Hans Joachim Pflug IT-Center, RWTH Aachen; FH Aachen, Campus Jülich

Übungsblatt 1

Aufgabe 1.1: Bearbeitungsregeln

Lesen Sie sich die folgenden Regeln zur Bearbeitung der Aufgaben durch:

- **Kurzaufgaben** werden auf Papier angefertigt. Kurzaufgaben müssen *nur von Übungsgruppe 1* angefertigt werden.
- **Aufgaben** werden am Computer gelöst und sind von allen Übungsgruppen anzufertigen. **Dabei sind immer alle Aufgaben zu testen!**
- **Fragen** sind auf Papier zu beantworten. Sie sind von allen Übungsgruppen zu bearbeiten.
- **Zusatzaufgaben** werden am Computer gelöst. Der Leiter der Übungsgruppe bestimmt, ob und in welchem Umfang Zusatzaufgaben zu lösen sind.

Aufgabe 1.2: Eclipse

Machen Sie sich mit **Eclipse** vertraut, also unserer Entwicklungsumgebung zur Programmierung mit Java: Befolgen Sie die Schritte aus den Kapiteln 1-4 der Eclipse-Anleitung, die Sie separat erhalten haben. Machen Sie dabei folgende Angaben:

- Fenster *„Select a Workspace“*:

  Wählen Sie als Workspace z.B. `C:\Java-Vorkurs`.

- Fenster *„Create a Java Project“*

  Wählen Sie als *Project name* `Blatt1`. Legen Sie für jedes Aufgabenblatt ein Projekt an.

- Fenster *„Java Class“*

  Geben Sie als Name `HelloWorld` an. Ignorieren Sie die Warnung *The use of the default package is discouraged*. Mit Packages werden wir uns erst am Ende des Semesters beschäftigen.

- Geben Sie als Programm ein:

Listing 1: Hello World

```
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello World");
  }
}
```

Das Programm sollte die Zeile *Hello World* im Console-Fenster ausgeben.

Aufgabe 1.3: Dokumentation

Eine Art „Befehlsliste“ zu Java ist die *Java API*. API steht dabei für „Application Programming Interface“, was im Deutschen mit „Programmierschnittstelle“ übersetzt wird. Die Java API ist in html geschrieben. Falls Sie die Java-API noch nicht auf Ihrem Computer gespeichert haben, wenden Sie sich bitte an einen Betreuer. Dieser wird Ihnen dann die API auf einem USB Stick zur Verfügung stellen. Lassen Sie sich im Unterordner `docs\api` die Seite `index.html` anzeigen und wählen Sie im linken unteren Fenster die Klasse *Math*. Im rechten Fenster sehen Sie alle mathematischen Funktionen, die Sie benutzen können. Die Syntax ist z.B.:

```
double x = Math.PI;
double d = Math.abs(x);
```

Übungsblatt 2

Kurzaufgabe 2.1:

Deklarieren Sie eine Integer-Variable *menge*. Weisen Sie Ihr den Wert 2 zu.

Kurzaufgabe 2.2:

Geben Sie mit einer Zeile den Wert von $\frac{3}{17}$ auf dem Bildschirm aus.

Kurzaufgabe 2.3:

Weisen Sie der Variablen *bruch* den Wert $\frac{1}{7}$ zu und geben Sie den Wert auf dem Bildschirm aus.

Kurzaufgabe 2.4:

Weisen Sie der Variablen *pi* den Wert π (Math.PI) zu. Geben Sie den Satz „Pi hat den Wert“, gefolgt vom Wert der Variablen aus.

Kurzaufgabe 2.5:

Berechnen Sie die Wurzel von 2 (Math.sqrt). Weisen Sie das Ergebnis a) einer Double- Variablen und b) einer Integer-Variablen zu.

Kurzaufgabe 2.6:

Erzeugen Sie eine double-Variable *d* mit dem Wert 4,3. Wandeln Sie *d* in einen Integer-Wert. Wandeln Sie den Wert in einen double-Wert zurück. An welcher Stelle brauchen Sie einen expliziten Cast?

Kurzaufgabe 2.7:

Lesen Sie einen String von der Tastatur ein und weisen Sie den String einer Variablen `s` zu.

Kurzaufgabe 2.8:

Lesen Sie eine Integer-Zahl von der Tastatur ein und weisen Sie die Zahl der Variablen `a` zu.

Kurzaufgabe 2.9:

Lesen Sie eine Double-Zahl von der Tastatur ein und weisen Sie die Zahl der Variablen `d` zu.

Kurzaufgabe 2.10:

Vertauschen Sie den Wert zweier Variablen *a* und *b*. Hinweis: Sie benötigen eine Hilfsvariable.

Aufgabe 2.1: Reziproke Zahl

Lesen Sie mit JOptionPane eine Zahl ein und geben Sie eine Zeile der folgenden Form aus: (Hinweis: Reziproke = Kehrwert)

```
Zahl: 5, Reziproke Zahl: 0.2
```

Aufgabe 2.2: Rechenoperationen

Schreiben Sie ein Programm, das eine double-Zahl `x` einliest und die folgenden Werte auf dem Bildschirm ausgibt:
1. den Absolutwert von x, 2. den natürlichen Logarithmus von x, 3. den Sinus Hyperbolikus von x. Benutzen Sie dazu die Java-API.

Aufgabe 2.3: Temperaturumrechnung

Schreiben Sie eine Funktion

```
public static double getFahrenheit(double celsius)
```

Die Funktion erhält eine Temperatur in Grad Celsius. Sie rechnet die Temperatur in Fahrenheit um und gibt das Ergebnis zurück. Es gilt:
$temp[^oC] = (temp[^oF] - 32) \cdot \frac{5}{9}$

Aufgabe 2.4: Punkt-Abstand

Schreiben Sie eine Funktion

```
public static double getDistance(double x1, double y1, double x2, double y2)
```

Die Funktion soll den Abstand zweier Punkte $P_1 = (x_1, y_1), P_2 = (x_2, y_2)$ berechnen und zurückgeben.
(Wurzelberechnung in Java: `Math.sqrt(4); // ergibt 2`)

Abstand $(P_1, P_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

Aufgabe 2.5: Zufallszahl

Nutzen Sie die Methode `Math.random()`. Lesen Sie sich die Beschreibung in der API durch. Schreiben Sie die beiden Funktionen:

```
public static double getDoubleRandom(double min, double max)
```

`getDoubleRandom` gibt eine `double`-Zufallszahl zwischen min (einschließlich) und max (ausschließlich) zurück.

```
public static int getIntRandom(int min, int max)
```

`getIntRandom` gibt eine `int`-Zufallszahl zwischen min (einschließlich) und max (ausschließlich) zurück.

