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

Einführung in die Programmierung
WS 19/20
Ausgabe : 21.10.2019
Bearbeiten bis : 27.10.2019

Übung 4

Aufgabe 1 :
Angenommen, Sie haben zur Darstellung von reellen Zahlen nur 6 Binärziffern zur Verfügung. In einer Festkommadarstellung (keine Gleitkommadarstellung!) werden davon 3 Vorkommabinärziffern und 3 Nachkommabinärziffern definiert, also kein Vorzeichen und kein Exponent. Oder anders ausgedrückt: jede Zahl hat das Aussehen xxx,yyy, wobei xxx die binären Vorzeichenziffern und yyy die binären Nachkommaziffern sind. Was ist:

1. die kleinste exakt darstellbare Zahl größer Null
2. die größte exakt darstellbare Zahl

Geben Sie jeweils die Darstellung mit Binärziffern und den jeweiligen Wert im Dezimalsystem an.

Aufgabe 2 :
Schreiben Sie ein Java-Programm, das für die Float-Werte x = 2304f, y = -4096f und z = 4096.001953125f die Berechnungen x*(y+z) und x*y + x*z ausführt. Wie erklären Sie sich das Ergebnis? Wie lautet das exakte Ergebnis?

Hinweise, um eine Erklärung zu finden:

1. Stellen Sie die Werte für x,y und z zuerst als Zweierpotenzen dar. Beispiel zu 2304f: $2304 = 2^{11} + 2^8$.
2. Führen Sie die Berechnungen anschließend Schritt für Schritt per Hand aus (ohne Taschenrechner!), indem Sie die Einzeloperation auf den Zweierpotenzwerten berechnen und das (Zwischen-)Ergebnis auch wieder über Zweierpotenzen ausdrücken. Berechnen Sie jedes Zwischenergebnis einzeln! Beispiel: $y+z = (-2^{12}) + (2^{12} + 2^{-9}) = 2^{-9}$.
3. Führen Sie die Berechnungen per Hand einmal mathematisch exakt aus. Was ist das Ergebnis? Vergleichen Sie dies mit der Ausgabe ihres Programms.
4. Führen Sie anschließend die Berechnung unter der Restriktion aus, dass das IEEE-Format zu 32 Bit auf eine Mantissenlänge von 23 Bit beschränkt ist. Überlegen Sie sich, wo das in ihrer Berechnung eine Rolle spielt und was dies für Auswirkungen während der Berechnung hat. Vergleichen Sie den darüber erhaltenen Wert wiederum mit der Ausgabe ihres Programms.

Aufgabe 3 :
Schauen Sie sich die Funktionalität an, die die Klasse `Math` laut der Java API-Beschreibung in Form von Methoden anbietet.

Allgemeine Anmerkung: Sie finden diese Beschreibung grundlegender Java-Klassen, wenn Sie auf die Java-Seite bei Oracle gehen (`https://docs.oracle.com/javase/13/`), dort auf den Link `API Documentation` klicken (unter `Specifications` zu finden), dort bei den Modulen auf `java.base` klicken und zuletzt auf `java.lang` klicken. Durch den Klick auf einen der Links unter `Class Summary` erhalten Sie detaillierte Informationen zu dieser Klasse.

Aufgabe 4 :
Schreiben Sie drei Java-Programme, die jeweils drei ganze Zahlen addieren und das Ergebnis auf dem Bildschirm ausgeben. Die Programme sollen sich in der Eingabe der drei Zahlen unterscheiden: Eingabe über die Kommandozeile, Eingabe über die Tastatur während des Programmlaufs und Eingabe aus einer Datei.

Hinweis: Schauen Sie sich auf der Java-Seite (ganz unten) die Hinweise und Beispiele zum Einlesen von Werten an.

Aufgabe 5 (1 Punkt) :
Dies ist eine Praktomataufgabe!

Es gibt in der Finanzmathematik verschiedene Berechnungsansätze der Verzinsung auf ein Anfangskapital K_0. Beispiele sind:

1. Einfache Verzinsung: $K_n = K_0 * (1 + \frac{p}{100} * n)$
2. Zinseszins: $K_n = K_0 * (1 + \frac{p}{100})^n$
3. Effektive unterjährige Verzinsung: $K_n = K_0 * (1 + \frac{i}{m})^{m*n}$

Dabei bedeuten:

K_0	Startkapital
K_n	Endkapital nach n Jahren
p	Zinsfuß (Beispiel 5, üblicherweise angegeben als 5%)
i	Zinssatz ($= p/100$; Beispiel: bei 5% Zinsfuß ist der Zinssatz 0,05)
n	Anzahl Jahre der Verzinsung
m	Anzahl Verzinsungen pro Jahr

Geben Sie ein Programm **`Zinsen`** an, das für beliebige zulässige Werte K_0, p, n, m jeweils das Endkapital nach den drei Berechnungsmethoden berechnet und jeden dieser Werte in einer Zeile ausgibt.

