Java - les méthodes (classiques, statiques, surcharge)

Date de dernière mise à jour : 27/06/2007 à 19:36

Les méthodes :

Les méthodes (aussi appelées fonctions par abus de langage) vont vous permettre d'effectuer des traitements généralement que vous effectuerez au moins deux fois dans le code (sinon la création d'une méthode ne vous permettra juste que d'éclaircir un peu votre code avec en contrepartie une petite perte de performances).

Une méthode se délimite comme une classe, c'est à dire par deux accolades. Vous êtes cependant obligé d'y ajouter des parenthèses, même si elles ne contiennent rien. Ce qu'on peut placer à l'intérieur des parenthèses s'appellent les paramètres.

Il existe différents types de méthodes que l'on peut globalement classer en trois familles :

-Les accesseurs : ces méthodes vont permettre de modifier un attribut privé d'une classe par une autre classe (il s'agit d'une méthode publique accessible aux autres classes).
-Les modificateurs : ces méthodes ne retournent rien et modifient la valeur d'un attribut privé. Il s'agit de méthodes publiques.
-Le reste : toutes les fonctions qui ne rentrent pas dans ces deux catégories.

Les droits d'accès :

Comme pour les attributs, les méthodes disposent des mêmes droits d'accès qui limitent leur utilisation :

-public : la méthode est accessible de n'importe où.
-private : la méthode est accessible uniquement dans la classe où elle figure.
-protected : la méthode est accessible dans la classe où elle figure et dans les classes héritées.

Exemple d'une méthode **MaMethode()** qui ne renvoie rien (le mot clé void est facultatif), et qui ne prend pas de paramètre d'entrée :

```
public
class
 Test{
   public
 Test(
)
   {
      MaMethode(
)
;   }
      public
void
 MaMethode(
)
   {
      System.out.println(
&amp;quot;Appel de la méthode&amp;quot;
)
;     }
```

```
}
```

Lorsque votre méthode doit retourner un type de donnée, vous devez remplacer "void" par le type de donnée que vous souhaitez.

Lorsque votre méthode doit retourner une donnée, on utilise le mot clé return. Attention, ce mot clé provoque l'arrêt de la fonction, c'est à dire que la donnée sera retournée et l'éventuel code se trouvant après le mot return ne sera pas exécuté. On peut donc utiliser le mot clé return pour interrompre une fonction comme ceci : **return;** (ceci est valable si votre fonction ne renvoie rien, sinon vous devez retourner le type de données approprié, par exemple 0 pour un type de données int).

Exemple d'une méthode acceptant un paramètre et retournant un entier :

```
public class
 Test{
   public
 Test()
   {
      MaMethode(50)
;   }
      public
int
 MaMethode(int
 variable)
   {
      System.out.println(
&amp;quot;Le nombre que vous avez passé en paramètre vaut : &amp;quot;
 + variable)
;       return
 variable + 50; //variable vaut maintenant 100
   }
}
```

La surcharge de méthodes :

La surcharge survient lorsque l'on a deux méthodes du même nom mais qui ne prennent pas les mêmes paramètres. Voici un exemple de surcharge de méthode :

```
public
class
 Test{
      public
 Test()
   {
      MaMethode()
;       MaMethode(50)
;   }


   public
void
 MaMethode(int
```

```
 variable)
    {
        System.out.println(
&amp;quot;Le nombre que vous avez passé en paramètre vaut : &amp;quot;
 + variable)
;    }


    public
void
 MaMethode()
    {
        System.out.println(
&amp;quot;Vous avez appelé la méthode sans paramètre &amp;quot;
)
;    }
}
```

Méthodes statiques :

Une méthode statique est une méthode qui peut être appelée même sans avoir instancié la classe. Une méthode statique ne peut accéder qu'à des attributs et méthodes statiques.

Exemple d'application avec une méthode statique :

```
public class
 Test{
    public int
 test;    public static
 String chaine =
&amp;quot;bonjour&amp;quot;
;        public
 Test(
)
    {
        MaMethodeStatique(
)
;    }
        public static void
 MaMethodeStatique(
)
    {
            int
 nombre =
 10;                System.out.println(
&amp;quot;Appel de la méthode statique : &amp;quot;
+
 nombre +
 chaine)
;       }
}
```

Vous pouvez sans avoir instancié la classe accéder à la méthode statique en tapant ceci : Test.MaMethodeStatique(); n'importe où dans votre code.