Hans Joachim Pflug IT-Center, RWTH Aachen; FH Aachen, Campus Jülich

Übungsblatt 3

Kurzaufgabe 3.1:

Gegeben sei eine boolean-Variable *b1*. Erstellen Sie eine boolean-Variable *b2*, die nicht-*b1* enthält.

Kurzaufgabe 3.2:

Schreiben Sie eine if-Anweisung, die `"ja"` auf den Bildschirm schreibt, falls die Integer-Variable x den Wert 5 oder 37 hat.

Kurzaufgabe 3.3:

Schreiben Sie eine if-else-Anweisung. Falls die Integer-Variable *x* kleiner als 1 ist, soll *x* auf dem Bildschirm ausgegeben werden, ansonsten $x + 1$.

Kurzaufgabe 3.4:

Gegeben sei eine Integer-Variable *a*. Belegen Sie eine boolean-Variable *b* mit true, falls *a*=5 ist und sonst mit false. Benutzen Sie dazu einmal eine if-Anweisung. Lösen Sie das Problem auch ohne if-Anweisung.

Kurzaufgabe 3.5:

Schreiben Sie das Gerüst für eine switch-Anweisung. Der Selektor ist die Integer-Variable *x*. *x* kann die Werte 1, 2 oder 3 annehmen.

Aufgabe 3.1: Temperatursensor

Schreiben Sie eine Funktion

```
public static String getTemperatureText(double temp)
```

Geben Sie den String `"kalt"` zurück, falls $temp \leq 10$ ist, den String `"lauwarm"`, falls $10 < temp \leq 25$, `"warm"`, falls $25 < temp \leq 40$ und `"heiss"`, falls $40 < temp$.

Aufgabe 3.2: Zubringerbus

Zwischen zwei Terminals eines Flughafens verkehrt ein Zubringerbus, der jeweils zur halben und zur vollen Stunde abfährt. Schreiben Sie eine Funktion

```
public static int getWaitingTime(int h, int min)
```

die die aktuelle Uhrzeit (Stunden/Minuten) erhält und berechnet, wie viele Minuten es bis zur Abfahrt des nächsten Busses noch dauert. Hinweis: Wenn der Bus genau zur eingegebenen Uhrzeit abfährt, so soll 0 zurückgegeben werden. Es muss niemals 30 Minuten gewartet werden.

Aufgabe 3.3: Lineare Gleichung

Schreiben Sie eine Funktion

```
public static double solveLinearEquation(double a, double b)
```

Die Funktion soll die Gleichung $a \cdot x + b = 0$ lösen und die Lösung zurückgeben. Denken Sie daran, dass sowohl a als auch b den Wert 0 annehmen können. Falls die Gleichung nicht lösbar ist oder falls es unendlich viele Lösungen gibt, soll eine Fehlermeldung ausgegeben werden. Testen Sie dazu die Wirkung der folgenden Programmzeile:

```
throw new ArithmeticException("Gleichung nicht eindeutig loesbar!");
```

In der Fachsprache heißt das: Diese Zeile wirft eine Exception (in diesem Fall eine ArithmeticException). Eine genaue Erklärung folgt später in der Vorlesung.

Aufgabe 3.4: Bestellung

Eine Elektrofirma erhebt für Bestellungen unter 100,- Euro einen Porto- und Verpackungsanteil von 5,50 Euro, von 100,- bis 200,- einen Betrag von 3,- Euro, ab 200,- Euro werden keine Porto- und Verpackungskosten berechnet. Schreiben Sie ein Funktion

```
public static int getPostage(int order)
```

die den Wert der Bestellung (in Cent) erhält und die Porto- und Verpackungskosten (ebenfalls in Cent) zurückgibt.

Aufgabe 3.5: Monatsname

Schreiben Sie eine Funktion

```
public static String getNameOfMonth(int nr)
```

Die Funktion erhält eine Zahl zwischen 1 und 12 und gibt die entsprechenden Monatsnamen „Januar“ bis „Dezember“ zurück. Werfen Sie eine ArithmeticException, wenn die Zahl nicht zwischen 1 und 12 liegt.

Aufgaben zur Veranstaltung
Java-Blockkurs, WS 2020/21

Hans Joachim Pflug IT-Center, RWTH Aachen; FH Aachen, Campus Jülich

Übungsblatt 4

Kurzaufgabe 4.1:

Schreiben Sie eine for-Schleife, die die Zahlen 1 bis 10 auf dem Bildschirm ausgibt.

Kurzaufgabe 4.2:

Schreiben Sie eine for-Schleife, die die Wurzel der Zahlen 2 bis 30 auf dem Bildschirm ausgibt.

Kurzaufgabe 4.3:

Suchen Sie mit einer while-Schleife die kleinste positive Integer-Zahl *n*, deren Quadrat größer ist als 12000.

Aufgabe 4.1: Fakultät

Schreiben Sie eine Funktion

```
public static int getFactorial(int x)
```

die die Fakultät der Zahl `x` zurückgibt. Werfen Sie eine ArithmeticException, falls `x` kleiner als 1 ist. Hinweis: $x!$(x Fakultät) ist $1 \cdot 2 \cdot 3 \cdot \ldots \cdot x$.

Aufgabe 4.2: Geburtstage

Die Wahrscheinlichkeit, dass zwei Menschen in einer Gruppe von n Menschen am gleichen Tag Geburtstag haben, beträgt

$$p(n) = 1 - \frac{365}{365} \cdot \frac{364}{365} \cdot \frac{363}{365} \cdot \ldots \cdot \frac{365 - n + 1}{365}$$

Schreiben Sie eine Funktion

```
public static double doubleBirthday(int size)
```

die die Wahrscheinlichkeit zurückgibt, dass in einer Gruppe von `size` Personen zwei Personen am gleichen Tag Geburtstag haben. Werfen Sie eine ArithmeticException, falls `size` kleiner als 1 ist.

