1. Aufgabe

Schreiben Sie ein Java-Konsolprogramm, das folgendes leistet:

- Es definiert drei Variable vom Typ **`int`**.
- Es fordert den Bediener auf, dazu drei Zahlen einzugeben, wobei die erste Zahl dezimal, die zweite oktal und die dritte hexadezimal eingegeben werden soll. Es gibt anschließend jede dieser drei Zahlen mit geeigneter Beschriftung in den drei Formaten aus (d.h. drei Zahlen eingeben, neun Zahlen ausgeben). Die Definition (Eingabeformate = Literalformate) oktaler und hexadezimaler Literale (Werte) können Sie dem Skript entnehmen, für die Ausgabe (dezimal, oktal, hexadezimal) siehe „Ausgabeformatierung - Konvertierungszeichen“. Beispiel:

```
Dezimalzahl> 10                              //Eingaben
Oktalzahl> 010
Hexazimalzahl> 0x10
dezimal: 10, oktal: 12, hexadezimal: a       //Ausgaben
oktal: 10, dezimal: 8, hexadezimal: 8
hexadezimal: 10, oktal: 20, dezimal: 16
```

2. Aufgabe

Zum Testen der Integer- Arithmetik unter Java schreiben Sie ein Konsolprogramm, das folgendes leistet:

- Es definiert drei Integer-Variable **`a`**, **`b`**, **`c`** vom Typ **`int`**.
- Es fordert den Bediener auf, zwei ganze Zahlen einzugeben und legt die eingegebenen Zahlen in den Variablen a und b ab.
- Es berechnet nacheinander die folgenden fünf Ausdrücke

```
(1) c = a + b
(2) c = a - b
(3) c = a * b
(4) c = a / b
(5) c = a % b
```

und gibt sofort nach jeder Berechnung das Ergebnis **`c`** mit geeigneter Beschriftung am Bildschirm aus.

Machen Sie Experimente:

- Versuchen Sie die Resultate zu erklären,
  welche Sie bei der Eingabe sehr großer Werte bei der Multiplikation erhalten.
- Erklären Sie die Resultate, die sich bei der Division ergeben?
  Was passiert speziell bei der Division durch 0?
- Welche Ergebnisse liefert der „**`%`**“-Operator (z.B. auch bei **`%0`**)?
  Hinweis: wenn **`b == 0`** ist sollten Sie den vierten Ausdruck auskommentieren,
  um den fünften Ausdruck zu erreichen und zu testen.

3. Aufgabe

Schreiben Sie ein Konsolprogramm (analog zur vorherigen Aufgabe) das folgendes leistet:

- Es definiert drei **`double`**-Variable **`a, b, c`**. Für die Variablen **`a`** und **`b`** wird jeweils wieder ein Wert eingelesen.
- Berechnen Sie die gleichen Ausdrücke wie in Aufgabe 2.
- Variieren und testen Sie Ihr Programm auch mit Extremwerten: groß, klein, 0.
- Beobachten und interpretieren Sie die Ergebnisse.

Zu seltsamen Ergebnissen siehe auch Anhang 1.

4. Aufgabe

Berechnen sie folgende Anweisungen und geben Sie jeweils das Ergebnis von **`j`** aus.
Die Variable **`j`** soll vom Typ "**`double`**" sein.

- **`j = 7/5*3`**
- **`j = 8.0 * 3 / 4`**
- **`j = 8.0 * (3 / 4)`**

Erklären Sie sich die Ergebnisse.

5. Aufgabe

Der Bruttopreis ist mit 100 Euro inkl. 19 % USt. angegeben. Wie viel macht die Umsatzsteuer aus?
Wie hoch ist der Nettopreis?
Anmerkungen:

Der Steuerbetrag ist brutto * prozentsatz / (100 + prozentsatz)
Der Nettopreis ist der Bruttopreis minus den Steuerbetrag.

Definieren Sie geeignete Variablen und dazugehörige Datentypen.
Setzen Sie damit die Berechnung um.
Umsatzsteuer und Nettopreis sollen mit 2 Stellen hinter dem Komma ausgegeben werden.

6. Aufgabe

Überlegen Sie, wie man mit Hilfe der Modulo-Operation bei einer Gleitkommazahl den gebrochenen Anteil ermitteln bzw. abspalten kann. Erstellen Sie ein entsprechendes Beispiel-Programm, welches zu einer eingegebenen Gleitkommazahl den ganzzahligen Anteil und den gebrochenen Anteil getrennt ausgibt.

7. **Aufgabe –** *Semantische Fehler*

Die Datei Durchschnitt.java hat folgenden Inhalt:

```
public class Durchschnitt
{
      public static void main(String[] args)
      {
            int summe = 1;

            int ersteZahl = 3;
            System.out.println("erste Zahl : "+ersteZahl);
            summe = summe + ersteZahl;

            int zweiteZahl = 2;
            summe = summe + ersteZahl;
            System.out.println("zweite Zahl :"+zweiteZahl);
            double durchschnitt = summe / 2;
            System.out.println("Der Durchschnitt der beiden Zahlen ist: "
            + durchschnitt);
      }
}
```

Das Programm enthält drei semantische (logische) Fehler. Diskutieren Sie, worin diese Fehler bestehen und wie man sie beheben kann. Korrigieren Sie das Programm entsprechend.

