2. Héritage et constructeurs

- Si un constructeur d'une classe dérivée n'appelle pas le constructeur de sa classe de base, c'est le constructeur sans argument de celle-ci qui sera d'abord exécuté, puis celui de la classe dérivée ;

Règle 1

Exemple

```
class Mere
{ public Mere()
  { System.out.println ("Constructeur mère") ; }
}
class Fille extends Mere
{
  public Fille()
  { System.out.println ("Constructeur fille") ; }
}
public class TestFille
{
  public static void main (String args[])
  { Fille a = new Fille() ; }
}
```

Qu'affichera l'exécution du programme ?

2. Héritage et constructeurs

Règle 2

- Si un constructeur d'une classe dérivée appelle un constructeur de sa classe de base, il doit obligatoirement s'agir de la première instruction du constructeur et ce dernier est désigné par le mot-clé **super** ;

Exemple 1

```
class Mere
{   private String prenom ;
    public Mere(String prenom)
    { this.prenom = prenom ; }
}
class Fille extends Mere
{
  public Fille(String nom)
  { super (nom) ;
    System.out.println ("Constructeur fille") ; }
}
public class TestFille
{
  public static void main (String args[])
  { Fille a = new Fille("Céline") ; }
}
```

Exemple 2

On dispose de la classe suivante :

```
class Point
{ public Point (int x, int y)    this.x = x ; this.y = y ;
  public void affCoord()
    System.out.println ("Coordonnees : " + x + " " + y) ;
  private int x, y ;
}
```

Ecrire le code de la classe PointNom, dérivée de Point, qui disposera d'un constructeur pour définir les coordonnées et le nom d'un objet de type PointNom.

Solution

```
class PointNom extends Point
{
    public PointNom (int x, int y, char nom)
    {
        super (x, y) ;
        this.nom = nom ;
    }
    public void affCoordNom()
    {
        System.out.print ("Point de nom " + nom + " ") ;
        affCoord() ;
    }
    private char nom ;
}
```