Aufgabe 4.3: Pi

Schreiben Sie eine Funktion

```
public static double getPi(int n)
```

Die Funktion berechnet den Wert π näherungsweise und benutzt dafür die Reihenentwicklung:

$$\pi = 4(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13} - \ldots)$$

Das Programm berücksichtig dabei nur die ersten `n` Reihenglieder. Werfen Sie eine ArithmeticException, falls `n` kleiner als 1 ist.

Aufgabe 4.4: Summe

Schreiben Sie eine Funktion

```
public static int getNextPartialSum(int v)
```

Die Summe

$$1 + 2 + 3 + 4 + \ldots + n$$

wird ab einem bestimmten Wert $n = n_0$ größer (oder gleich) werden als der Übergabeparameter `v`. Geben Sie den Wert

$$1 + 2 + 3 + 4 + \ldots + n_0$$

für diesen Wert zurück. Werfen Sie eine ArithmeticException, falls `v` kleiner als 1 ist.

Aufgabe 4.5: Summe und Durchschnitt

Schreiben Sie ein Programm mit der folgenden Funktionalität: In einer Schleife kann der Anwender Zahlen eingeben. Wenn der Anwender die Zahl 0 eingibt, wird die Schleife abgebrochen und die Gesamtsumme, sowie der Durchschnitt der bisher eingegebenen Zahlen auf dem Bildschirm ausgegeben.

Aufgabe 4.6: ASCII-Werte

Schreiben Sie ein Programm, das einen ASCII-Code (eine Integer-Zahl zwischen 0 und 127) einliest und das entsprechende ASCII-Zeichen auf dem Bildschirm ausgibt. Bei einer 0 soll das Programm stoppen.

Aufgabe 4.7: Fibonacci-Zahlen

Die Fibonacci-Folge besteht aus einer Folge von Zahlen, in der jede Zahl die Summe der beiden vorangehenden ist, z.B.:

```
1 1 2 3 5 8 13 21 34 55
```

Schreiben Sie eine Funktion

```
public static int getFibonacciNumber(int n)
```

die das n-te Element der Fibonacci-Folge zurückgibt. Werfen Sie eine ArithmeticException, falls `n` kleiner als 1 ist.

Übungsblatt 5

Frage 5.1:

Folgender Code sei gegeben:

```
char c = 'a';
int i = c;
```

Welchen Wert hat jetzt `i`? Welchen Wert hat `i` bei `c='0'`?

Frage 5.2:

Was erwarten Sie als das Ergebnis der folgenden Zeile?

```
int x = Integer.parseInt("Hallo");
```

Frage 5.3:

Was ist das Ergebnis von

a) `5+7+""` b) `""+5+7` c) `""+(5+7)`

Frage 5.4:

Character-Variablen Eine char-Variable *c* habe den Wert `'a'`. Notieren Sie das Ergebnis der folgenden Zuweisungen, falls es ein Ergebnis gibt:

```
int i = c;
int j = c+1;
char d = c+1;
char e = (char) c+1;
char f = (char) (c+1);
```

Frage 5.5:

Wie würde die nachfolgende for-Schleife als while-Schleife aussehen?

```
int x = 5;
for (int i=0; i<x; i++) {
  System.out.println(i);
}
```

Frage 5.6:

Welcher Operator invertiert eine `boolean`-Variable?

Frage 5.7:

Was ist die Signatur einer Funktion?

Frage 5.8:

Welche Datentypen können in einer `switch`-Anweisung stehen?

Frage 5.9:

Wie lauten die printf-Formatierungszeichen für

a) Integer-Variablen

b) Double-Variablen

c) String-Variablen

d) Integer-Variablen (Mindestbreite 5 Zeichen, linksbündig)

e) String-Variablen (Mindestbreite 20 Zeichen, rechtsbündig)

Frage 5.10:

Wie wandelt man einen String in eine int-Variable um?

Frage 5.11:

Wie oft wird die folgende Schleife durchlaufen?

```
for (int i=0; i>10; i++) {
  System.out.println(i);
}
```

Frage 5.12:

Was bedeutet die Anweisung

```
for (;;)
```

Übungsblatt 6

Zusatzaufgabe 6.1: Größter gemeinsamer Teiler

Schreiben Sie eine Funktion

```
public static int gcd(int a, int b)
```

die den größten gemeinsamen Teiler zweier Zahlen `a` und `b` nach dem Euklidischen Algorithmus berechnet und ihn zurückgibt. Der Euklidische Algorithmus funktioniert wie folgt:

```
solange b != 0
    wenn a > b
        dann a = a - b
        sonst b = b - a
```

Der größte gemeinsame Teiler steht nun in der Variablen `a`. *gcd* steht übrigens für „greatest common divisor“. Werfen Sie eine ArithmeticException, falls `a` oder `b` kleiner als 1 sind.

Zusatzaufgabe 6.2: Zahlendreieck

Schreiben Sie ein Programm, das mit *for*-Anweisungen und *print*- bzw. *println*-Anweisungen das folgende Dreieck auf dem Bildschirm ausgibt. Geben Sie die benötigte Zeilenanzahl (hier fünf) durch eine Variable mit festem Wert vor:

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

Passen Sie das Programm zur Ausgabe des Dreiecks so an, dass die Zahlen, wie unten dargestellt, zentriert werden:

```
    1
   2 2
  3 3 3
 4 4 4 4
5 5 5 5 5
```

Zusatzaufgabe 6.3: Kreis

Schreiben Sie eine Funktion

```
public static void printCircle(int radius)
```

die einen (angenäherten) Kreis mit dem übergebenen Radius, wie im Beispiel gezeigt, ausgibt.

