Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences
Fachbereich Informatik
Prof. Dr. Rudolf Berrendorf

Einführung in die Programmierung
WS 19/20
Ausgabe : 28.10.2019
Bearbeiten bis : 03.11.2019

Übung 5

Aufgabe 1 :
Auf unserer Java-Webseite `http://berrendorf.inf.h-brs.de/Java/Java.html` finden Sie unter dem Punkt `Dokumentationen` die Java Code Conventions. Dies sind (freiwillige) Konventionen zum Schönschreiben von Java-Programmen, die jeder Programmierer anwenden sollte. Teilweise werden diese Konventionen automatisch von eclipse angewandt (zum Beispiel das Einrücken von Zeilen). Lesen und verstehen Sie die Code Conventions. Notieren Sie in einer Zusammenfassung (z.B. tabellarisch) die aus ihrer Sicht wesentlichen Punkte dieser Konventionen.

An einigen Stellen kommen Java-Konstrukte vor, die Sie noch nicht kennen. Ignorieren Sie diese Stellen im Moment.

Ab sofort sind diese Code Conventions verbindlich für alle Java Programme, die Sie schreiben.

Der Zeitbedarf für diese Aufgabe beträgt ca. 2-3 Stunden!

Aufgabe 2 :
Übergeben werden an ihr Programm zwei ganzzahlige Werte, die Sie in zwei `int`-Variablen `i, j` speichern sollen. Sortieren Sie den Inhalt dieser Variablen `i` und `j` so, dass nachher gilt, dass der Inhalt von `i` kleiner gleich dem Inhalt von `j` ist.

Aufgabe 3 :
Die Reihe $\sum_{i=0}^{\infty} \frac{(-1)^i}{2i+1} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + - \ldots$ konvergiert gegen $\frac{\pi}{4}$. Schreiben Sie drei Programme, die daraus eine Näherung für π bestimmen. Dafür soll

1. das erste Programm solange Reihenglieder berechnen und aufaddieren, bis insgesamt $n = 100000$ Reihenglieder aufaddiert wurden,
2. das zweite Programm solange Reihenglieder berechnen und aufaddieren, bis der Betrag des neuen Summanden $\leq \varepsilon = 0.0000001$ ist,
3. das dritte Programm solange Reihenglieder berechnen und aufaddieren, bis der berechnete Wert vom Wert `Math.PI` weniger als 0.00000001 abweicht.

Hinweise:

- Der Ausdruck $(-1)^i$ bewirkt nur einen Wechsel des Vorzeichens in jeder Iteration. Dazu brauchen (sollen!) Sie keine Potenz berechnen, sondern es genügt, den Faktor 1 in jeder Iteration mit -1 zu multiplizieren.
- Beachten Sie, dass die Formel oben gegen $\frac{\pi}{4}$ konvergiert, Sie aber π annäherungsweise berechnen sollen.

Aufgabe 4 :
Erweitern Sie die frühere Aufgabe zur Caesar-Verschlüsselung dahingehend, dass statt eines einzelnen Buchstabens ein kompletter String verschlüsselt werden kann, indem sukzessive alle einzelnen Zeichen des Strings verschlüsselt und auf dem Bildschirm ausgegeben werden.

Hinweis: `t.length()` und `t.charAt(i)`.

Beispiel:

```
java Caesar2
```

mit der Eingabe

```
1
HALLO
```

produziert die Ausgabe

```
Buchstabe H verschluesselt I
Buchstabe A verschluesselt B
Buchstabe L verschluesselt M
Buchstabe L verschluesselt M
Buchstabe O verschluesselt P
```

Aufgabe 5 :
Ein Rattenpärchen kann jeden Monat ca. 12 kleine Ratten werfen, die nach ca. zwei Monaten selbst geschlechtsreif sind. Unter der Annahme, dass genau gleich viele männliche wie weibliche Ratten in einem Wurf sind und Ratten niemals sterben: wieviele Ratten existieren mit obigen Annahmen nach n Monaten, wenn ein Wurf genau 12 kleine Ratten hat und die Geschlechteraufteilung gleichmäßig ist und wir davon ausgehen, dass das erste Rattenpärchen bereits im ersten Monat einen Wurf Junge bekommt?

Aufgabe 6 :
Nach der speziellen Relativitätstheorie nimmt die dynamische Masse eines Körpers zu, je schneller er sich (in einem Inertialsystem) bewegt. Die Formel dazu ist: $m = \frac{m_0}{\sqrt{1-v^2/c^2}}$, wobei m_0 die Ruhemasse (das Gewicht) des Körpers ist, v in m/s die Geschwindigkeit und $c = 3 \cdot 10^8\ m/s$ die Lichtgeschwindigkeit ist.

