Polymorphisme

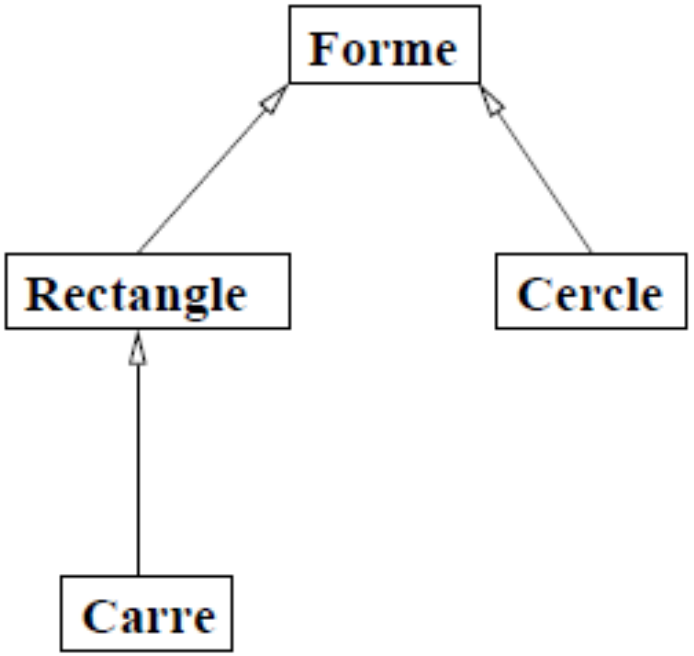
Rappel

- La différence entre surcharge et redéfinition

Définition

- Le polymorphisme est la faculté attribuée à un objet d'être une instance de plusieurs classes.
- Il a une seule classe "réelle" qui est celle dont le constructeur a été appelé en premier (c'est-à-dire la classe figurant après le new) mais il peut aussi être déclaré avec une classe supérieure à sa classe réelle.

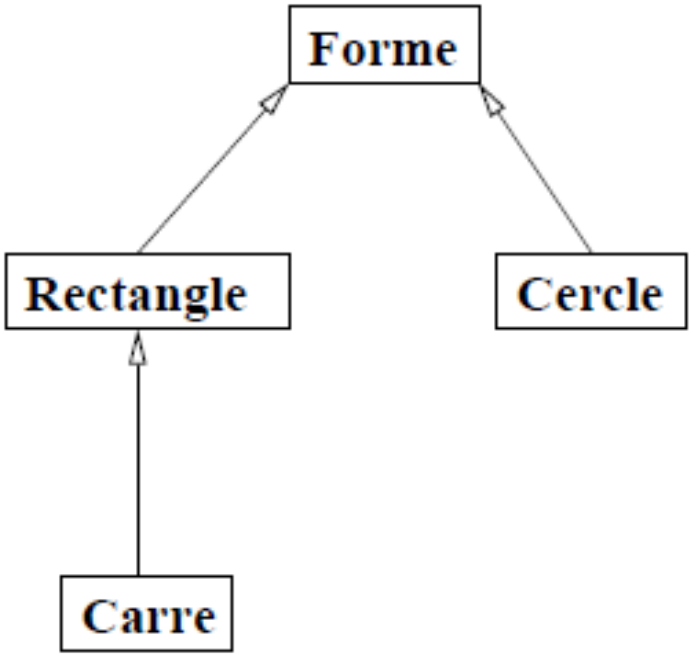
Exemple




```
public class Forme {
private int origine_x ;
private int origine_y ;
public Forme() {
    this.origine_x = 0 ;
    this.origine_y = 0 ;
}
public void affiche(){
            System.out.println("x="+x+ " et  y="+y);}
}
```

Exemple

```
public class Forme {
private int origine_x ;
private int origine_y ;
public Forme() {
    this.origine_x = 0 ;
    this.origine_y = 0 ;
}
public void affiche(){
            System.out.println("x="+x+ " et
y="+y);
}
}
```

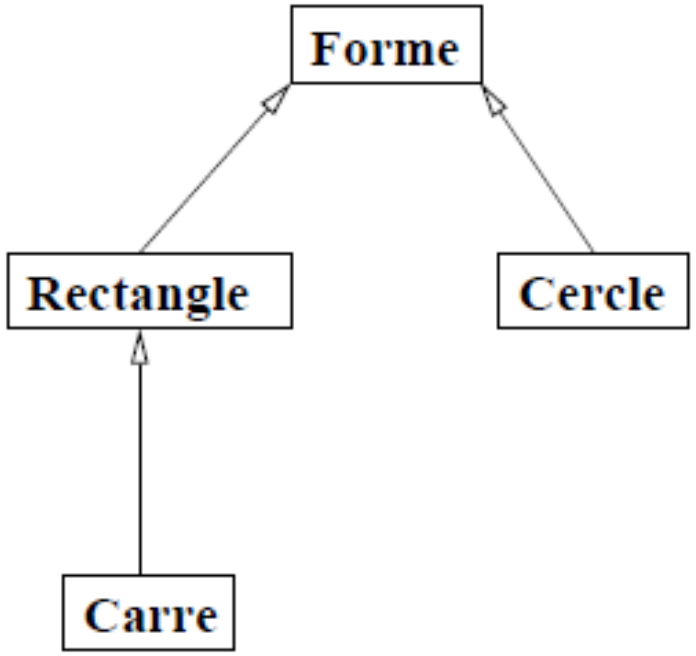



```
public class Rectangle extends Forme {
private int largeur ;
private int longueur ;
public Rectangle(int x, int y) {
        this.largeur = x ;
        this.longueur = y ;
}
public void affiche() {
System.out.println("rectangle " + longueur + "x"
+ largeur);
}}
```

```
public class Carre extends Rectangle {
public Carre(int cote) {
            super(cote, cote);
}
public void affiche() {
            System.out.println("carré " +
            this.getLongueur());
}
}
```