Beispiel: Kreis mit Radius 5. Bei höheren Radien ist die Annäherung an die Kreisform deutlich besser.

```
   #####
  #     #
 #       #
#         #
#         #
#         #
#         #
#         #
 #       #
  #     #
   #####
```

Hinweis: Gegeben sei ein Kreis mit Mittelpunkt (m_x, m_y). Der Abstand eines Punktes (p_x, p_y) vom Mittelpunkt ist dann $a = \sqrt{(m_x - p_x)^2 + (m_y - p_y)^2}$. Der Kreis kann durch die unterschiedliche Zeichenhöhe und -breite zu einer Ellipse verzerrt sein.

Übungsblatt 7

Die folgenden Aufgaben beziehen sich auf die Klasse `Bruch`, deren API Sie im Skript auf Seite 197 finden und die Sie von Ihrem Übungsbetreuer erhalten.

Kurzaufgabe 7.1:

Erzeugen Sie ein `Bruch`-Objekt mit dem Wert `1/5` und geben Sie ihm den Namen `b`.

Kurzaufgabe 7.2:

Geben Sie dem Objekt `b` einen zweiten Alias-Namen `alias_c`.

Kurzaufgabe 7.3:

Erzeugen Sie einen Klon von `b` und geben Sie ihm den Namen `d`.

Kurzaufgabe 7.4:

Invertieren Sie `b`. Welcher Befehl bewirkt dies? Welche Werte haben jetzt `b`, `alias_c` und `d`?

Aufgabe 7.1: Bruch

a) Schreiben Sie ein Programm, das Zähler und Nenner eines Bruchs von der Tastatur einliest und daraus ein `Bruch`-Objekt erzeugt.

b) Erzeugen Sie ein zweites `Bruch`-Objekt, dass genau $\frac{1}{10}$ größer als das erste Objekt ist. Überprüfen Sie, ob der Nenner des zweiten Bruchs größer, kleiner oder gleich dem Nenner des ersten Bruchs ist. Geben Sie Ihre Resultate auf dem Bildschirm aus. Finden Sie für alle 3 Fälle ein Beispiel.

Aufgabe 7.2: Aliasse und Klone

Erzeugen Sie ein `Bruch`-Objekt mit dem Wert `1/3`. Erzeugen Sie einen Alias und einen Klon dieses Objektes. Vergleichen Sie die Objekte untereinander mit == und `equals`. Welche Ergebnisse erhalten Sie?

Aufgabe 7.3: Zufallszahlen

Testen Sie die Klasse `Random` im Paket `java.util`:

a) Erstellen Sie ein `Random`-Objekt. Ziehen Sie die API zu Hilfe.

b) Geben Sie mit Hilfe dieses Objekts 100 Zufallszahlen zwischen 0 und 100 (einschließlich) aus.

Aufgabe 7.4: Eingabe im Ausgabefenster

Schreiben Sie ein Programm, dass es ermöglicht, im Ausgabefenster von Eclipse Daten einzugeben. Benutzen Sie die Klasse `Scanner` im Paket `java.util`:

a) Erzeugen Sie ein Scanner-Objekt mit der Zeile

```
Scanner sc = new Scanner(System.in)
```

b) Lesen Sie mit `nextLine` eine Zeile von der Tastatur ein und geben Sie sie anschließend wieder aus.

Aufgabe 7.5: Distanz zwischen zwei Punkten

Schreiben Sie ein Programm, dass zwei Objekte der Klasse `Point` (Paket `java.awt`) erzeugt und testet.

a) Lesen Sie die Koordinaten zweier Punkte von der Tastatur ein und erzeugen Sie daraus zwei `Point`-Objekte.

b) Verschieben Sie beide Objekte um (10/10) und geben Sie die neuen Koordinaten aus.

c) Geben Sie die Distanz zwischen beiden Punkten aus (Methode `distance`).

Aufgaben zur Veranstaltung
Java-Blockkurs, WS 2020/21

Hans Joachim Pflug IT-Center, RWTH Aachen; FH Aachen, Campus Jülich

Übungsblatt 8

Kurzaufgabe 8.1:

Erzeugen Sie ein eindimensionales Integer-Feld mit den Werten 6,9,2,1 und 3. Verwenden Sie 2 Methoden: 1. Belegen der Elemente gleich bei der Initialisierung. 2. Belegen der Elemente nach der Initialisierung.

Kurzaufgabe 8.2:

Erzeugen Sie ein 10 Elemente großes double-Feld. Setzen Sie die Feldelemente auf Math.sqrt(i) für $i = 0, \ldots, 9$

Kurzaufgabe 8.3:

Erzeugen Sie ein 10×10 großes Integer-Feld. Setzen Sie das Element (4,5) auf den Wert 7.

Kurzaufgabe 8.4:

Gegeben ist ein zweidimensionales double-Feld `d`. Stellen Sie fest, ob das Feld groß genug ist, sodass das Element (Zeile 5/Spalte 5) existiert.

Aufgabe 8.1: Feld erzeugen

Schreiben Sie eine Funktion

```
public static double[][] getDoubleArray(int x, int y)
```

die ein double-Feld der Dimension $x \times y$ zurückgibt.

Aufgabe 8.2: Minimum und Minimum-Index

Schreiben Sie zwei Funktionen

```
public static double getMinimum(double[] a)
public static int getMinimumIndex(double[] a)
```

die das Minimum bzw. den Index des Minimums von *a* zurückgeben.

Aufgabe 8.3: Test auf quadratische Form

Schreiben Sie eine Funktion

```
public static boolean isSquare(int[][] test)
```

die feststellt, ob `test` quadratisch ist.

Aufgabe 8.4: Monatsname 2

Schreiben Sie eine Funktion

```
public static String getNameOfMonth(int no)
```

Die Funktion erhält eine Zahl zwischen 1 und 12 und gibt die entsprechenden Monatsnamen „Januar“ bis „Dezember“ zurück. Vereinfachen Sie Ihre Lösung aus Aufgabe 3.5, indem Sie ein Feld statt eines if- oder switch-Konstrukts verwenden.

Aufgabe 8.5: Verkettung von Arrays

Schreiben Sie ein Funktion

```
public static int[] concat(int[] x1, int[] x2)
```

die die beiden *int*-Arrays aneinanderhängt. Benutzen Sie dabei die Methode *System.arraycopy*. Das Resultat wird zurückgegeben.

