JAVA EXAMPLES

REMARK

It turned out that all Workstation in the classroom are NOT set equally. This is why I wil demonstrate all examples using an on-line web tool http://www.browxy.com and/or http://www.compileonline.com .

Please consider using this tool as an alternative to your NetBeans.

GENERAL

This explanations are for absolute beginners. Skilled programmers should (and probably will) use more effective approach.

EXAMPLE 1 ACCOUNT

TASK

Imagine you have $10 000 on your account. Your bank credits 4% interest for you and you pay maintenance fee $5 monthly. What will be your amount at the end of 5th year?

THOUGHTS

Imagine how you would do it manually, i.e. with paper, pen and calculator.

To calculate amount at the end of the 1st year, you will take the original amount, increase it by 4% (interests) and decrease by 12(months) * $5 (monthly fee). Interests are calculated **before** fee subtraction.

To calculate amount at the end of the 2nd year, it will be very similar. You will take the amount from the end of 1st year, increase it by 4% (interests) and decrease by 12(months) * $5 (monthly fee).

To calculate amount at the end of the 3rd year, it will be very similar. You will take the amount from the end of 2nd year, increase it by 4% (interests) and decrease by 12(months) * $5 (monthly fee).

And so on.

You see, that you have a block of computations, that repeats for every year. So you will create a loop that loops 5-times to calculate 5 years and inside the loop will be your every-year-calculation.

SOLUTION

```
public class Test {
  public static void main(String args[]) {

      double amount = 100000;
```

```
        /* amount MUST be double, because we will add very small parts
        (interests) to it */

        double interests = 0.04;
        /* interests must be double. Remember you cannot write 4%,
        because in JAVA character % has a special meaning. */

        double fee = 50;
        /* It is not necessary to use double for monthly fee; we could
        use int as well. But double is better, because in the future we
        may need fees like decimal number.  */

        int no_years = 5;
        /* number of years to calculate */

        /* now, we will create the loop. We can use for loop here. To
        pass the loop several times, we need a special variable, pass
        counter. We can name it current_year, because in fact it
        describes for which year we do the calculation. Variable
        current_year will be int, as we do not expect fraction of the
        year. It is not necessary to initialize current_year, it will
        be initialized later */
        int current_year;

        for (current_year = 1; current_year<=no_years; current_year++)
        /* program will loop from year 1 to year 5 inclusive; after
        each pass the current_year increases by one */

              {
              /* calculating new amount from the old one */
              amount = amount + (amount * interests) - (12.0 * fee);

              /* print the results */
              System.out.print("At the end of " + current_year );
              System.out.println("th year, amount is " + amount );
              };

    }
}
```

RESULTS

```
At the end of 1th year, amount is 103400.0
At the end of 2th year, amount is 106936.0
At the end of 3th year, amount is 110613.44
At the end of 4th year, amount is 114437.9776
At the end of 5th year, amount is 118415.496704
```

EXAMPLE 2 LOANS

TASK

You took a loan of $10,000 with interests 12% p.a. and with $75 monthly fee. You pay $3000 pro year. How many years you need to pay the loan?

THOUGHTS

This is a task very similar to Example 1.

But there still is a difference: you do not know number of years. So you cannot use condition `current_year<=no_years.` Instead, you must loop until you paid, i.e. till your loan is greater than zero. For this case you alternatively can use `while` loop.

To calculate loan at the end of the 1st year, you will take the original loan, increase it by 12% (interests) and decrease by 12(months) * $75 (monthly fee).

