Übung05: Methoden, Rekursion

Abgabetermin: 12.04.19, 15:30 Uhr

Name: _____ Matrikelnummer: _____

Aufgabe	**schriftlich abzugeben**	**elektronisch abzugeben**	**gelöst**
Aufgabe13	Java Programm, Testausgaben	Java Programm	☐
Aufgabe14	Java Programm, Testausgaben	Java Programm	☐
Aufgabe15	Java Programm, Testausgaben	Java Programm	☐
Aufgabe16	Java Programm, Screenshots	Java Programm	☐

Achtung!

Bitte auf diesem Deckblatt:

- **Name** und **Matrikelnummer** ausfüllen,
- gelöste Aufgaben **ankreuzen**

und dann der schriftlichen Abgabe als erste Seite anheften.

Aufgabe13: Implementieren einfacher Methoden

In dieser Aufgabe sollen einfache Methoden implementiert werden, die für gegebene Zahlen bzw. Zahlenfolgen Berechnungen durchführen. Setzen Sie folgende Methoden um:

- `static int abs(int x)`
  Berechnet für eine gegebene Zahl den Betrag.
- `static int min(int[] values)`
  Liefert das kleinste Element aus einer gegebenen Zahlenfolge.
- `static double average(double[] values)`
  Berechnet für eine gegebene Folge von Gleitkommazahlen den Durchschnitt.
- `static boolean exists(int[] values, int x)`
  Prüft, ob eine gegebene Zahl in einer Zahlenfolge vorkommt.

Hinweise:

Implementieren Sie die Methoden in der selben Datei und innerhalb einer gemeinsamen Java Klasse. Rufen Sie diese Methoden dann zum Testen für mehrere festgelegte Input-Werte in einer `main` Methode auf. Geben Sie die Ergebnisse auf der Konsole aus.

Abzugeben sind:

- Java Programm
- Testausgaben

Aufgabe14: CaesarCipher mit Methoden

Gegeben ist die Lösung der Aufgabe *CaesarCipher* aus der letzten Übung. Strukturieren Sie das Java Programm sinnvoll um und erstellen Sie neue Methoden für in sich geschlossene Ablaufe im Programm (z.B. `encryptFile`, `decryptFile`, `encrypt`, `decrypt`, `transform`, …). Testen Sie das neue Programm anschließend in mehreren Durchläufen, um die Korrektheit nach wie vor sicherzustellen.

Abzugeben sind:

- Java Programm
- Testausgaben

Aufgabe15: Rekursive Taylorreihe

Das Ergebnis der Exponentialfunktion e^x kann für ein gegebenes x mit Hilfe einer Taylorreihe annähernd berechnet werden. Eine Taylorreihe wird wie folgt gebildet:

$$taylor(x) = \sum_{i=0}^{\infty} \frac{1}{i!} x^i = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \frac{1}{4!}x^4 + \frac{1}{5!}x^5 + \frac{1}{6!}x^6 \ldots \approx e^x$$

Schreiben Sie eine Methode

```
static double expTaylor(double x, int i)
```

welche die Exponentialfunktion e^x mit Hilfe einer Taylorreihe approximiert. Die Summe der einzelnen Glieder soll dabei rekursiv gebildet werden. D.h. ein Aufruf von `expTaylor` soll das aktuelle Glied für ein gegebenes `x` und `i` berechnen, dann `expTaylor` rekursiv für `x` und `i-1` aufrufen und beide Ergebnisse addieren. `i==0` ist die Abbruchbedingung für die Rekursion. In diesem Fall wird als Ergebnis eins zurückgegeben.

Implementieren Sie weiters für die Fakultätsfunktion, die zum Bilden der Glieder in der Taylorreihe bentigt wird, eine rekursive Methode

```
static double factorial(double n)
```

welche für ein gegebenes `n` den Wert n! berechnet.