Aufgabe 8.6: Indexsumme

Schreiben Sie eine Funktion

```
public static double[][] getIndexSumArray(int a, int b)
```

Diese Funktion gibt ein Feld der Größe $a \times b$ zurück, deren Element nach dem folgenden Schema gebildet wird:

$$a_{ij} = i + j + 1$$

Aufgabe 8.7: Spur

Berechnen Sie die Spur der Matrix aus Aufgabe 8.6 nach der Formel

$$\text{spur}(A) = \sum_{i=1}^{n} a_{ii}$$

(es werden also alle Elemente auf der Diagonalen von links oben nach rechts unten addiert). Schreiben Sie dazu eine Funktion

```
public static double getTrace(double[][] matrix)
```

Übungsblatt 9

Kurzaufgabe 9.1:

Der String *s* habe den Wert `"abcde"`. Weisen Sie der char-Variablen *c* mit charAt(int i) das 2. Zeichen des Strings zu. Geben Sie *c* aus.

Kurzaufgabe 9.2:

Testen Sie, ob das erste und letzte Zeichen eines Strings `p` identisch sind.

Aufgabe 9.1: Zeichen zählen

Schreiben Sie eine Funktion

```
public static int getCharCount(String s, char c)
```

die zählt, wie häufig das Zeichen `c` im String `s` vorkommt.

Aufgabe 9.2: Großbuchstaben einlesen

Schreiben Sie eine Funktion

```
public static String getCapitalString()
```

die einen String von der Tastatur einliest, die Zeichen in Großbuchstaben wandelt und das Ergebnis zurückgibt.
Verwenden Sie nicht die API Methode String.toUpperCase(). Hinweis: Buchstaben sind A-Z und a-z. Umlaute müssen nicht beachtet werden.

Aufgabe 9.3: ASCII-Tabelle

Schreiben Sie ein Programm, das eine ASCII-Tabelle von ASCII-Wert 0 bis 127 auf dem Bildschirm ausgibt.

Aufgabe 9.4: Umkehrung

Schreiben Sie eine Funktion

```
public static String reverse(String s)
```

die eine Zeichenreihe umdreht. Beispiel:
Aus `Donaudampfschiffahrtsgesellschaftskapitän`
soll `nätipakstfahcsllesegstrhaffihcsfpmaduanoD` werden.

Aufgabe 9.5: Lückentext

Schreiben Sie eine Funktion

```
public static String clozeText(String s)
```

die aus einem beliebigen String `s` einen Lückentext macht, indem sie jedes 4. Zeichen durch ein Leerzeichen ersetzt.

Aufgabe 9.6: Trennung von Parametern

Schreiben Sie eine Funktion

```
public static int[] getParameters()
```

Die Funktion soll den Benutzer nach einer Parameterzeile fragen. In der Parameterzeile kann der Benutzer beliebig viele, durch ein Komma getrennte Integer-Werte eingeben. Diese Zeile soll das Programm in ein Integer-Feld verwandeln und das Ergebnis zurückgeben.

Aufgabe 9.7: Neue deutsche Rechtschreibung

Schreiben Sie eine Funktion

```
public static String newGermanOrthography(String s)
```

Die Funktion verwandelt alle „sch“ eines eingegebenen Textes in „sh“. Der Titel dieser Aufgabe wird also zu „Neue deutshe Rechtshreibung“. Beachten Sie auch die Sonderfälle in denen ein Wort mit „sch“ anfängt oder mit „sch“ aufhört. Großschreibung und Kleinschreibung soll auch berücksichtigt werden. Verwenden Sie dazu nicht den `replace`-Befehl der API.

Aufgabe 9.8: Häufigkeit der Ziffern

Schreiben Sie eine Funktion

```
public static int[] getDigitCount(int n)
```

Die Methode gibt ein Feld mit 10 Elementen zurück, in dem die Häufigkeit jeder Ziffer von `n` eingetragen ist. Die Funktion soll auch mit negativen Zahlen funktionieren. Das Minuszeichen wird nicht gezählt.

Übungsblatt 10

Frage 10.1:

Wie überprüfen Sie, ob zwei Brüche `x` und `y` Klone voneinander sind? Die beiden Brüche müssen unterschiedliche Objekte sein, aber den gleichen Wert besitzen.

Frage 10.2:

Welche Zeilen in Listing 2 enthalten eine korrekte Definition eines Arrays? Kreuzen Sie die korrekten Definitionen an.

Listing 2: potentielle Definitionen eines Array in Java

```
a)  int f1 [2] = {5,3};
b)  int f2 [] = {7,1,9};
c)  int [] f3 = {5,3};
d)  int [] f4 = new int [2];
e)  int f5 [2];
f)  int f6 [] = new int [2];
g)  int f7 [] = new int [] {f3[0], f3[1]};
```

Frage 10.3:

Nennen Sie eine Funktion, die ein Feld kopiert.

Frage 10.4:

Was bedeutet das Schlüsselwort `null`?

Frage 10.5:

Wie deklariert man ein dreidimensionales `int`-Feld?

Frage 10.6:

Nennen Sie 4 Methoden der Klasse `String`.

Frage 10.7:

Wie lautet das Schlüsselwort zur Erzeugung eines Objekts?

Frage 10.8:

Was ist am Kommentar in der folgenden Zeile falsch?

```
Bruch b = new Bruch(1,2);  //Erzeugt eine neue Klasse Bruch.
```

Frage 10.9:

Wie erzeugt man ein Feld mit ausgefranstem Rand?

Frage 10.10:

Wie kann man verhindern, das eine Exception zum Programmabbruch führt?

Übungsblatt 11

Zusatzaufgabe 11.1: Formatierung

Schreiben Sie eine Funktion

```
public static void printPi(int decimals)
```

Die Funktion gibt die Zahl `Math.PI` mit `decimals` Nachkommastellen auf dem Bildschirm aus. Nutzen Sie `System.out.printf`.

Zusatzaufgabe 11.2: Feld ausgeben

Schreiben Sie eine Funktion

```
public static void printArray(int[][] arr)
```

die das Feld `arr` spaltenbündig formatiert auf dem Bildschirm ausgibt. Die Breite der Spalte richtet sich nach dem breitesten Eintrag der Matrix.