8. Aufgabe

Zum Testen der Fließkommaarithmetik unter Java schreiben Sie ein Konsolprogramm, das folgendes leistet:

- Es definiert drei **`double`**- Variable **`x, y, z`**.
- Es fordert den Bediener zur Eingabe von zwei Gleitkommazahlen auf und liest diese in die Variablen **`x`** und **`y`** ein.

Es berechnet der Reihe nach mit möglichst wenigen Operationen (Operatorzeichen) die folgenden in mathematischer Notation geschriebenen Ausdrücke:

a) $z = x^2y^2 - 4xy + 4$
b) $z = \frac{(1+xy)^2}{1+(1+xy)^4}$
c) $z = xy + (3-x)y - y$
d) $z = 2y$

Nach jeder Berechnung soll das Ergebnis **`z`** mit 7 gültigen Stellen angezeigt werden.
Eine einfache Umformung zeigt, dass die Ausdrücke c) und d) übereinstimmen.
Probieren Sie aus, welche Werte Sie für diese Ausdrücke erhalten,
wenn Sie **`x = 1.5e30`** und **`y = 2e10`** eingeben.
Wie erklärt sich die auffallende Diskrepanz zwischen Ergebnissen von c und d in diesem Fall, obwohl **`x`** und **`y`** und sämtliche Zwischenergebnisse hier noch weit innerhalb der **`double`**-Genauigkeit liegen?

9. Aufgabe

Schreiben Sie ein Windows- Konsole- Programm, das folgende Variable definiert:

- **`n, m`** : Typ **`int`**,
- **`x, y`** : Typ **`double`**
- **`Ax`** : Variablen **`A0`** bis **`A6`** vom Typ double, die mit den Werten
  `1.0, 2.5, 0.1, 0.0, 0.8, 0.0, 1.5` vorbelegt sind.

Das Programm fordert zunächst den Bediener zur Eingabe von Werten für **`n, m`** und **`x`** auf. Anschließend berechnet es mit möglichst wenigen Rechenoperationen (Operatorzeichen) der Reihe nach die folgenden fünf in mathematischer Notation geschriebenen Ausdrücke für **`y`** und gibt das Ergebnis jeweils am Bildschirm aus
(Hinweis: Hilfsvariable dürfen nach Bedarf definiert werden - Beachten Sie evtl. Auslöschungseffekte).
Die mathematischen Funktionen und Konstanten sollen der „**`Math`**“-Library entnommen werden:

Ausdruck 1: $y = log(|n - m|x)$

Ausdruck 2: $y = sin(\frac{n+m}{2}x) - cos(\frac{n-m}{2}x)$

Ausdruck 3: $y = \frac{1}{2}\frac{n-m}{n+m}xe^{-(n+m)x^2}$

Ausdruck 4: $y = A_0 - A_1x + A_2x^2 - A_3x^3 + A_4x^4 - A_5x^5 + A_6x^6$

Ausdruck 5: $y = \frac{(e^x+x)^2}{\frac{1}{2}x+e^x} - e^x$

Das jeweilige Ergebnis y soll mit 4/7 gültigen Nachkomma-Stellen angezeigt werden
(mit **`System.out.printf`**).
Kontrolle der Berechnung (abschließende Nullen sind weggelassen):

n	m	x	y
2000	1000	0.002	Ausdruck 1: 0.6931 Ausdruck 2: -0.3992 Ausdruck 3: 0.0003294 Ausdruck 4: 0.9950 Ausdruck 5: 0.003001
-50000	40000	0.01	Ausdruck 1: 6.802 Ausdruck 2: 0.9925 Ausdruck 3: 0.1223 Ausdruck 4: 0.9750 Ausdruck 5: 0.01502
4	6	10.0	Ausdruck 1: 2.996 Ausdruck 2: 0.5767 Ausdruck 3: 0 Ausdruck 4: 1.508e+6 Ausdruck 5: 15
1	2	100.0	Ausdruck 1: 4.605 Ausdruck 2: -1.68 Ausdruck 3: 0 Ausdruck 4: 1.5e+12 Ausdruck 5: 0

Hinweis bei falschen Ergebnissen: Prüfen Sie ob nicht eine (ungewollte) Integer-Division das Problem ist.

Anhang 1

Betrachten (und probieren) Sie folgendes Beispielprogramm:

```
public class Prog
{
      public static void main(String[] args)
      {
            double big_d = 1.79769313486231570e308 * 10;
            System.out.println("Unendlich =\t\t" + big_d);
            System.out.println("Unendlich - Unendlich =\t"+(big_d-big_d));
            System.out.println("Unendlich / Unendlich =\t"+(big_d/big_d));
            System.out.println("Unendlich * 0.0 =\t" + (big_d * 0.0));
            System.out.println("0.0 / 0.0 =\t\t" + (0.0/0.0));
      }
}
```

Es liefert die Ausgabe:

```
Unendlich =       Infinity
Unendlich - Unendlich = NaN
Unendlich / Unendlich = NaN
Unendlich * 0.0 =       NaN
0.0 / 0.0 =             NaN
```

NaN steht für Not a Number!

Optionale Aufgabe zum Anhang:

Die Lösung der quadratischen Gleichung $ax^2 + bx + c = 0$ liefert die Formel: $x_{1,2} = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$

Schreiben Sie ein Programm, das drei double-Koeffizienten a, b, c einliest und die Lösungen (x_1, x_2) ausgibt. Was liefert das Programm mit zwei, einer und keiner Lösung?

Anmerkung: Wurzeln zu geraden Exponenten aus negativen Zahlen können keine reellen Zahlen sein, weil gerade Potenzen reeller Zahlen nie negativ sind.