6. Übungsblatt: Programmierpraktikum I (WS 2002/03)

Abgabe: 27. Januar 2002

Bem. 1: Für jede Aufgabe sind die angegebenen Dateien mit den jeweils angegebenen Namen abzugeben (Groß-/Kleinschreibung beachten!).

Bem. 2: Im Gegensatz zu den bisherigen Übungsblättern dürfen jetzt alle Java-Bibliotheken verwendet werden, die im Rechnerpool im D7 zur Verfügung stehen.

Java: Pythagoras

(`Pythagoras.java`, 5 Punkte)

Sind die Koordinaten (x, y) eines Punktes im zweidimensionalen Raum gegeben, so kann man dessen Abstand d zum Nullpunkt mit der bekannten Formel

$$d = \sqrt{x^2 + y^2}$$

berechnen (Satz des Pythagoras).
Schreibe ein Programm, das genau diese Aufgabe erfüllt, allerdings mit der Besonderheit, dass die Hauptmethode `main(String[] args)` ihre Argumente direkt von der Konsole erhält , also z.B. in der Form

```
java Pythagoras 12 -6.2
```

Beachte dabei wie immer das korrekte Abfangen und Behandeln möglicher Eingabefehler!
Die Ausgabe muss in jedem Fall den berechneten Abstand zum Nullpunkt zurückgeben, die weitere Ausgestaltung bleibt dir überlassen. Ein Beispiel:

```
Koordinaten:            (12.0, -6.2)
Abstand zum Nullpunkt:  13.507
```

Java: Verschlüsselungsprogramm

(`Cipher.java`, 11 Punkte)

Eine *Substitutionschiffre* ersetzt die Zeichen eines Textes nach einem bestimmten Schlüssel durch andere Zeichen. Eine ganz einfache Variante einer solchen Substitutionschiffre soll nun implementiert werden. Die Klasse `Cipher` hat dabei die folgende Struktur:

```
public class Cipher {
  private char key[] = {'u','g','j','m','a','k','s','d','o','y',
                'b','r','f','t','i','h','c','l','v','n',
                'e','z','p','q','x','w'};
  public void convert() {...}
  public void encrypt() {...}
  public void decrypt() {...}
}
```

Das Hauptprogramm, mit dem du die Klasse `Cipher` testest, sollte Teil einer anderen Klasse sein und ist nicht mit abzugeben! Die drei Methoden der Klasse sollen folgendes leisten:

Konvertieren: Die Methode `convert` fasst den Inhalt der Datei `text.txt` als String auf und wandelt ihn zeichenweise nach folgenden Regeln um:

- Die Kleinbuchstaben 'a' bis 'z' bleiben unverändert.
- Die Großbuchstaben 'A' bis 'Z' werden in Kleinbuchstaben umgewandelt.
- Die Ziffern '0' bis '9' werden in die entsprechenden Worte 'null' bis 'neun' umgewandelt.
- Alle anderen Zeichen (Umlaute, Leerzeichen, Tabstopps, Zeilenumbrüche etc.) werden ignoriert.

Das Resultat wird in Blöcken zu je 5 Zeichen und in Zeilen zu je 8 Blöcken in die Datei `input.txt` geschrieben. Nach diesen Regeln würde beispielsweise der Text

```
Der Beamte S. wurde am 17. Januar mit 3 roten Rosen am Parade-
platz beobachtet.
```

in den Text

```
derbe amtes wurde amein ssieb enjan uarmi tdrei
roten rosen ampar adepl atzbe obach tet
```

umgewandelt.

Verschlüsseln: Die Methode `encrypt` verschlüsselt alle Kleinbuchstaben in der Datei `input.txt` mit Hilfe des Schlüssels im Feld `key`. Das bedeutet, dass der Buchstabe 'a' als 'u' verschlüsselt wird, der Buchstabe 'b' als 'g' usw. Das Ergebnis wird in die Datei `output.txt` geschrieben (immer noch in Blockform).

Entschlüsseln: Die Methode `decrypt` leistet das Gegenteil von `encrypt`: Sie entschlüsselt den Inhalt einer Datei `output2.txt` und schreibt das Ergebnis in Blockform in die Datei `input2.txt`.