Zusatzaufgabe 11.3: Zahlen beliebiger Länge

Ganze Zahlen beliebiger Länge kann man mit der Klasse `BigInteger` (Paket `java.math`) erzeugen.

a) Erzeugen Sie ein `BigInteger`-Objekt mit dem Wert 1. Übergeben Sie im Konstruktor den String `"1"`.

b) Schreiben Sie ein Programm, dass eine ganze Zahl n von der Tastatur einliest und die Fakultät $n! = 1 \cdot 2 \cdot \cdots \cdot n$ auf dem Bildschirm ausgibt. Benutzen Sie dazu `BigInteger`-Objekte.

   Zur Kontrolle:

```
100!=933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565182862536979208272237582511852109168640000000000000000000000000
```

Hinweis: `BigInteger`-Objekte sind wie Strings unveränderlich. Die Multiplikation mit einer Zahl erzeugt stets ein neues `BigInteger`-Objekt.

Hans Joachim Pflug IT-Center, RWTH Aachen; FH Aachen, Campus Jülich

Übungsblatt 12

Schreiben Sie die im folgenden genannten Klassen jeweils mit:

- den benötigten Attributen (private),
- Getter- und Setter-Methoden, die dafür sorgen, dass die Attribute keine unerlaubten Werte annehmen können (solange bei der Aufgabe nichts anderes genannt wird, soll eine Exception geworfen werden),
- einer toString-Methode
- und einem passenden Konstruktor.
- Schreiben Sie jeweils eine Testfunktion, in der Sie alle Eigenschaften der Klasse austesten.

Aufgabe 12.1: Punkt

Schreiben Sie eine Klasse `Punkt` für einen Vektor im $\mathbb{R}^2$.

Aufgabe 12.2: Komplexe Zahl

Schreiben Sie eine Klasse `Komplex` für eine komplexe Zahl.

Aufgabe 12.3: Gerade Zahl

Schreiben Sie eine Klasse `GeradeZahl`. Die Klasse hat ein einziges Attribut, das eine gerade Zahl sein muss. Wird versucht, eine ungerade Zahl zu setzen, wird statt dessen die nächstniedrigere gerade Zahl genommen.

Aufgabe 12.4: Notenverwaltung

Schreiben Sie eine Klasse `Note` für eine Notenverwaltung. Gültige Noten sind an der Hochschule: 1,0; 1,3; 1,7; 2,0; 2,3; 2,7; 3,0; 3,3; 3,7; 4,0; 5,0.

Aufgabe 12.5: Geburtstag

Schreiben Sie eine Klasse `Geburtstag`, die den Tag und den Monat eines Geburtstags enthält. Der 29. Februar ist als Geburtstag erlaubt. Die Anzahl der Tage der einzelnen Monate ist: Januar 31, Februar 29, März 31, April 30, Mai 31, Juni 30, Juli 31, August 31, September 30, Oktober 31, November 30 , Dezember 31.

Aufgabe 12.6: Dominostein

Schreiben Sie eine Klasse `Dominostein`, die einen Dominostein repräsentiert. Beide Seiten des Dominostein können die Zahlen 0-6 enthalten.

Übungsblatt 13

Aufgabe 13.1: Copy-Konstruktor

Fügen Sie den Klassen `Punkt`, `Komplex`, `GeradeZahl`, `Note`, `Geburtstag` und `Dominostein` einen Copy-Konstruktor hinzu.

Aufgabe 13.2: Punkt

Fügen Sie der Klasse `Punkt` folgende Methode hinzu:

- `public void schiebe(double dx, double dy)` verschiebt den Punkt um den Wert (dx, dy).

Aufgabe 13.3: Komplex

Fügen Sie der Klasse `Komplex` folgende Methoden hinzu:

- `public double getBetrag()` gibt den Betrag der Zahl zurück.
- `public void konjugiertKomplex()` konjugiert die Zahl.
- `public Komplex getKonjugiertKomplex()` gibt ein neues Objekt zurück, welches den konjugiert komplexen Wert von `this` hat. Das `this`-Objekt bleibt unverändert.
- `public void add(Komplex k)` addiert den Wert `k`.
- `public void mult(Komplex k)` multipliziert den Wert `k`.

Aufgabe 13.4: GeradeZahl

Fügen Sie der Klasse `GeradeZahl` folgende Methoden hinzu:

- `public GeradeZahl getNachfolger()` gibt die nächsthöhere gerade Zahl zurück.
- `public GeradeZahl getSum(GeradZahl g2)` gibt die Summe von `this` und g2 zurück.
- `public GeradeZahl getProd(GeradZahl g2)` gibt das Produkt von `this` und g2 zurück.

Aufgabe 13.5: Note

Fügen Sie der Klasse `Note` folgende Methode hinzu:

- `public boolean hatBestanden()` gibt `true` zurück, falls die Note maximal 4,0 ist und ansonsten `false`.

Aufgabe 13.6: Geburtstag

Fügen Sie der Klasse `Geburtstag` folgende Methode hinzu:

- `public boolean equals(Gebutstag g2)` gibt `true` zurück, falls die Geburtstage `this` und `g2` übereinstimmen und ansonsten `false`.

Aufgabe 13.7: Dominostein

Fügen Sie der Klasse `Dominostein` folgende Methoden hinzu:

- `public void dreheUm()` dreht den Dominostein um, vertauscht also beide Seiten.
- `public int[] getValues()` gibt die Werte in einem `int`-Feld der Größe 2 zurück.

Hans Joachim Pflug IT-Center, RWTH Aachen; FH Aachen, Campus Jülich

Übungsblatt 14

Frage 14.1:

Wie heißen die drei Prinzipien der Objektorientierung?

Frage 14.2:

Dürfen Konstruktoren schon andere Methoden des `this`-Objekts aufrufen?

Frage 14.3:

Darf ein Objekt sich selbst als Attribut besitzen? Warum oder warum nicht?

Frage 14.4:

Zeichnen Sie das UML-Diagramm der Klasse `Komplex` vom letzten Aufgabenblatt.

