   Die Werte und Beschriftungen für das Diagramm sollen von der Standardeingabe gelesen werden (bis die Eingabe leer ist). Dabei soll immer zuerst ein `String` gelesen werden, der die Beschriftung angibt und dann ein dazugehöriger `double`-Wert.

   Das gezeichnete Diagramm soll entsprechend den gegebenen Parametern die Prozedur `drawBars` verwenden, um ein Balkendiagramm zu zeichnen.

   Hinweis: Zum Einlesen der Daten können Sie die Prozedur `readStrings` aus Aufgabe 2 verwenden. Zum Testen können Sie sich die Datei `wahlen.txt` von der Vorlesungsseite laden und Ihr Programm mit dem folgenden Befehl starten:

   ```
   java BarChart 0.1 0.03 < wahlen.txt
   ```

e) (Zusatzaufgabe) Verbessern Sie Ihr Programm so, dass die Breite der Balken und der Abstand zwischen den Balken automatisch sinnvoll berechnet werden, wenn sie nicht als Programm-Parameter angegeben werden.

f) (Zusatzaufgabe) Schreiben Sie eine Prozedur `void normalize(double[] values, double max)`, welche ein Array von `double`-Werten nimmt und alle Werte im Array proportional skaliert, so dass danach der größte Wert im Array gleich dem Wert `max` ist, welcher auch als Parameter an die Prozedur übergeben wird. Sie können annehmen, dass das Array nicht leer ist und das alle Werte im Array echt positiv sind.

   Beispiel: Wenn die Eingaben 30, 50, und 20 sind, dann sollen nach dem Normalisieren mit $max = 10$ die Werte 6, 10, und 4 heraus kommen.

   Verwenden Sie dann diese Prozedur, um die Eingaben vor dem Zeichnen auf max=0.8 zu normalisieren, so dass diese gut auf die Zeichenfläche passen.