SOLUTION

```
public class Test {
  public static void main(String args[]) {

      double loan = 10000;
      /* loan MUST be double, because we will add very small parts
      (interests) to it */

      double interests = 0.12;
      /* interests must be double. Remember you cannot write 12%,
      because in JAVA character % has a special meaning. */

      double fee = 75;
      /* It is not necessary to use double for monthly fee; we could
      use int as well. But double is better, because in the future we
      may need fees like decimal number.  */

      double payment = 3000;
      /* your payment at the end of every year */

      int no_years;
      /* our result: number of years to pay */

      /* now, we will create the loop. We can use for loop here. To
      pass the loop several times, we need a special variable, pass
      counter. We can name it current_year, because in fact it
      describes for which year we do the calculation. Variable
      current_year will be int, as we do not expect fraction of the
      year. It is not necessary to initialize current_year, it will
      be initialized later */
      int current_year;

      for (current_year = 1; loan>=0; current_year++)
      /* program will loop from year 1 till the loan is paid; after
      each pass the current_year increases by one */

            {
            /* calculating new loan from the old one */
            loan = loan + (loan * interests) + (12.0 * fee);
```

```
            loan = loan - payment;

            /* print the results */
            System.out.print("At the end of " + current_year );
            System.out.println("th year, loan is " + loan );
            };

    }
}
```

RESULTS

```
At the end of 1th year, loan is 9100.0
At the end of 2th year, loan is 8092.0
At the end of 3th year, loan is 6963.040000000001
At the end of 4th year, loan is 5698.604800000001
At the end of 5th year, loan is 4282.437376000001
At the end of 6th year, loan is 2696.3298611200007
At the end of 7th year, loan is 919.8894444544007
At the end of 8th year, loan is -1069.7238222110711
```

You can see that your loan is fully paid during 8th year.

EXAMPLE 3 MONTHLY PAYMENT

TASK

You took a loan of $10,000 with interests 12% p.a. and with $75 monthly fee. You pay $250 pro year. How many months you need to pay the loan? All calculations will be done monthly

THOUGHTS

This is a task very similar to Example 2. The only difference is that all calculations take place on the end of month instead of the year.

SOLUTION

```
public class Test {
  public static void main(String args[]) {

      double loan = 10000;
      /* loan MUST be double, because we will add very small parts
      (interests) to it */

      double interests = 0.12;
      /* interests must be double. Remember you cannot write 12%,
      because in JAVA character % has a special meaning. */

      double fee = 75;
```

```
        /* It is not necessary to use double for monthly fee; we could
        use int as well. But double is better, because in the future we
        may need fees like decimal number.  */

        double payment = 250;
        /* your payment at the end of every month */

        int no_months;
        /* our result: number of months to pay */

        /* now, we will create the loop. We can use for loop here. To
        pass the loop several times, we need a special variable, pass
        counter. We can name it current_month, because in fact it
        describes for which month we do the calculation. Variable
        current_month will be int, as we do not expect fraction of the
        month. It is not necessary to initialize current_month, it will
        be initialized later */
        int current_month;

        for (current_month = 1; loan>=0; current_month++)
        /* program will loop from month 1 till the loan is paid; after
        each pass the current_month increases by one */

              {
              /* calculating new loan from the old one
                 p.a. interests are divided into 12 months */
              loan = loan + (loan * interests/12) + fee;
              loan = loan - payment;

              /* print the results - shortened */
              if (loan<2000)
                {
                System.out.print("At the end of " + current_month );
                System.out.println("th month, loan is " + loan );
                }
              };

    }
}
```

RESULTS

```
At the end of 73th month, loan is 1993.2227108415123
At the end of 74th month, loan is 1838.1549379499274
At the end of 75th month, loan is 1681.5364873294268
At the end of 76th month, loan is 1523.3518522027211
At the end of 77th month, loan is 1363.5853707247484
At the end of 78th month, loan is 1202.2212244319958
At the end of 79th month, loan is 1039.2434366763157
At the end of 80th month, loan is 874.6358710430789
At the end of 81th month, loan is 708.3822297535096
At the end of 82th month, loan is 540.4660520510447
At the end of 83th month, loan is 370.87071257155515
```

```
At the end of 84th month, loan is 199.57941969727068
At the end of 85th month, loan is 26.575213894243348
At the end of 86th month, loan is -148.1590339668142
```

You can see that your loan is fully paid during 86th month. Also, you can see that after 84th month, i.e. after 7th year, your loan still is 199.58, while in example 2 (yearly calculating) is fully paid. Results that monthly calculating is better for banks (…and this is why it is used by banks).

EXAMPLE

Compute area and perimeter of a circle, `r = 25`.