Frage 14.5:

Welche Art von Variable findet sich im folgenden Codestück und wie ruft man sie auf?

```
public class Element {
  public static int pos;
}
```

Frage 14.6:

Was ist der Default-Wert für double-Attribute?

Frage 14.7:

```
String a = "abc";
String b = a;
a = a + "def";
```

Welche Werte haben `a` und `b`?

Frage 14.8:

```
int[] x = {1,2,3};
int[] y = x;
x[0]=4;
```

Welche Werte haben `x` und `y`?

Frage 14.9:

Wann müssen Invarianten erfüllt sein?

Frage 14.10:

In welchen Ausnahmefällen darf man Attribute `public` machen?

Übungsblatt 15

Zusatzaufgabe 15.1: Kreis

Implementieren Sie die Klasse `Kreis` für einen Kreis nach dem folgenden UML-Diagramm. Beachten Sie beim Copy-Konstruktor, dass das Attribut `mittelpunkt` eine Referenz auf einen Punkt ist. Die Methode `getAbstand` berechnet den kürzesten Abstand des Punktes `p` zum Rand des Kreises.

Kreis
- radius : double - mittelpunkt : Punkt
+ Kreis(radius : double, mittelpunkt : Punkt) + Kreis(k2 : Kreis) + getRadius() : double + setRadius(radius : double) + getMittelpunkt() : Punkt + setMittelpunkt(mittelpunkt : Punkt) + getFlaeche() : double + getUmfang() : double + getAbstand(p : Punkt) : double

Zusatzaufgabe 15.2: Dominokette

Schreiben Sie eine Klasse `Dominokette`, die eine Kette von Dominosteinen repräsentiert (siehe Bild).

Es ist möglich, an beide Enden der Kette weitere Steine anzufügen, wenn die Symbole passen (siehe Abbildung). Implementieren Sie folgende Konstruktoren und Methoden:

- Einen Konstruktor, der den ersten Dominostein als Übergabeparameter erhält.
- Eine `toString`-Methode. Das Ausgabeformat des abgebildeten Beispiels soll dabei sein:

  `[4,2][2,5][5,0][0,6][6,3][3,3]`

- Eine Methode

  ```
  public void fuegeRechtsAn(Dominostein d)
  ```

  die den Dominostein am rechten Ende anfügt. Gegebenenfalls muss der Stein vorher herumgedreht werden. Die Methode löst eine `NumberFormatException` aus, falls der Stein nicht angelegt werden kann. Die Methode `fuegeLinksAn` muss aus Zeitgründen nicht implementiert werden.

Zusatzaufgabe 15.3: Schachstellung

Schreiben Sie eine Klasse Schachstellung. Die Klasse erzeugt aus einem String, der eine Schachstellung in Kurzform enthält, die grafische Wiedergabe der Stellung. Der String enthält eine Folge von Kombinationen aus 3 Zeichen, die jeweils eine Figur darstellen. Das erste Zeichen steht für die Figur, wobei die weißen Figuren König, Dame, Turm, Springer, Läufer und Bauer durch die Großbuchstaben K, D, T, S, L und B gekennzeichnet sind. Die schwarzen Figuren haben die entsprechenden Kleinbuchstaben k, d, t, s, l und b. Diesem Zeichen folgen zwei Ziffern, die die Koordinaten (Zeile und Spalte) der Figur angeben. Z.B. kennzeichnet der String b22B23K24 eine Stellung mit einem schwarzen Bauern auf (Zeile 2/Spalte 2), einem weißen Bauern auf (2/3) und dem weißen König auf (2/4).
Schreiben Sie folgende Methoden:

a) Schachstellung(): Erzeugt ein leeres Schachfeld.

b) setzeSchachstellung(String s): Setzt die Figuren aus dem String s in das Schachfeld.

c) toString(): Überschreibt die toString()-Methode und gibt die gespeicherte Stellung zurück. Für eine Stellung b22K24l28B23k66 soll die Darstellung wie folgt aussehen:

```
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   | b | B | K |   |   |   | l |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   | k |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
```

Schreiben Sie eine main-Methode, die die Klasse Schachstellung testet.
Werfen Sie eine `IllegalArgumentException` mit entsprechender Fehlermeldung, wenn

a) die Bezeichnung der Figur ungültig ist (falsche Buchstaben),

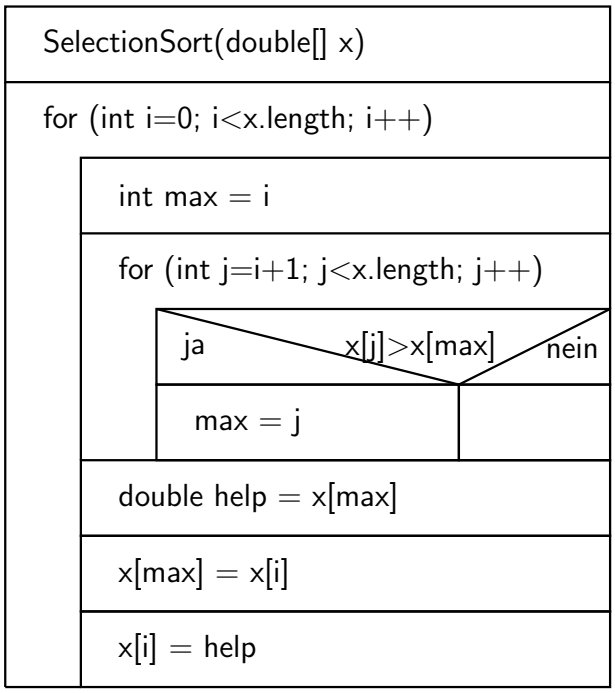
b) die Koordinaten nicht auf dem Feld liegen,

c) ein Feld zweimal besetzt wird.

Übungsblatt 16

Aufgabe 16.1: Code aus Struktogrammen

Das folgende Struktogramm beschreibt den Sortieralgorithmus „Selection Sort“.



a) Übersetzen Sie das Struktogramm in Java-Code.

b) Übersetzen Sie das Struktogramm in ein Flussdiagramm.