Beispiel: Der Aufruf des Programms

```
java Zinsen 100000 2 30 12
```

(100000 Euro für 2% Zinsfuß für 30 Jahre und unterjähriger Verzinsung jeden Monat) erzeugt die Ausgabe

```
160000.0
181136.15841033548
182120.89792366745
```

Verwenden Sie folgendes Programmgerüst, das es ermöglicht, dass Sie (und der Praktomat) ihrem Programm beliebige Werte für die Parameter K_0, p, n, m mitgeben können:

```
public static void main(String[] args) {
    double K0 = Double.parseDouble(args[0]);
    double p = Double.parseDouble(args[1]);
    double n = Double.parseDouble(args[2]);
    double m = Double.parseDouble(args[3]);
    ...
```

Aufgabe 6 :
Schauen Sie sich die Funktionalität an, die die Klasse **`String`** laut der Java API-Beschreibung in Form von Methoden anbietet. Nach welchen Arten von Funktionalitäten könnte man die Methoden klassifizieren?

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

Eine einfache und alte Verschlüsselungmethode für Texte ist die Caesar-Verschlüsselung. Ausgangspunkt ist ein Buchstabe b und ein Schlüssel $k \in \mathbb{N}$. Das Verfahren arbeitet so, dass der Buchstabe durch den Buchstaben ersetzt wird, der k Buchstaben weiter im Alphabet erscheint. Dabei ist zu beachten, dass beim "Hinauslaufen" über Z mit A weiter gezählt wird. Der nächste Buchstabe nach Z ist also A. Beispiel: $b = H$ und $k = 1$ ergibt als Ergebnis I.

Schreiben Sie ein Java-Programm **`Caesar`**, das für einen Schlüssel k mit $k \in \mathbb{N}$ (also z.B. $k = 100$) und einen Großbuchstaben b den verschlüsselten Großbuchstaben auf dem Bildschirm ausgibt. Die beiden Argumente werden über die Tastatur dem Programm übergeben, zuerst der Schlüssel k in einer Zeile und anschließend der Buchstabe b in einer Zeile.

Die Ausgabe besteht nur aus dem kodierten Resultatbuchstaben zu Beginn der Ausgabezeile gefolgt von einem Zeilenumbruch.

Beispiel: Für die Eingabe

```
5
X
```

ist die Ausgabe

```
C
```

Überlegen Sie sich, wie Sie mit den Hilfmitteln, die bis jetzt bekannt sind, das "Hinauslaufen" geeignet behandeln können. Nutzen Sie Eigenschaften der Zeichencodierung aus sowie geeignete arithmetische Operationen. Wie könnte die Dekodierung funktionieren?

Hinweise: Modulo-Operation, Zeichen - 'A', Typumwandlung zwischen **`int`** und **`char`**

Aufgabe 8 :
Schreiben Sie ein Java-Programm, das bei Laufzeit von der Tastatur eine Eingabezeile in folgendem Format (in EBNF beschrieben) einliest:

```
(vor|nach) <c> in <string>
```

Darin bezeichnet **`<c>`** ein vom Nutzer gewähltes Zeichen und **`<string>`** eine vom Nutzer gewählte Zeichenfolge — **`vor`**, **`nach`** und **`in`** sind buchstäbliche Zeichenfolgen ("Terminale" in der EBNF-Formulierung).

Die Zeichenfolge **`<string>`** soll kein Leerzeichen enthalten.
Ein Beispiel einer solchen Eingabe wäre:

```
nach s in durstiger
```

Das Programm soll dann die Zeichenkette ausgeben, die nach (bzw. vor bei Eingabe von **`vor`**) dem ersten (bzw. letzten) Vorkommen des Zeichens **`<c>`** in der Zeichenkette **`<string>`** steht. Nach der Ausgabe soll ein Zeilenumbruch erfolgen.
Im gegebenen Beispiel wäre dies:

```
tiger
```

Kommt das Zeichen **`<c>`** nicht in der Zeichenkette **`<string>`** vor, soll eine leere Zeile ausgegeben werden.

Wenn die Eingabe nicht dem vorgeschriebenen Format entspricht, soll eine Fehlermeldung ausgegeben werden.

Hinweise:

- Verwenden Sie Methoden der Klasse **`String`** (siehe Aufgabe 1)!
- Sie können mit einer Selektionsanweisungen arbeiten, die in Java den Aufbau hat:

```
if ( ein logischer Ausdruck ) {
   Aktion 1
} else {
   Aktion 2
}
```