# Cours.

# SLAM4 – Programmation Objet



Thème 3 – Classe abstraite.

Une classe abstraite est une classe qui a été déclarée avec le mot-clé « abstract ». Dans le diagramme de classe, son nom est imprimé en italique.

On ne peut **pas créer d'instances** d'une classe abstraite, et une telle classe doit forcément **être héritée**.



En règle générale, une classe abstraite contient une ou plusieurs méthode abstraites, mais ce n'est pas obligatoire : une classe peut être déclarée abstraite sans pour autant posséder de méthodes abstraites.

Une méthode abstraite est une méthode déclarée avec le mot-clé « abstract » et qui n'a pas de corps. Par exemple :

```
public  abstract  class A {  // déclarée dans le fichier A.java

     public  abstract  void ouSuisJe() ;  // pas de corps de méthode

}
```

On peut déclarer autant de méthodes abstraites que l'on veut dans une classe abstraite, y compris ne pas en déclarer du tout. Une classe abstraite devant forcément être dérivée, la (ou les) fille(s) doit (ou doivent) **obligatoirement** définir la méthode abstraite, c'est-à-dire, lui donner un corps. Par exemple :

```
public  class B  extends A {  // déclarée dans le fichier B.java

     public  void ouSuisJe() {
        System.out.println("J'y suis j'y reste !") ;
    }

}
```

D'une façon générale :

- une classe abstraite peut étendre une autre classe abstraite ;
- une classe abstraite peut étendre une classe concrète ;
- une classe concrète qui étend une ou plusieurs classes abstraites (indirectement), doit obligatoirement fournir une implémentation pour toutes les méthodes abstraites existantes.

```
public  abstract  class A {
     public  abstract  void ouSuisJe() ;
}

 public  abstract  class B  extends A {
     public  void ouSuisJe() {
        System.out.println("Il est passé par ici") ;
    }
}

 public  abstract  class C  extends B {
     public  abstract  void ouVasTu() ;
}

 public  class D  extends C {
     public  void ouVasTu() {
        System.out.println("J'y retourne") ;
    }
}
```



Dans cet exemple :

- on ne peut pas instancier A, qui est abstraite ;
- on ne peut pas non plus instancier B, car même si elle n'a pas de méthode abstraite, elle est tout de même déclarée abstraite ;
- on ne peut pas instancier C ;
- on peut instancier D, qui possède bien une implémentation pour ouSuisJe() (par extension de B), et pour ouVasTu().

**Pour s'entrainer avec les classes abstraites** : écrire en C# et en Java : (source : Wikipédia !)

**En Java** [ modifier | modifier le code ]

```
public abstract class Forme {
   public abstract float aire() ;
}

public class Carre extends Forme {
   private float cote;
@override
   public float aire() {
       return cote * cote;
   }
}

public class Cercle extends Forme {
   private float rayon;
@override
   public float aire() {
       return (float) Math.PI*rayon*rayon;
   }
}
```

**En C#** [ modifier | modifier le code ]

```
public abstract class Forme
{
    public abstract float Aire();
}

public class Carre : Forme
{
    private float cote;

    public override float Aire()
    {
        return (float) Math.Pow(cote, 2);
    }
}

public class Cercle : Forme
{
    private float rayon;

    public override float Aire()
    {
        return (float) ( Math.PI * Math.Pow(rayon, 2) );
    }
}
```



Et même en Python !

**En Python** [ modifier | modifier le code ]

```
import math


class Forme:
    def aire(self):
        raise NotImplementedError()


class Carre(Forme):
    def __init__(self, cote):
        self.cote = cote

    def aire(self):
        return self.cote * self.cote


class Cercle(Forme):
    def __init__(self, rayon):
        self.rayon = rayon

    def aire(self):
        return math.pi * self.rayon * self.rayon
```

Pour s'exercer avec les classes abstraites : réaliser le jeu décrit ici :
https://openclassrooms.com/fr/courses/2818931-programmez-en-oriente-objet-avec-c/2818991-tp-creez-un-petit-jeu-oriente-objet

On développera une classe abstraite « Monstre » avec une méthode abstraite « attaque » et les classes décrites dans l'énoncé.