Aufgabe 16.2: Struktogramme aus Code

Gegeben ist der folgende Code, der das größte Element eines double-Feldes zuückgibt:

```
public static double getBiggest(double[] x) {
    //Groesstes Element ermitteln
    int index = 0;  //Index des groessten Elements
    for (int i=1; i<x.length; i++) {
        if (x[i]>x[index]) {
            index = i;
        }
    }
    return x[index];
}
```

Schreiben Sie zum Code

a) das Struktogramm.

b) das Flussdiagramm.

Aufgabe 16.3: Struktogramm aus Aufgabenstellung

In einem zweidimensionalen double-Feld soll ermittelt werden, welche Zeile die größte Zeilensumme (die Summe der Elemente einer Zeile) besitzt. Schreiben Sie zu diesem Problem das passende Struktogramm.

Übungsblatt 17

Aufgabe 17.1: Einbinden von Bibliotheken

FreeTTS ist eine in Java geschriebene Programmbibliothek zur Sprachausgabe.

- Finden Sie das Paket im Internet (Google oder Wikipedia) und laden Sie es herunter (bin-Paket, neueste Version). Entpacken Sie die Datei.
- Binden Sie alle jar-Dateien aus dem Unterordner lib in Eclipse ein. Eine Anleitung dazu finden Sie im Skript auf Seite 163 oben.
- Testen Sie die Programmbibliothek mit folgendem Code:

```
import com.sun.speech.freetts.*;

public class Sprachausgabe {

    public static void main(String[] args) {
        System.setProperty("freetts.voices",
          "com.sun.speech.freetts.en.us.cmu_us_kal.KevinVoiceDirectory");
        VoiceManager voiceManager = VoiceManager.getInstance();
        Voice voice = voiceManager.getVoice("kevin");
        voice.allocate();
        voice.speak("Hello student. Here is the Java program.");
        voice.deallocate();
    }
}
```

Aufgabe 17.2: Import-Anweisungenen

Lesen Sie den Abschnitt 9.2.1 im Skript und beantworten Sie die folgende Frage:
Beschleunigt sich ein Java-Programm, wenn man

```
import java.util.ArrayList;
```

statt

```
import java.util.*;
```

benutzt? Begründen Sie Ihre Antwort.

Aufgabe 17.3: Pakete

Tippen Sie das kleine graphische Hello World-Programm unten ab. Geben Sie vorher beim Erzeugen der Datei in Eclipse einen für Sie passenden Package-Namen an. Lesen Sie sich dazu zuerst die Abschnitte 9.2.2 und 9.2.3 durch. Der Code des Programms ist:

```
import javax.swing.JOptionPane;

public class GraphicHelloWorld {

    public static void main(String[] args) {
        JOptionPane.showInputDialog("Hello World");
    }
}
```

Beantworten Sie die folgenden Fragen:

- Wie lautet die package-Anweisung, die Eclipse automatisch erzeugt hat?
- In welchem Ordner ist die Datei abgelegt?

Aufgabe 17.4: jar-Dateien

a) Packen Sie Ihr Hello World-Programm aus Aufgabe 3 in eine jar-Datei. Benutzen Sie die Anleitung in Kapitel 9.1.2 des Skripts.

b) Starten Sie die jar-Datei durch Doppelklick im Windows-Explorer (nur Windows).

c) Öffnen Sie die jar-Datei mit WinZip (eventuell Umbennen der Endung in .zip):

d) Welche Dateien sind in der jar-Datei enthalten? Was ist der Inhalt der Datei Manifest.mf?

Aufgabe 17.5: Mehrere Klassen in einer Datei

Haben Sie sich auch schon geärgert, dass Sie für jede Java-Klasse eine eigene Datei schreiben mussten? Lesen Sie in den Kapiteln 9.3. und 9.4. im Skript nach, wie man mehrere Klassen in einer Datei zusammenfassen kann. Beantworten Sie dann die folgenden Fragen:

- Was ist die Einschränkung einer zweiten Klasse in einer Datei?
- Wie umgeht man diese Einschränkung?

Übungsblatt 18

Aufgabe 18.1: Programmierrichtlinien

Lesen Sie Abschnitt 10.1.1 des Skripts durch. Beantworten Sie danach die folgenden Fragen:

- Wozu dienen Programmierrichtlinien?
- Welche Namen sollen in Java groß, welche klein geschrieben werden?
- Warum sollen in arithmetischen Ausdrücken Klammern großzügig gesetzt werden?
- Warum sollen auch um einzeilige if-Blöcke geschweifte Klammern gesetzt werden?
- Wann soll man in Java Konstanten (Schlüsselwort `final`), wann Literale benutzen?

Aufgabe 18.2: JUnit-Tests

Testen Sie die Klasse `Bruch` mit einfachen JUnit-Tests. Erstellen Sie einen neuen JUnit-Test in Eclispe mit
$File \rightarrow New \rightarrow JUnitTestCase$.
Alle Methoden, vor denen die Annotation `@Test` steht, werden automatisch als Testfälle erkannt. Ersetzen Sie die automatisch erzeugte Methode durch:

```
@Test
public void multTest1() {
    Bruch b = new Bruch(32,17);
    b.mult(17);
    assertEquals("Zaehler falsch", 32 , b.getZaehler());
    assertEquals("Nenner falsch", 1, b.getNenner());
    //Alternative
    assertEquals("Ergebnis falsch", new Bruch(32, 1) , b);
}
```

Starten Sie Ihren Test mit *Run As../JUnit Test*. Beobachten Sie die Ausgabe auf der linken Seite. Der vorletzte Parameter der `assertEquals`-Anweisung ist der Erwartungswert. Ändern Sie ihn ab und provozieren Sie damit eine Fehlermeldung.

Aufgabe 18.3: JUnit-Tests 2

Sehen Sie sich die weiteren `assert`-Methoden unter
http://junit.org/junit4/
und dem Link *Assertions* an. Der Link *Getting started* ist als kurze Einführung auch lohnenswert. Fügen Sie anschließend die folgenden Tests hinzu:

- Testen Sie, ob die Berechnung der Inversen von $0/1$ eine ArithmeticException auslöst.
- Testen Sie, ob der Copy-Konstruktor wirklich eine Kopie erzeugt und nicht nur eine weitere Referenz.