

Leçon 5 : Classe abstraite et interface

Université Virtuelle de Côte d'Ivoire

Table des matières



I - 1- Classe abstraite	3
II - Application 1 :	4
III - 2- Interface	5
IV - Application 2 :	7
Solutions des exercices	8

1- Classe abstraite

I

1.1- Définition

Une classe abstraite est une classe incomplète. Elle regroupe un ensemble de attributs et de méthodes mais **certaines de ses méthodes ne contiennent pas d'instructions**, elles devront être définies dans **une classe héritant de cette classe abstraite**.

Quand on ne peut pas écrire d'implémentation pour une méthode donnée, **cette méthode est qualifiée d'abstraite**. Cela signifie que l'on laisse le soin aux sous-classes d'implémenter cette méthode.

En java, c'est le mot clef **abstract** qui permet de qualifier d'abstraite une classe ou une méthode

1.2- Syntaxe

```
public abstract class nomclass{ }
```

NB : Une classe abstraite ne peut pas être instanciée, on ne peut créer d'objet à partir d'une classe abstraite.

Exemple :

```
1 public abstract class Forme {  
2     public Forme() { // Le constructeur  
3  
4 }  
5     abstract void perimetre();  
6     abstract void aire();  
7  
8 }
```

Application 1 :



II

Exercice

[solution n°1 p.8]

Énoncé 1 :

Peut-on instancier une classe abstraite ?

☐ oui

☐ non

Dans certains

☐ cas

Exercice

[solution n°2 p.8]

Énoncé 2 :

- Créer une classe abstraite animal
- Créer des attributs couleur et poids en protected
- Créer les méthodes manger,boire en protected
- Créer les méthodes déplacement et crier comme des méthodes abstraites

NB : Respecter l'ordre de créations des attributs et méthodes

Solution :

```
public abstract class Animal {  
    protected String couleur;  
    protected int poids;  
    protected void manger()  
    {  
        System.out.println("Je mange de la viande.");  
    }  
    protected void boire()  
    {  
        System.out.println("Je bois de l'eau !");  
    }  
    protected void seDeplacer()  
    {  
    }  
    protected void crier()  
    {  
    }  
}
```

2- Interface



2.1- Définition

Une interface définit un comportement (d'une classe) qui doit être implémenté par une classe, sans implémenter ce comportement. C'est un ensemble de méthodes abstraites, et de constantes.

2.2- Syntaxe

- **Interface :**

```
public interface nominterface{
<Liste des méthodes>
}
```

- **Implémentation :**

```
public class nomclass implements nominterface{
<Bloc Instruction>
}
```

Exemple :

```
1 //-----Création de l'interface vehicule
2 public interface Vehicule {
3
4     void rouler();
5
6     void freiner();
7
8 }
9 //*****Implémentation de l'interface
10 public class Auto implements Vehicule {
11     //Champs
12
13     private String marque;
14     private int poids;
15
16     //Constructeurs
17
18     public Auto(String marque, int poids)
19     {
20         this.marque = marque;
21         this.poids = poids;
22     }
23 }
```

```
24 //Methodes
25
26 public void rouler() {
27     System.out.println("Je roule !!!");
28 }
29
30 public void freiner() {
31     System.out.println("Je freine !!!");
32 }
33
34 }
```

Application 2 :

IV

Exercice

[solution n°3 p.9]

Énoncé 1 :

Peut-on instancier une interface ?

☐ non

Dans certains

☐ cas

☐ oui

Exercice

[solution n°4 p.9]

Énoncé 2 :

- Créer un interface PeriAire avec les méthodes perimetre et aire ;
- Créer ensuite une classe carre avec pour attribut cote qui implémentera l'interface ci-dessus ;

Solution :

```

interface PeriAire {
    double perimetre();
    double aire();
}

class Carre implements PeriAire {
    private double cote;

    Carre(double cote) {
        this.cote = cote;
    }

    double perimetre() {
        return 4 * cote;
    }

    double aire() {
        return cote * cote;
    }
}

```

Solutions des exercices



> Solution n°1

Exercice p. 4

Énoncé 1 :

Peut-on instancier une classe abstraite ?

☐ oui

☒ non

Dans certains

☐ cas

> **Solution n°2**

Exercice p. 4

Énoncé 2 :

- Créer une classe abstraite animal
- Créer des attributs couleur et poids en protected
- Créer les méthodes manger,boire en protected
- Créer les méthodes déplacement et crier comme des méthodes abstraites

NB : Respecter l'ordre de créations des attributs et méthodes

Solution :

```
public abstract class animal {
protected String couleur;
protected int poids;
protected void manger() {
System.out.println("Je mange de la viande.");
}
protected void boire() {
System.out.println("Je bois de l'eau !");
}
abstract void déplacement();
abstract void crier();
}
```

> **Solution n°3**

Exercice p. 7

Énoncé 1 :

Peut-on instancier une interface ?

☒ non

Dans certains

☐ cas

☐ oui

> **Solution n°4**

Exercice p. 7

Énoncé 2 :

- Créer un interface `PeriAire` avec les méthodes `perimetre` et `aire` ;
- Créer ensuite une classe `carre` avec pour attribut `cote` qui implémentera l'interface ci-dessus ;

Solution :

```
public interface Periaire {  
    double perimetre();  
    double aire();  
}  
  
public class carre implements Periaire {  
    double cote;  
  
    public double perimetre(){  
        return cote*4;  
    }  
  
    public double aire(){  
        return cote*cote;  
    }  
}
```