Les constantes

Dans une application il arrive fréquemment que l'on utilise des valeurs numériques ou chaînes de caractères qui ne seront pas modifiées pendant le fonctionnement de l'application. Il est conseillé, pour faciliter la lecture du code, de définir ces valeurs sous forme de constantes.

La définition d'une constante se fait en ajoutant le mot clé final devant la déclaration d'une variable. Il est obligatoire d'initialiser la constante au moment de sa déclaration (c'est le seul endroit où il est possible de faire une affectation à la constante).

```
final double TAUXTVA=1.196;
```

La constante peut être alors utilisée dans le code à la place de la valeur littérale qu'elle représente.

```
prixTtc=prixHt*TAUXTVA;
```

Les règles concernant la durée de vie et la portée des constantes sont identiques à celles concernant les variables.

La valeur d'une constante peut également être calculée à partir d'une autre constante.

```
final double TOTAL=100;
final double DEMI=TOTAL/2;
```

De nombreuses constantes sont déjà définies au niveau du langage Java. Elles sont définies comme membres static des nombreuses classes du langage. Par convention les noms des constantes sont orthographiés entièrement en majuscules.

Les énumérations

Une énumération va nous permettre de définir un ensemble de constantes qui sont liées entre elles. La déclaration se fait de la manière suivante :

```
public enum Jours
{
    DIMANCHE,
    LUNDI,
    MARDI,
    MERCREDI,
    JEUDI,
    VENDREDI,
    SAMEDI
}
```

La première valeur de l'énumération est initialisée à zéro. Les constantes suivantes sont ensuite initialisées avec un incrément de un. La déclaration précédente aurait donc pu s'écrire :

```
public class Jours
{
```

Support exclusif ne peut être utilisé autrement que par du personnel NEEDEMAND

```
    public static final int DIMANCHE=0;
    public static final int LUNDI=1;
    public static final int MARDI=2;
    public static final int MERCREDI=3;
    public static final int JEUDI=4;
    public static final int VENDREDI=5;
    public static final int SAMEDI=6;
}
```

C'est approximativement ce que fait le compilateur lorsqu'il analyse le code de l'énumération.

En fait la déclaration d'une énumération est une déclaration de classe « déguisée ». Cette classe hérite implicitement de la classe java.lang.Enum. Les éléments définis dans l'énumération sont les seules instances possibles de cette classe. Comme n'importe quelle classe, elle peut contenir des attributs, des constructeurs et des méthodes. L'exemple de code suivant présente ces possibilités.

```
public enum Daltons
{
    JOE (1.40, 52),
    WILLIAM    (1.68, 72),
    JACK    (1.93, 83),
    AVERELL     (2.13, 89);


    private final double taille;
    private final double poids;


    private Daltons(double taille, double poids)
    {
        this.taille = taille;
        this.poids = poids;
    }
    private double taille() { return taille; }
    private double poids() { return poids; }


    double imc()
    {

        return poids/(taille+taille);
    }

}
```

Le constructeur est utilisé de manière implicite pour initialiser les constantes de chacun des éléments de l'énumération. Le constructeur d'une énumération doit obligatoirement être déclaré private. Plusieurs méthodes définies dans la classe de base (java.lang.Enum) permettent d'obtenir des informations sur les éléments de l'énumération. La méthode toString retourne une chaîne de caractères représentant le nom de la constante de l'énumération.

Support exclusif ne peut être utilisé autrement que par du personnel NEEDEMAND

```
Daltons d;
d=Daltons.JACK;
System.out.println(d.toString());
```

La méthode valueOf effectue l'opération inverse en fournissant un des éléments de l'énumération dont le nom est indiqué par la chaîne de caractères passée en paramètre.

```
d=Daltons.valueOf("JOE");
System.out.println("poids : "+ d.poids());
System.out.println("taille : "+ d.taille());
```

La méthode values retourne sous forme d'un tableau toutes les valeurs possible de l'énumération.

```
System.out.println("les frères Dalton");
for(Daltons d: Daltons.values())
{
System.out.println(d.toString());
}
```

Une fois définie, une énumération peut être utilisée comme un nouveau type de données. Vous pouvez donc déclarer une variable avec pour type votre énumération.

```
Jours repere;
```

Il est alors possible d'utiliser la variable en lui affectant une des valeurs définies dans l'énumération.

```
repère=Jours.LUNDI;
```

Lorsque vous faites référence à un élément de votre énumération, vous devez le faire précéder du nom de l'énumération comme dans l'exemple précédent. L'affectation à la variable d'autre chose qu'une des valeurs contenues dans l'énumération est interdite et provoque une erreur de compilation.

La déclaration d'une énumération ne peut pas se faire dans une procédure ou une fonction. Elle peut par contre être déclarée dans une classe mais il faudra dans ce cas préfixer le nom de l'énumération par le nom de la classe dans laquelle elle est définie lors de son utilisation. Pour que l'énumération soit autonome il suffit simplement de la déclarer dans son propre fichier.

La portée d'une énumération suit les mêmes règles que celle des variables (utilisation des mots clés public, private, protected).

Une variable de type énumération peut facilement être utilisée dans une structure switch ... case, il n'est dans ce cas pas nécessaire de faire précéder les membres de l'énumération du nom de l'énumération.

```
public static void testJour(Jours j)
    {
        switch (j)
        {
        case LUNDI:
```

Support exclusif ne peut être utilisé autrement que par du personnel NEEDEMAND

```
        case MARDI:
        case MERCREDI:
        case JEUDI:
            System.out.println("c\'est dur de travailler");
            break;
        case VENDREDI:
            System.out.println("bientot le week end !");
            break;
        case SAMEDI:
            System.out.println("enfin !");
            break;
        case DIMANCHE:
            System.out.println("et ca recommence !");
            break;
        }
    }
```

Support exclusif ne peut être utilisé autrement que par du personnel NEEDEMAND