# RÉVISION MODULE POO

# Classe et objets

| |
|---|
| Nom de la classe |
| Exemple: Rectangle, cercle, voiture…… |
| Propriété de la classe |
| Exemple: longueur, largeur, poids, couleur…. |
| Méthode de la classe: (fonctions) |
| Constructeurs, calcule de périmètre…etc |

**Pour créer un objet:**

**Nom de classe nom d'objet= new nom de la classe()**

**Exemple:**

**Rectangle R1= new Rectangle()**

# Les constructeur:

- Des méthodes publiques pour créer les objets.
- Dans une même classe on peut définir plusieurs constructeurs.
- Le constructeurs par défaut et explicitement ajouté par le compilateur.

```
Public   Rectangle() {
}
```

# Encapsulation

Les attributs et les méthodes sont:

| Modificateur | Classe | paquetage | Sous-classe | Autre classe |
|---|---|---|---|---|
| Private | visible | | | |
| Aucun | visible | visible | | |
| Protected | visible | visible | visible | |
| Public | visible | visible | visible | visible |

# Getters / setters

Les Getters sont des méthodes publiques pour renvoyer la valeurs d'attribut privé:

```
Public float get_longueur(){
return longueur;
}
```

Les Setters sont des méthodes publiques pour modifier la valeurs d'attribut privé:

```
Public void Set_longueur(float longueur){
this.longueur= longueur;
}
```

```
public class Personne{
private string Nom;
private int Age;
public void Set_Nom(string Nom){
this.Nom=Nom;
public string Get_Nom(){
return Nom;
}
public void Set_Age(int Age){
this.Age=Age;
}
public int Get_Age(){
return Age;
}
```

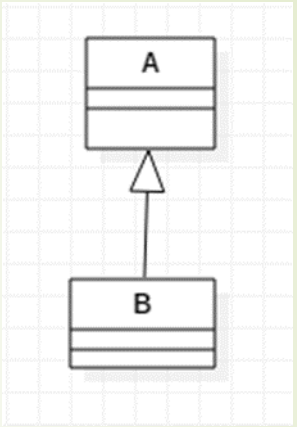
```
main (class){
Personne P = new Personne();
P.Nom= "Benahmed ali";
P.Age= 18;
}

main (hors la classe){
Personne P= new Personne();
P.set_Nom("benazi med");
P.Set_Age(18);
S.O.P("le nom:"+ P.get_Nom());
}
```

```
public class encapsulation{  voir la série ex3
public int i1=10;
private int i2=20;
protected int i3=30;
private void m2(){
i2=i2+10;
}
public int Get_i2(){
return i2;
}
public void set_i2(int i2){
this.i2=i2;
}}
```

```
public class Livre{
private int Id;
private String Titre;
Private String Auteur;
private float Prix;
public static int compteur=0;
Public Livre (){
this.id=++compteur; // pour le par défaut;
}
public Livre(String Titre, String Auteur, float prix){
this.id=++compteur;
this.titre=titre;
this.auteur=auteur;
this.prix=prix;
}
main(){
Livre L1= new Livre("POO","Benahmed Ali", 800);=> id=1;
```

# Héritage

- Une classe mère a des classe filles:
- La classe fille ou sous-classe a une seule classe mère.
- Dans cet exemple A est la classe mère de B
- B est la clase fille de A




Pour créer une classe fille on utilise le mot clé **extends**

```
public class UneClasseFille extends UneClasseMere{
...
}
```

Exemple:

```
public class EtudiantsSportif extends Etudiant{
...
}
```

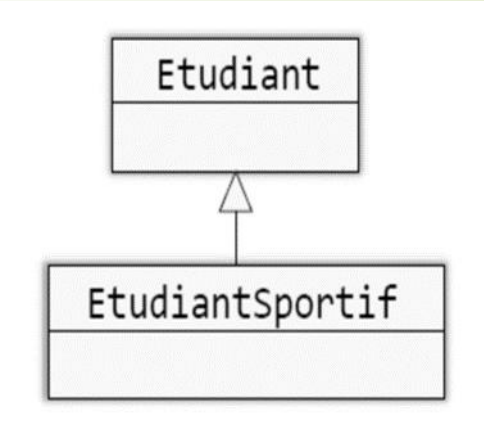
# Héritage

**Si on a une classe B dérivée de la classe A alors:**

- ✓ tout objet instance de la classe B peut être aussi vu comme une instance de la classe A. Cette relation est directement supportée par le langage JAVA

on a que *EtudiantSportif* est une classe dérivée de la classe *Etudiant* alors en Java on peut écrire:

```
Etudiant e;
e= new EtudiantSportif(…);
```



# Héritage

- **Exemple:** soit la hiérarchie des classes suivantes, On a:

```
A a; B b; C c;
a =new B();☑
a =new F(); ☑
b= new A(); ☒
b= new F(); ☑
c=new E(); ☒
c=new F(); ☑
c=new A(); ☒
c=new B(); ☒
```