SOLUTION

```
public class Pi0{

     public static void main(String []args){
        double pi = 3.141592653589793;
        double r = 25.0;
        double perimeter = 2*pi*r;
        double area = pi*r*r;
        System.out.println("area=" + area);
        System.out.println("perimeter=" + perimeter);
     }
}
```

EXAMPLE: SIMPLE INPUT/OUTPUT

this example was deleted: sorry for discomfort

EXAMPLE: LUDOLF´S NUMBER Π (1)

Compute Ludolf´s number π as a sum of infinite queue, using formula:

$$\pi = 4 \cdot \sum_{n=0}^{\infty} (-1)^n \cdot \frac{1}{2n+1}$$

Pi = 4*(1 - 1/3 + 1/5 - 1/7 + 1/9 - 1/11 + …)

SOLUTION

```
public class Pi1{

    public static void main(String []args){

    int sig = 1;
    double pi = 0.0;
    double denom = 1.0;
```

```
        int n = 1;

        do {
          System.out.print("pass n= "+n);
            pi = pi + sig/denom;
          System.out.println(", pi= " + pi);
            sig = -sig;
            denom = denom + 2;
          n++;
        } while (denom < 10);

        pi = 4*pi;
        System.out.print("Result: Pi = " + pi);
      }

}
```

RESULT

Result, after 5 passes, is:

```
pass n= 1, pi= 1.0
pass n= 2, pi= 0.6666666666666667
pass n= 3, pi= 0.8666666666666667
pass n= 4, pi= 0.7238095238095239
pass n= 5, pi= 0.8349206349206351
Result: Pi = 3.3396825396825403
```

EXAMPLE: LUDOLF´S NUMBER Π (2)

Compute Ludolf´s number π as a sum of infinite queue, using formula:

$$\pi = 4 \cdot \sum_{n=0}^{\infty} (-1)^n \cdot \frac{\frac{1}{2^{2n+1}} + \frac{1}{3^{2n+1}}}{2n+1}$$

Pi = 4*((1/2+1/3) - (1/2^3+1/3^3)/3 + (1/2^5+1/3^5)/5 - (1/2^7+1/3^7)/7 + ...)

GUIDE

Theoretically, we can evaluate the expression in a loop, again and again. Unfortunately, computing high powers of 2 and 3 is too time-consuming. This is why, in practice, consider to compute the new expression for **n=n+1** using old resulrs for **n**.

When going from n → n+1 we obtain:

$$f(n) = \frac{\frac{1}{2^{2n+1}} + \frac{1}{3^{2n+1}}}{2n+1}$$

and

$$f(n+1) = \frac{\frac{1}{\frac{2^{2n+1}.1}{4}} + \frac{1}{\frac{3^{2n+1}.1}{9}}}{(2n+1)+2}$$

This will be used for computation. We will define a variable *clen2* (i.e. fraction with power of 2 in its denominator) and *clen3* (fraction with power of 3 in its denominator). During each cycle we will divide it by 4 or 9 respectively.

ŘEŠENÍ 1 – TRIVIÁLNÍ

```
OOPS! this is in C++, sorry
/* ludolf3.cpp */
/* urceni hodnoty Ludolfova cisla - trivialni */

#include <iostream>
using namespace std;

int main()
{
   double znamenko = -1;
   double suma = 0;
   double clen2 = 2;
   double clen3 = 3;
   double jmenovatel = 1;

   for (int n=0; n<=4; n++)
   {
      cout << "n=" << n;
      znamenko = -znamenko;
      cout << ", zn=" << znamenko;
      clen2 = clen2/4;
      cout << ", clen2=" << clen2;
      clen3 = clen3/9;
      cout << ", clen3=" << clen3;
      suma += znamenko*(clen2+clen3)/jmenovatel;
      jmenovatel = jmenovatel + 2;
      cout << ", jmenovatel="<< jmenovatel;
      cout << ", pi=" << 4*suma << endl;
   }
   cout << "vypoctena hodnota pi je "<< 4*suma << endl;
   cin.get();
}
```

PŘÍKLAD LUDOLFOVO ČÍSLO 5

Vypočítejte Ludolfovo číslo π jako součet nekonečné matematické řady podle vzorce

$$\pi = 4 \cdot \sum_{n=0}^{\infty} (-1)^n \cdot \frac{\frac{4}{5^{2n+1}} - \frac{1}{239^{2n+1}}}{2n+1}$$

Pi = 4*((4/5-1/239) - (4/5^3-1/239^3)/3 + (4/5^5-1/239^5)/5 - (4/5^7-1/239^7)/7 + ...)

