JAVA

Construction et initialisation
des objets dérivés

Construction et initialisation des objets dérivés

Quand il n'y a pas de constructeur, un pseudo-constructeur par défaut est appelé, nous nous intéressons maintenant au constructeur de la classe dérivée.

En Java, le constructeur de la classe dérivée doit prendre en charge l'intégralité de la construction de l'objet. Pour l'initialiser certains champs private de la classe de base, on peut soit utiliser des méthodes de modifications publiques, soit appeler le constructeur de la super-classe.

Attention. Si un constructeur d'une classe dérivée appelle un constructeur d'une classe de base, il doit obligatoirement s'agir de la première instruction du constructeur (même principe que pour l'utilisation du `this`). La référence à un constructeur est désignée par `super`.

Construction et initialisation des objets dérivés

```
public class Point
{
  private int x,y;

  public Point(int x,int y)
  {
    this.x = x;
    this.y = y;
  }
  ...
}

public class PointCouleur extends Point
{
  public PointCouleur(int x,int y,byte couleur)
  {
    super(x,y); //appel du constructeur de la classe Point
    this.couleur = couleur;
  }
  ...
}
```

Attention.

- Les mots clés **this** et **super** ne peuvent pas être utilisés en même temps.
- Lorsqu'une classe dérive d'une classe qui dérive elle aussi d'une autre classe, l'appel par **super** ne concerne que le constructeur de la classe de base de niveau immédiatement supérieur.

Construction et initialisation des objets dérivés

Remarques importantes sur la définition d'un constructeur.

- Si la super-classe ne possède pas de constructeur, il est possible d'appeler le constructeur par défaut à partir de la classe dérivée via `super();`. Cet appel peut paraître superflu, mais ne nuit pas. Ceci est pratique lorsque l'on construit une classe dérivée sans connaître les détails de la classe de base : on s'assure que les différents éléments de la super-classe seront correctement initialisés. Ceci justifie également le principe de toujours mettre un constructeur par défaut sans argument dans une classe.
- Si la classe dérivée ne possède pas de constructeur, le constructeur par défaut de la classe sera appelée et par conséquent le constructeur par défaut de la super-classe. Si ce constructeur n'existe pas (s'il y a que des constructeurs avec arguments qui sont définis), il y a une erreur de compilation. Par exemple ;

Construction et initialisation des objets dérivés

```
class A
{
  public A(int n)
  { ... }
  ...
}

class B extends A
{
  // pas de constructeur
}
```

Cet exemple provoque une **erreur** car il n'y a pas de constructeur sans argument dans `A`.

Construction et initialisation des objets dérivés

La construction d'un objet `B` dérivé d'un objet `A` entraîne 6 étapes :

1. Allocation mémoire pour un objet de type `B` (y compris les champs définis dans la super-classe `A`).
2. Initialisation par défaut de tous les champs de `B` (aussi bien ceux hérités de `A` que ceux définis dans `B`) aux valeurs nulles classiques.
3. Initialisation explicite des champs hérités de `A`.
4. Exécution du corps du constructeur de `A`.
5. Initialisation explicite des champs propres à `B`.
6. Exécution du constructeur de `B`.

Tester