TD 5 — Classes abstraites et interfaces

Objectif(s)

- ✯ compréhension du fonctionnement des classes abstraites et des interfaces
- ✯ ainsi que leurs différences

Exercice 1 – Les classes abstraites

Les moyens de transport tels qu'une voiture, un bateau ou un train sont des véhicules capables de se déplacer jusqu'à une certaine vitesse maximale.

Question 1

Exprimer ce concept abstrait de véhicule à l'aide d'une classe abstraite `Vehicule` possédant un attribut `nom` donnant son nom (type `String`), une méthode `getNom()`, une méthode `setNom(String s)`, une méthode abstraite `getCategorie()` retournant une chaîne de caractère (String) correspond à la catégorie du véhicule (bateau à rame ou voiture dans la suite) et une méthode abstraite retournant un `float` donnant sa vitesse maximum nommée `getVitesseMaximum()`.

La classe `Vehicule` possède également une méthode `estVitessePossible(float)` retournant un booléen vrai si et seulement si son argument est une vitesse acceptable pour ce véhicule. Définissez-la.

Question 2

Créer ensuite deux classes dérivées nommées `BateauARames` et `Voiture` qui implémentent uniquement un constructeur paramétré et les deux méthodes abstraites de la super-classe. Le bateau à rames possède un attribut donnant son nombre rameurs (la vitesse maximum est égale à 0.5 fois ce nombre) et la voiture un attribut donnant sa puissance (la vitesse maximum est égale à 1.5 fois ce nombre).

Question 3

Écrire une classe illustrant l'utilisation de ces 3 classes en créant un tableau de `Vehicule`, rempli avec un ensemble de bateaux et de voitures avec des caractéristiques différentes, et en affichant du nom, de la catégorie et de la vitesse maximale des éléments du tableau. Écrire une méthode qui teste si la vitesse de 15 Km/h est possible pour tous ces véhicules.

Exercice 2 – Que fait ce programme ?

Expliquer ce que fait le code suivant et donner une trace de son exécution. S'il y a une ou des erreur(s) dire pourquoi et corriger la(es).

```
interface InterfaceValeur {
    public int valeur();
}
class A implements InterfaceValeur {
    private int nombre;
    A(int n) {
        this.nombre=n;
    }
    int valeur() {
        return nombre;
    }
    public String toString() {
        return "("+nombre+")";
    }
```

```
    public int valeurBis() {
        return 5*valeur();
    }
}
class B implements InterfaceValeur {
    private String couleur;
    B(String c) {
        this.couleur=c;
    }
    public int valeur() {
        if (couleur.equals("rouge")) return 1;
        else return -1;
    }
    public String toString() {
        return "("+couleur+")";
    }
    public int valeurBis() {
        return 2*valeur();
    }
}
class Test {
    public static void main(String args[]) {
        InterfaceValeur[] tab=new InterfaceValeur[2];
        tab[0]=new B("rouge");
        tab[1]=new A(5);
        for(int i=0;i<tab.length;i++) {
            System.out.println(tab[i]);
            System.out.println(tab[i].valeur());
            System.out.println(tab[i].valeurBis());
        }
    }
}
```

Exercice 3 – Animaux

On veut modéliser l'héritage chez les animaux et les mammifères. Créer une classe mère `Animal` avec :

— Un constructeur prenant comme paramètre le nom de l'animal,

Créer une classe fille de `Animal` nommée `Mammifere`, avec :

— Un constructeur prenant comme paramètre le nom du mammifère,

— Une méthode `toString()` redéfinie retournant une chaîne de caractère du type "Je suis un animal de nom ... . Je suis un mammifère.".

Créer une interface nommée `Deplacement` qui contient une méthode sans argument ni type de retour qui se nomme `seDeplacer`. Créez également une interface nommée `Raisonnement` qui contient une méthode sans argument ni type de retour qui se nomme `reflechir`.

Créer 2 classes filles de `Mammifere` nommées `Chien` et `Homme` qui implémentent l'interface `Deplacement`. La classe `Homme` implémente en plus l'interface `Raisonnement`. Vous remplirez le contenu des méthodes d'interface pour obtenir les affichages suivant lorsque les méthodes des deux interfaces sont appelées :

— Pour un chien : " Je me déplace à quatre pattes en remuant la queue ".

— Pour un homme : " Je me déplace debout sur mes deux jambes "

— Pour un homme : " Je suis en en pleine réflexion ".

De plus :

— Pour la classe fille `Chien`, une méthode `toString` redéfinie retournera une chaîne de caractère du type "Je suis un animal de nom ... . Je suis un mammifère. Je suis un chien."

— Pour la classe fille `Homme`, une méthode `toString` redéfinie retournera une chaîne de caractère du type "Je suis un animal de nom ... . Je suis un mammifère. Je suis un homme.".

Réaliser un tableau de type `Animal` contenant un mammifère, un chien et un homme (pour ces derniers, vous choisirez les valeurs d'initialisation). Réalisez ensuite une boucle exploitant les méthodes `toString()`, `seDeplacer()` et `reflechir()` en vérifiant à chaque fois que cela est possible.

Réaliser ensuite deux tableaux nommée `dep` et `rai` avec les types suivants : `Deplacement[]` et `Raisonnement[]`, puis remplissez les avec des objets de types appropriés.

Est-il interessant de définir les classes `Animal` et `Mammifere` comme classe abstraite ?