Les nombres

Le programme de démonstration est contenu dans le projet NetBeans *DemoNombres.java*.

1. Types simples des nombres

Les nombres peuvent être représentés selon plusieurs types simples :

- *byte* : entier signé codé sur 8 bits ;
- *short* : entier signé codé sur 16 bits ;
- *int* : entier signé codé sur 32 bits ;
- *long* : entier signé codé sur 64 bits ;
- *float* : les nombres à virgule flottante codé en simple précision (norme IEEE 754) ;
- *double* : les nombres à virgule flottante codé en double précision (donc plus précis que *float*).

Les types les plus utilisés sont : *int* et *double*.

2. Écriture des nombres dans le code source

Les nombres peuvent être écrit de différentes manières dans le code source.

Ainsi, avec l'écriture on peut préciser le type du nombre manipulé. On a :

- 10.0 ou 10.0d représente 10 en *double* ;
- 10.0f représente 10 en *float* ;
- 10 représente 10 en *int* ;
- 10L représente 10 en *long*.

Par ailleurs, pour faciliter la lecture des nombres, il est possible d'utiliser le caractère "_". Par exemple, 10000000 est équivalent à 10_000_000 qui est bien plus lisible.

Les nombres hexadécimaux sont écrits avec le préfixe 0x et les nombres binaires avec le préfixe 0b.

Il est possible d'utiliser la notation scientifique : 1.2e3 représente $1,2 * 10^3$.

3. Codage des nombres

Le codage en virgule flottante (qui correspond aux types *float* et *double*) ne permet pas toujours de mémoriser les valeurs exactes des nombres. Il y a des approximations qui sont réalisées.

Par exemple, le nombre 4735,67 est mémorisé sous la forme : 4735,67000000000007275957614183425903320312 5.

Démonstration :

```
  double n11 = 4735.67; // 4735.67 n'est pas mémorisé sous sa forme exacte
  System.out.println(n11); // donne   4735.67
  BigDecimal n12 = new BigDecimal(n11);
  System.out.println(n12);// donne 4735.67000000000007275957614183425903320312 5
```

Pour les montants monétaires, il est conseillé d'utiliser *BigDecimal*.

4. Formatage des nombres

Suivant les cultures, les nombres ne sont écrits de la même manière. Ainsi, les Anglais utilisent le point comme séparateur décimal et la virgule comme séparateur de milliers alors que les Français utilisent la virgule et l'espace.

Pour formater les nombres suivant des spécificités culturelles, il faut procéder comme suit :

```
  NumberFormat formatage1 = NumberFormat.getInstance(Locale.FRANCE);
  String chaine = formatage1.format(1234.56);
  System.out.println(chaine);
```

Pour les montants monétaires, il y aussi des particularités locales à prendre en considération. On procédera donc comme ceci :

```
   NumberFormat formatage2 = NumberFormat.getCurrencyInstance(Locale.FRANCE);
   chaine = formatage2.format(1234.56);
   System.out.println(chaine);
```

5. Code source de l'exemple

```
package demonombres;

import java.math.BigDecimal;
import java.text.NumberFormat;
import java.util.Locale;

/**
 *
 * @author A. FRANÇOIS
 */
public class DemoNombres {
    public static void main(String[] args) {
        // écriture des nombres
        double n1 = 10.0; // 10.0 : 10 en double
        double n2 = 10.0d;  // 10.0d : 10 en double
        float n3 = 10.0f; // 10.0f : 10 en float
        int n4 = 10; // 10 : 10 en int
        long n5 = 10L; // 10L : 10 en long

        // pour faciliter la lecture des nombres, il est possible d'utiliser le
caractère "_"
        // par exemple 10000000 est équivalent à 10_000_000
        int n6 = 10_000_000;
        double n7 = 0.123_456;
```

```
        int n8 = 0x41; // nombre codé en hexadécimal
        System.out.println("" + n8);
        int n9 = 0b1000001; // nombre codé en binaire
        System.out.println("" + n9);

        double n10 = 1.2e3; // notation scientifique : 1.2e3 équivaut à 1.2*10^3
soit 1200
        System.out.println("" + n10);


        // problème du codage des nombres
        double n11 = 4735.67; // 4735.67 n'est pas mémorisé sous sa forme exacte
        System.out.println(n11); // l'affichage nous trompe
        BigDecimal n12 = new BigDecimal(n11); // BigDecimal va nous permettre
d'afficher la valeur exacte du nombre
        // en réalité, en machine, 4735.67 est représenté approximativement avec
un double (même si l'approximation est très précise)
        System.out.println(n12); // on voit ici la vraie valeur de n11

        // si on veut mémoriser 4735.67 sous sa forme exacte
        // il faut utiliser BigDecimal et une chaine de caractères
        BigDecimal n13 = new BigDecimal("4735.67");
        System.out.println(n13); // 4735.67, valeur exacte cette fois


        // formatage des nombres
        NumberFormat formatage1 = NumberFormat.getInstance(Locale.FRANCE);
        String chaine = formatage1.format(n13);
        System.out.println(chaine);

        NumberFormat formatage2 =
NumberFormat.getCurrencyInstance(Locale.FRANCE);
        chaine = formatage2.format(n13);
        System.out.println(chaine);
    }
}
```