POZNÁMKA 1

Tento výpočet je, až na jiné hodnoty koeficientů, prakticky shodný s předcházejícím.

ŘEŠENÍ

```
OOPS! this is in C++, sorry
/* ludolf5.cpp */
/* urceni hodnoty Ludolfova cisla - trivialni */

#include <iostream>
using namespace std;

int main()
{
   double znamenko = -1;
   double suma = 0;
   double clen2 = 4.0*5;
   double clen3 = 1.0*239;
   double jmenovatel = 1;

   for (int n=0; n<=4; n++)
   {
      cout << "n=" << n;
      znamenko = -znamenko;
      cout << ", zn=" << znamenko;
      clen2 = clen2/25;
      cout << ", clen2=" << clen2;
      clen3 = clen3/(239*239);
      cout << ", clen3=" << clen3;
      suma += znamenko*(clen2-clen3)/jmenovatel;
      jmenovatel = jmenovatel + 2;
      cout << ", jmenovatel="<< jmenovatel;
      cout << ", pi=" << 4*suma << endl;
   }
   cout << "vypoctena hodnota pi je "<< 4*suma << endl;
   cin.get();
}
```

Výsledky pro prvních 5 průchodů:

```
n=0, zn=1, clen2=0.8, clen3=0.0041841, jmenovatel=3, pi=3.18326
n=1, zn=-1, clen2=0.032, clen3=7.32498e-08, jmenovatel=5, pi=3.1406
n=2, zn=1, clen2=0.00128, clen3=1.28236e-12, jmenovatel=7, pi=3.14162
n=3, zn=-1, clen2=5.12e-05, clen3=2.24499e-17, jmenovatel=9, pi=3.14159
n=4, zn=1, clen2=2.048e-06, clen3=3.93024e-22, jmenovatel=11, pi=3.14159
vypoctena hodnota pi je 3.14159
```

PŘÍKLAD LUDOLFOVO ČÍSLO METODOU MONTE CARLO

Máme-li čtverec a do něj vepsanou kružnici, výběrem nekonečného počtu náhodných bodů uvnitř čtverce získáme čtvrtinu Ludolfova čísla PI, jako podíl počtu zásahů dovnitř kružnice a vně kružnice.

POZNÁMKA:

Výpočet pomocí metody Monte Carlo je zatížen chybou generátoru pseudonáhodných čísel a pro daný počet pokusů těžko určíme přesnost, s jakou se výsledek číslu PI blíží.

VYSVĚTLENÍ

Podíl obsahů čtverce a **do něj vepsané kružnice** je čtvrtina čísla PI:

$$\frac{S_{kružnice}}{S_{čtverec}} = \frac{\frac{\pi d^2}{4}}{d^2} = \frac{\pi}{4}$$

Podíl obsahů čtverce a do něj vepsané kružnice je čtvrtina čísla PI.

Odtud můžeme vyjádřit PI = ...

Zatímco plochu čtverce umíme exaktně spočítat jako druhou mocninu strany čtverce, plochu kružnice exaktně spočítat neumíme, protože k tomu bychom potřebovali znát právě číslo PI.

Ale namísto toho můžeme provést jednoduchou úvahu: kdybychom brali úplně náhodné body uvnitř čtverce (tzn. kdybychom pro každý bod generovali náhodná X a Y z rozsahu 0<=x,y<=d), tak procento případů, kdy se "trefíme" dovnitř kružnice, je úměrné poměru ploch kruhu a čtverce.

Stačí tedy vygenerovat dostatečný počet náhodných bodů a počítat, kolikrát se "trefíme" dovnitř čtverce a kolikrát ven. Podle výše uvedeného vzorce pak dopočítáme PI.