Exemple

- Cette propriété est très utile pour la création d'ensembles regroupant des objets de classes différentes comme dans l'exemple suivant :



Forme[] tableau = **new Forme[4]**;

tableau[0] = **new** Rectangle(10,20);

tableau[1] = **new** Cercle(15);

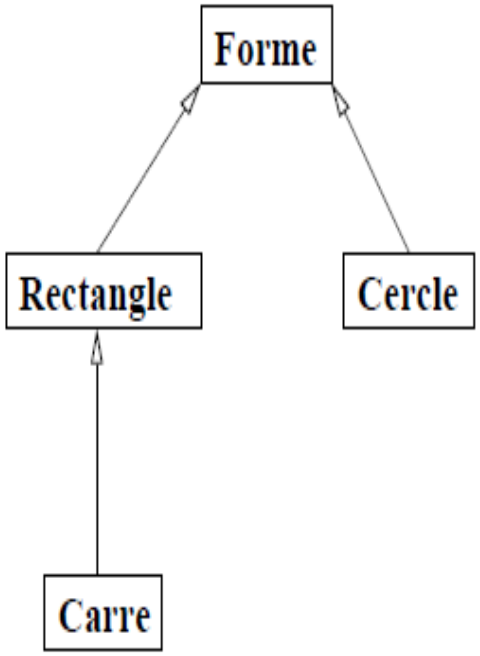
tableau[2] = **new** Rectangle(5,30);

tableau[3] = **new** Carre(10);

```
Forme[] tableau = new Forme[4];
tableau[0] = new Rectangle(10,20);
tableau[1] = new Cercle(15);
tableau[2] = new Rectangle(5,30);
tableau[3] = new Carre(10);
```

L'opérateur **instanceof** peut être utilisé pour tester l'appartenance à une classe comme suit :

```
for (int i = 0 ; i < tableau.length ; i++) {
        if (tableau[i] instanceof Forme)
        System.out.println("element " + i + " est une forme");
        if (tableau[i] instanceof Cercle)
        System.out.println("element " + i + " est un cercle");
        if (tableau[i] instanceof Rectangle)
        System.out.println("element " + i + " est un rectangle");
        if (tableau[i] instanceof Carre)
        System.out.println("element " + i + " est un carré");
}
```



L'exécution de ce code sur le tableau précédent affiche le texte suivant :

element[0] est une forme
element[0] est un rectangle
element[1] est une forme
element[1] est un cercle
element[2] est une forme
element[2] est un rectangle
element[3] est une forme
element[3] est un rectangle
element[3] est un carré

Explication

- L'ensemble des classes Java, forme une hiérarchie avec une racine unique. Cette racine est la classe Object dont hérite toute autre classe.
- Une des propriétés induites par le polymorphisme est que l'interpréteur Java est capable de trouver le traitement à effectuer lors de l'appel d'une méthode sur un objet. Ainsi, pour plusieurs objets déclarés sous la même classe (mais n'ayant pas la même classe réelle), le traitement associé à une méthode donné peut être différent. Si cette méthode est redéfinie par la classe réelle d'un objet (ou par une classe située entre la classe réelle et la classe de déclaration), **le traitement effectué est celui défini dans la classe la plus spécifique de l'objet et qui redéfinie la méthode**.

- Dans notre exemple, la méthode affiche() est redéfinie dans toutes les sous-classes de Forme et les traitements effectués sont :

```
for (int i = 0 ; i < tableau.length ; i++) {
tableau[i].affiche();
}
```

Le résultat:

rectangle 10x20

cercle 15

rectangle 5x30

carré 10

- Si la méthode afficher() n'est pas définie dans la classe Forme, ce code ne pourra cependant pas être compilé. En effet, la fonction affiche() est appelée sur des objets dont la classe déclarée est Forme mais celle-ci ne contient aucune fonction appelée affiche() (elle est seulement définie dans ses sous-classes).
- Pour compiler ce programme, il faut transformer la classe Forme en une interface ou une classe abstraite tel que cela est fait dans les sections suivantes.

Merci