T.P. n°10 en
Programmation Orientée Objet en JAVA

Exercice - La méthode equals

Créer un package « egalite ». Les classes suivantes doivent appartenir au package créé.

1. Développer les classes suivantes :

```
public Tester1{
public static void main (String [] args){
Integer i1 = 1000;
Integer i2 = 1000;
if(i1 != i2) System.out.println("objets différents ");
if(i1.equals(i2)) System.out.println("significativement égaux");
}
}
```

Quel est le résultat ?

objets différents

significativement égaux

```
public Tester2{
public static void main (String [] args){
Integer i3 = 10;
Integer i4 = 10;
if(i3 == i4) System.out.println("même object");
if(i3.equals(i4)) System.out.println("significativement égaux");
}
}
```

Quel est le résultat ?

même object

significativement égaux

Pourquoi? != produit: i1 et i2 sont: objets différents, quant à = = produit: même objet?

=> deux instances des objets des classes enveloppes (wrappers) suivantes sont toujours = = quand leurs valeurs sont égaux:

- Boolean
- Byte
- Character from \u0000 to \u007f (7f is 127 in decimal)
- Short and Integer from -128 to 127

La méthode equals

Si les primitives sont exclus, toute chose en Java est un object. Chaque exception, chaque événement, chaque array hérite de java.lang.Object. quelques méthodes sont hérités de la classe Object telque la méthode equals qui se déclare comme suit :

boolean equals (Object obj)

L'opérateur == evalue à true uniquement quand deux références réfèrent au même objet (car == simplement voit les bits dans la variable, s'ils sont identiques ou non). Les classes String et les classes enveloppe ont redéfinie la méthode equals() (héritée de la classe Object), de tel façon il est possible de

comparer deux objets différents de même type pour voir si leur contenus sont significativement équivalents. Si deux instances Integer différentes ayant la veleur int 5 ils seront toujours égaux.

Exemple : Ecrire le code suivant dans un même fichier appelé EqualsTest.java qui redéfinie la méthode equals():

```
public class EqualsTest {
  public static void main (String [] args) {
     Moof one = new Moof(8);
     Moof two = new Moof(8);
     if (one.equals(two)) {
        System.out.println("one and two are equal");
     }
   }
}
class Moof {
  private int moofValue;
  Moof(int val) {
     moofValue = val;
  }
  public int getMoofValue() {
     return moofValue;
  }
  public boolean equals(Object o) {
    if ((o instanceof Moof) && (((Moof)o).getMoofValue()
         == this.moofValue)) {
      return true;
    } else {
       return false;
    }
  }
}
```

Imaginer que deux objets Moof sont les mêmes si leurs moofValue sont identiques. Donc, la méthode equals() est redéfinie pour comparer les deux moofValues.

Expliquation de la méthode equals():

```
1.  public boolean equals(Object o) {
2.    if ((o instanceof Moof) && (((Moof)o).getMoofValue()
            == this.moofValue)) {
3.       return true;
4.      } else {
5.        return false;
6.      }
7.  }
```

- La ligne 1: déclare une redéfinition valide de la méthode equals() héritée de Object.
- Ligne 2 : Logiquement, il faut deux choses pour faire une comparaison d'égalité valide:
  - **instanceof** test utilisé pour tester si l'objet est du type voulu
  - L'objet argument doit être prototypé au type correct pour pouvoir effectuer la comparaison.

Exercice 2 : les opérations sur les tableaux

Créer une classe Operation. Dans sa méthode main :
1) remplir le tableau args à l'aide du clavier puis l'afficher
2) copier le tableau args dans un tableau t avec la méthode clone() de la classe Object
3) trier le tableau t dans l'ordre naturel de ses éléments et l'afficher

Les méthodes sort() et toString() de la classe Arrays.

Le tri d'un tableau d'entiers ou de String se fait suivant l'ordre naturel avec la méthode sort de la classe Arrays. Par exemple :

```
int [] t = {3, 1, 11, 25, -17};
Arrays.sort(t);
```

Pour afficher le contenu d'un tableau, il est possible d'utiliser la méthode toString de la classe Arrays comme suit :

```
Arrays.toString(t)
```

Il faut importer la classe Arrays comme suit :

```
import java.util.Arrays;
```