POZNÁMKA:

Jestli je bod uvnitř nebo vně kružnice poznáme snadno, protože pro body na kružnici platí Pythagorova věta

$$d^2 = x^2 + y^2$$

ŘEŠENÍ

Z výše uvedeného vzorce vychází

```
Pi = 4 * S_kruz / S_obd
Pi = 4 * Uvnitr / Vsechny
```

použitelnost vzorce samozřejmě závisí na kvalitě generátoru náhodných čísel. V knihovně *<cstdlib>* je generátor *rand()* , který generuje celá čísla v rozsahu *<0; RAND_MAX)* , přičemž RAND_MAX je konstanta, definovaná v knihovně. Kvalita tohoto generátoru je nevalná, jak se můžete přesvědčit v tomto příkladu.

POZOR!

Při psaní kódu narazíme na velmi zrádnou a nebezpečnou vlastnost C++. **Dělení dvou celých čísel dává celočíselný výsledek**! To znamená, že kdybychom dělili *rand() / RAND_MAX*, ve všech případech bychom dostali nulu (protože hodnota podílu se nachází v intervalu <0;1) .

Z tohoto důvodu je v kódu použito tak zvané **přetypování**:

```
(double) rand()
(double) RAND_MAX
```

Přetypování znamená, že kompilátor vezme za ním následující výraz (tedy *rand()* nebo *RAND_MAX*) a před použitím ho převede na uvedený typ, v našem případě *double*. Podíl dvou čísel *double* je také *double*, takže dostáváme to, co jsme očekávali.

Podobného efektu jsme dosáhli v řádce

```
Pi = 4.0 * Uvnitr / Vsechny;
```

tím, že jsme konstantu nezapsali jako 4 (což je *int*), ale jako 4.0 (což je *double*).

ZDROJOVÝ KÓD

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
    int Uvnitr=0, Vsechny;
    double x, y;
    double Pi;

    for (Vsechny=0; Vsechny<10; Vsechny++)
    {
        x = (double) rand()/(double) RAND_MAX;
        y = (double) rand()/(double) RAND_MAX;
        if ((x*x + y*y)<1)
            Uvnitr++;
        // ladici tisky
        cout << "x=" << x << ", y=" << y;
        cout << ", uvnitr=" << Uvnitr << endl;
    }
    Pi = 4.0 * Uvnitr / Vsechny;
    cout << "Pi = " << Pi << endl;

   return 0;
}
```

VÝSLEDEK

Výsledek pro prvních 10 iterací je:

```
x=0.840188, y=0.394383, uvnitr=1
x=0.783099, y=0.79844, uvnitr=1
x=0.911647, y=0.197551, uvnitr=2
x=0.335223, y=0.76823, uvnitr=3
x=0.277775, y=0.55397, uvnitr=4
x=0.477397, y=0.628871, uvnitr=5
x=0.364784, y=0.513401, uvnitr=6
x=0.95223, y=0.916195, uvnitr=6
x=0.635712, y=0.717297, uvnitr=7
x=0.141603, y=0.606969, uvnitr=8
Pi = 3.2
```

PŘÍKLAD LUDOLFOVO ČÍSLO METODOU "PIXEL COUNTING"

Tento výpočet je variantou předchozího.

Namísto nejistého generátoru náhodných čísel jednoduše rozdělíme čtverec na velký počet pixelů (třeba 1000 x 1000) a to, jestli je bod uvnitř nebo vně, spočítáme pro každý pixel. To znamená, že do sebe vnoříme dva cykly, jeden pro *x* a druhý pro *y*.

V našem příkladu jsme pro zrychlení použili čtvercovou síť jen 10 x 10.

ŘEŠENÍ

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
    int Uvnitr=0, Vsechny=100;
    int x, y;
    double Pi;

    for (x=0; x<10; x++)
        for (y=0; y<10; y++)
        {
            if ((x*x + y*y)<100)
                Uvnitr++;
            // ladici tisky
            cout << "x=" << x << ", y=" << y;
            cout << ", uvnitr=" << Uvnitr << endl;
        }
    Pi = 4.0 * Uvnitr / Vsechny;
    cout << "Pi = " << Pi << endl;

   return 0;
}
```

VÝSLEDEK:

Pro síť jen 10 x 10 vychází Pi = 3.44;