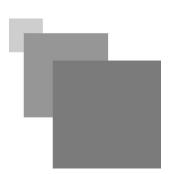
Leçon 5 : Classe abstraite et interface

Université Virtuelle de Côte d'Ivoire





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

1- Classe abstraite



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:



Exercice [solution n°1 p.8]

Énoncé 1:

Peut-on instancier une classe abstraite?

- O oui
- O non

Dans certains

O cas

Exercice [solution n°2 p.8]

Énoncé 2:

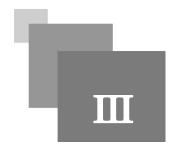
- Créer une classe abstraite animal
- Créer des attributs couleur et poids en protected
- Créer les méthodes manger, boire en protected
- Créer les méthodes deplacement et crier comme des méthodes abstraites

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

Solution:

			{
System.out.pri	ntln("Je mange o	de la viande.");	
}			
			{
System.out.pri	ntln("Je bois de	l'eau !");	
}			
}			

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

```
//Methodes

public void rouler() {

System.out.println("Je roule !!!");

public void freiner() {

System.out.println("Je freine !!!");

}

System.out.println("Je freine !!!");

}
```

Application 2:



Exercice [solution n°3 p.9]

Énoncé 1 :

Peut-on instancier une interface?

O non

Dans certains

O cas

O 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:

Solutions des exercices



> **Solution** n°1

Énoncé 1:

Peut-on instancier une classe abstraite?

- O oui
- o non

Dans certains

O cas

> **Solution** n°2

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 deplacement 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 deplacement();
  abstract void crier();
}
```

> **Solution** n°3

Exercice p. 7

Énoncé 1:

Peut-on instancier une interface ?

o non

Dans certains

- O cas
- O 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;
  }
}
```