Lesen Sie in der `main` Methode einen x Wert von der Konsole ein, sowie einen Maximalwert für i. Rufen Sie dann Ihre `expTaylor` Methode für die beiden Werte auf. Berechnen Sie zusätzlich mit Hilfe der JDK-Funktion `Math.exp(x)` einen zweiten Wert für die Exponentialfunktion. Geben Sie beide Werte, sowie die Differenz zwischen ihrer approximierten Lösung und dem Ergebnis der JDK-Funktion aus. So können Sie testen, wieviele rekursive Aufrufe bzw. Glieder für die Taylorreihe benötigt werden, um für ein gegebenes x das Ergebnis entsprechend zu approximieren. Testen Sie ihre Methode mit unterschiedlichen Eingaben, und vergleichen Sie ihre approximierte Lösung mit der der JDK-Funktion.

Beispielausgabe:

```
x: 5
i_max: 12

         x:      5,000000000
 expTaylor:    148,113534955
  Math.exp:    148,413159103
Difference:      0,299624148
```

Abzugeben sind:

- Java Programm
- Testausgaben

Aufgabe16: Rekursive drawLine-Implementierung

Gegeben ist eine Klasse `Window` die es erlaubt, ein grafisches Fenster mit der Methode `Window.open(int width, int height)` zu öffnen. Darin können grafische Elemente gezeichnet werden. Der Aufruf von `Window.drawCircle(int x, int y, int radius)` zeichnet z.B. einen Kreis mit einem bestimmten Radius an einer bestimmten Position im Fenster, wobei (0, 0) links oben den Koordinatenursprung markiert. Positive x/y Werte verschieben das Element nach rechts/unten. Implementieren Sie eine eigene Methode

```
static void drawLine(int x1, int y1,
                     int x2, int y2,
                     int radius, int distance)
```

die rekursiv eine punktierte Linie zwischen zwei gegebenen Punkten zeichnet. Die Linie soll wie folgt gezeichnet werden:

- `drawLine` berechnet den Abstand zwischen den gegebenen Punkten (x1, y1) und (x2, y2).
- Ist der Abstand kleiner als `distance`, werden die beiden Punkte über Aufrufe der Methode `Window.drawCircle` mit gegebenem Radius gezeichnet.
- Ist der Abstand größer oder gleich `distance`, wird der Mittelpunkt zwischen (x1, y1) und (x2, y2) berechnet. Anschließend erfolgen zwei rekursive Aufrufe von `drawLine` für (x1, y1) und den berechneten Mittelpunkt (mx, my), sowie für (mx, my) und den zweiten Punkt (x2, y2).

Testen Sie Ihre Methode, indem Sie in der `main` Methode ein Fenster öffnen und `drawLine` mehrmals für bestimmte Punkte aufrufen. Sie können zu Debugging-Zwecken in der Konsole für jeden rekursiven Aufruf ausgeben, mit welchen Parametern ihre Methode aktuell aufgerufen wird und so überprüfen, ob ihre Methode wie oben beschrieben arbeitet.

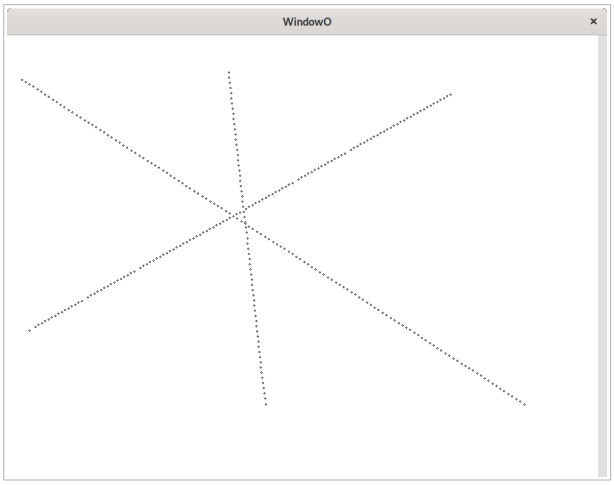
Beispielausgabe:

```
Window.open(800, 600);
int x1, y1, x2, y2;

x1 = 30; y1 = 400; x2 = 600; y2 = 80;
drawLine(x1, y1, x2, y2, 10, 1);

x1 = 300; y1 = 50; x2 = 350; y2 = 500;
drawLine(x1, y1, x2, y2, 10, 1);

x1 = 20; y1 = 60; x2 = 700; y2 = 500;
drawLine(x1, y1, x2, y2, 10, 1);
```



Abzugeben sind:

- Java Programm
- Screenshots der grafischen Ausgabe