Anmerkungen:

- Es ist ausdrücklich erlaubt, die obige Klasse um weitere Datenstrukturen und Methoden zu erweitern.
- Es gibt verschiedene Methoden, die Ein-/Ausgabe mit Dateien umzusetzen. Etwas Recherche in der Bibliothek, in der Online-Hilfe oder im WWW kann hier nicht schaden.
- Welche Klasse du letztlich benutzt, um die Dateien anzusteuern, bleibt dir überlassen. Es ist allerdings darauf zu achten, dass alle Ausnahmen, die auftreten können, auch abgefangen und behandelt werden.

Java: Integral-Abschätzung

(`Aufgabe3.java`, 9 Punkte)

Das Integral einer stetigen Funktion f im Intervall $[a, b]$ entspricht der Fläche unter dem Graphen von f, wie in Abbildung a) dargestellt. Diese Fläche kann man bekanntlich näherungsweise berechnen, indem man sie durch viele Rechtecke approximiert wie in Abbildung b). Dabei wird die Abschätzung umso besser, je feiner die Zerlegung des Intervalles $[a, b]$ ist.

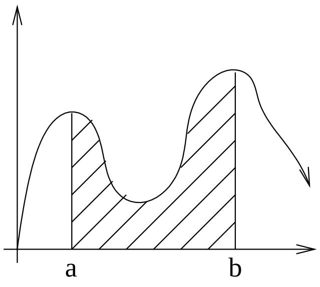

Abb. a)

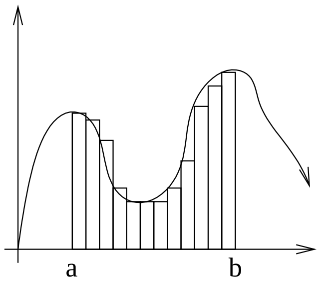

Abb. b)

Es soll nun ein Programm geschrieben werden, das diesen Approximationsalgorithmus für beliebige stetige Funktionen $f(x)$ implementiert, indem es das Intervall $[a, b]$ in n gleich große Teilintervalle $[x, y]$ zerlegt und für jedes dieser Intervalle die Fläche unter der Kurve durch ein entsprechendes Rechteck abschätzt. Die Abschätzung für das Gesamtintegral ergibt sich dann als Summe dieser Rechtecke. Um alle Unklarheiten zu beseitigen, hier noch einmal in formaler Schreibweise:

$$\int_a^b f(x)dx \quad \approx \quad \sum_{k=0}^{n-1} f(a + k \cdot \Delta x) \cdot \Delta x \quad \text{mit} \quad \Delta x = \frac{b-a}{n}$$

Zu implementieren sind die folgenden Elemente:

- Ein Interface `Zielfunktion`, das eine Methode `f(double)` definiert.

- Drei Klassen, die das Interface `Zielfunktion` implementieren und die Funktion f wie folgt definieren:

Klasse	*Funktion*
`Konst3`	$f(x) = 3$
`SqurPlus`	$f(x) = x^2 + 3x + 1$
`RootPlus`	$f(x) = \sqrt{x} + 1$

- Eine Methode

```
public static double integral(Zielfunktion g, double a,
                               double b, int n),
```

  die das Integral $\int_a^b g(x)$ durch n Rechtecke nach dem obigen Algorithmus approximiert.

- Ein Hauptprogramm

```
public static void main(String args[]),
```

  das die Methode `integral` durch geeignete Aufrufe für alle drei Funktionen testet.

Anmerkungen:

- Eine Ein-/Ausgabe-Schnittstelle ist nicht erforderlich; du kannst daher auch davon ausgehen, dass die Funktion `integral` nur korrekte Inputs erhält (stetige Funktion, $b > a$ usw.).
- Die korrekten Werte der Integrale für die Funktionen `Konst3`, `SqurPlus` und `RootPlus` sind leicht zu berechnen (siehe Analysis 1). Auf diese Weise lässt sich auch nachprüfen, ob die Ergebnisse deiner Integral-Funktion näherungsweise stimmen können!