Schreiben Sie ein Java-Programm, das einen Körper ihrer Wahl (zum Beispiel ihren Körper) mit dem Gewicht $m_0\ kg$ von $v = 0\ m/s$ Richtung Lichtgeschwindigkeit $v = 3 \cdot 10^8\ m/s$ „beschleunigt“ (im Sinne von: zu mehreren zunehmenden Geschwindigkeiten die dynamische Masse ausrechnen). Da es erst nahe der Lichtgeschwindigkeit interessant wird, erhöhen Sie ihre Geschwindigkeit so, dass sie zu Beginn sehr schnell zunimmt, zum Ende hin aber langsamer, also mehr Messpunkte vorhanden sind (wie kann man das in einer Schleife ausdrücken?). Geben Sie für jeden Berechnungspunkt (Geschwindigkeit) jeweils die Geschwindigkeit und die dynamische Masse aus.

Aufgabe 7 (1 Punkt) :
Dies ist eine Praktomataufgabe!

Schreiben Sie ein Java-Programm **`DatumErweiterung`**, das zu einer Datumsangabe eine Zeit addiert und das resultierende Datum ausgibt.

Die Eingabe geschieht dabei über die Tastatur zur Laufzeit und enthält in dieser Reihenfolge folgende Angaben (alles natürliche Zahlen):

- eine Jahreszahl j $(1600 \leq j \leq 3000)$
- eine Monatszahl m $(1 \leq m \leq 12)$
- eine Tageszahl t $(1 \leq t \leq 31)$ (je nach Monat bis zum zulässigen Maximalwert des Monats)
- eine Stundenzahl s $(0 \leq s \leq 23)$
- eine Minutenzahl m_1 $(0 \leq m \leq 59)$
- eine Minutenzahl m_2, die zu dem Datum addiert werden soll ($0 \leq m_2 \leq 40000$).

Auf das Datum, das mit den ersten 5 Werten spezifiert wird, soll man die letzte Angabe m_2 addieren, also das Datum bestimmen, das m_2 Minuten später ist.

Ignorieren Sie Schaltjahre, der Februar hat also in jedem Jahr 28 Tage! Aufgrund der Beschränkung bei m_2 kann es höchstens einen Monat Überlauf geben. (Sie können sich ja einmal überlegen, welche zusätzliche Komplexität durch den Einbezug von Schaltjahren und durch größere m_2-Werte hinzukommen würde).

Die Ausgabe soll nacheinander in einer Zeile und durch ein Leerzeichen getrennt die Resultatwerte für Jahr, Monat, Tag, Stunde und Minute ausgeben. Die Zeile wird durch ein Zeilenende abgeschlossen.

Beispiel 1: Bei Eingabe von

```
2018 10 7 16 30 30
```

muss als Ausgabe erscheinen:

```
2018 10 7 17 0
```

weil 30 Minuten nach 16:30 am 7.10.2018 ist: 17:00 am 7.10.2018.

Beispiel 2: Bei Eingabe von

```
2018 12 31 23 59 2
```

muss als Ausgabe erscheinen:

```
2019 1 1 0 1
```

weil 2 Minuten nach 23:59 am 31.12.2018 ist: 0:01 am 1.1.2019

Aufgabe 8 (1 Punkt) :
Dies ist eine Praktomataufgabe!

Geben Sie ein Java-Programm **`LaengsteKette`** an, das bestimmt, was die Länge der längsten Kette gleicher natürlicher Zahlen in der Eingabe von der Tastatur ist.
Die Beispieleingabe `1 1 2 2 2 3 3 3 3 4 3 3 1 5 .` liefert 4, weil 4 aufeinanderfolgende Dreien vorhanden sind und keine längere Kette existiert.

Die Zahlen der Eingabe werden abgeschlossen durch einen einzelnen Punkt. Sie können folgende Programmteile verwenden, um eine unbekannte Anzahl an Werten nach diesem Schema einzulesen.

```
import java.util.*;
...
    Scanner sc = new Scanner(System.in);
    while(sc.hasNextInt()) {
        int i = sc.nextInt();
    }
```

Die Anzahl der einzulesenden Werte wird größer als 0 sein und nach oben hin ist die Anzahl offen.

Überlegen Sie sich, was mögliche Fälle sind und wie Sie diese behandeln. Überprüfen Sie ihr Verfahren / Programm vor der Abgabe erst per Hand / auf dem Papier, dann in eclipse mit verschiedenen Testwerten und geben erst dann ab, wenn dies alles korrekte Ergebnisse liefert.

Wie immer: Sie dürfen nur Sprachkonstrukte verwenden, die in der Vorlesung auch bereits vorgestellt wurden!