Copie de Montre

```
Montre maMontre = new Montre(...);

// votre ami veut la même montre
// et dans ce cas on sait qu'il ne faut pas juste affecter
// les références
Montre montreToto = new Montre(maMontre);
```

Copie profonde

```
class Montre extends Produit {
   private Mecanisme coeur;
   private ArrayList<Accessoire> accessoires;
   //...
}
```

Si l'on veut faire des copies de Montres, on **doit** ici faire une **copie profonde** :
copie de chaque constituant (mécanisme, accessoires)

Constructeur de copie

Copie de surface :

```
public Montre(Montre autre) {
   super(autre);
   coeur = autre.coeur;
   accessoires = autre.accessoires;
}
```

Mais... comment copier chaque élément en tant que tel ?
(et non pas comme une instance des super-classes Mecanisme et Accessoire)

☞ **copie polymorphique**

Copie polymorphique

Pourquoi ceci ne suffit-il pas ?

```
public Montre(Montre autre)
{   super(autre);
   coeur = new Mecanisme(autre.coeur);
   accessoires = new ArrayList<Accessoire>();
   for (Accessoire acc : autre.accessoires) {
      accessoires.add(new Accessoire(acc));
   }
}
```

Copie polymorphique

```
public Montre(Montre autre) {
 super(autre);
 coeur = autre.coeur.copie();
 accessoires = new ArrayList<Accessoire>();
 for (Accessoire acc : autre.accessoires) {
  accessoires.add(acc.copie());
 }
}
```

Copie polymorphique

```
  ArrayList<Accessoire> accessoires;
  // ...
  accessoires.add(acc.copie());
```

```
abstract class Accessoire extends Produit {
   //...
    // copie polymorphique d'Accessoire
   public abstract Accessoire copie();
    //..
}
//-----------------------------------------------
class Bracelet extends Accessoire {
   //..
   public Bracelet(Bracelet autre) { super(autre); }
   // copie polymorphique de Bracelet
   @Override
   public Bracelet copie(){
      return new Bracelet(this);
   }
}
```

Autre copie

```
class Montre extends Produit {
   //...
   public Montre(Mecanisme depart)
   {
      coeur = depart.copie();
      accessoires = new ArrayList<Accessoire>();
   }
//...
}
```

La méthode `clone`

Les traitements de copie, tels que ceux montrés précedemment, sont usuellement fait en java au moyen de la redéfinition de la méthode `clone` héritée de `Object`

Les classes redéfinissant `clone` doivent implémenter l'interface `Cloneable`.

Test : un exemple de `main()`

```
public static void main(String[] args) {
   // le reste comme avant

   // Nous faisons une copie de la montre m
   Montre m2 = new Montre(m);
   System.out.println("Montre m2 :");
   m2.afficher();
}
```

Le code complet à ce stade (415 lignes) peut être téléchargé sur le site du cours :

`https://d396qusza40orc.cloudfront.net/java-fr/complements/Montres.java`