Mis-a-niveau-JAVA

Hossein Khani

Dauphine
UNIVERSITÉ PARIS

Exercice

- Considérons cinq classes A, B, C, D et E tel que A, B et C sont dans le même package package1 et les classes D et E sont mis dans un autre package package2. Les classes B et D héritent de la classe A.

```java
package package1;
public class A {
  private int champPrive;
  int champSansModificateur;
  protected int champProtege;
  public int champPublique;
}
```

- Complétez le tableau ci-dessous en cochant les cases pour lesquelles les variables d'instance de la classe A sont visibles.

	Classe A	Classe B	Classe C	Classe D	Classe E
champPrive					
champSansModificateur					
champProtege					
champPublique					

- Si la classe A n'était pas déclarée public, est ce que cela change la visibilité des variables?

Exercice

Analyse en Papier

Considérons le programme Java suivant :

```
package Ex1;
public class Livre {
        protected String titre, auteur, proprietaire ;
        protected int nb_page ;
        double prix ;

        public Livre(String t, String a, double p, int nb){
         titre = t ;
         auteur = a ;
         prix = p ;
         proprietaire = "" ;
         nb_page = nb ;
         }
        public void Afficher() {
         System.out.println("Titre : " + titre) ;
         System.out.println("Auteur : " + auteur) ;
         System.out.println("Prix : " + prix) ;
         System.out.println("Nombre de pages : " + nb_page);
         if ( this.Est_neuf() ) {
         System.out.println("Aucune proprietaire" ) ;
         } else {
         System.out.println("Proprietaire: "+proprietaire);
         }
         System.out.println() ;
         }
        public boolean Est_neuf() {
         if ( proprietaire =="" ) return true ;
         else return false ;
         }
        public void Acheter(String nom) {
         proprietaire = nom ;
         }
    }
```

Exercice

```
public class BD extends Livre {
    private boolean encouleur ;
    public BD(String t,String a,double p,int nb, boolean c){
     super(t,a,p,nb) ;
     encouleur = c ;
     }
    }

public class Album extends Livre {
    boolean page_coloree[];
    public Album(String t, String a, double p, int n){
     super(t,a,p,n) ;
     page_coloree = new boolean[n];
     int i ;
     for (i=0 ; i<100 ; i++)
     page_coloree[i] = false ;
     }
    public void Colore(int num_page){
     if((page_coloree[num_page] == false) && !Est_neuf()){
     page_coloree[num_page] = true ;
     } else {
     System.out.println("page deja coloriee" ) ;
     }
     }}

public class Test {
public static void main(String[] args) {

 Livre l1 = new Livre("Le petit prince","St Exupéry",10.40, 50) ;
 Livre l2 = new Livre("Contes","Grimm",14.40,254) ;
 l1.Afficher() ;
 l1.Acheter("moi") ;
 l1.Afficher() ;
 l1.prix = 0.0 ;
 l2.Acheter("lui") ;
 l2.Afficher();

 BD b1 = new BD("Lucky Luke","Morris",10.40, 45, true);
 BD b2 = new BD("Tintin","Herge",200.40, 45, false) ;
 b1.Acheter("moi");
 b1.Afficher() ;
 b2.Afficher() ;

 Album a1 = new Album("Dora","Dora", 3.5,300) ;
 a1.Afficher() ;
 a1.Colore(23) ;
 a1.Acheter("moi");
 a1.Colore(23) ;
 }
}
```

Exercice

1. Expliquez les informations que le programme va afficher lors de son exécution.
2. Dans la classe Livre, l'attribut prix a été défini sans règle d'encapsulation (public, private...) Comment Java va-t-il l'interpréter ? Comment pouvez-vous le tester ou comment est-il testé dans le programme ?
3. Expliquez comment on teste dans ce programme si un livre est neuf ou non.
4. Décrivez la hiérarchie de classe décrite dans ce programme et expliquer le processus d'appel entre les constructeurs.
5. Expliquez comme ce programme gère le fait de colorer une page d'un album à colorer.

A vous

Le but de cet exercice est de créer une classe Complexe (z = a+ib). Cette classe va nous permettre de manipuler des nombres complexes.

- Déclarez la classe Complexe dans un fichier constituée de ses paramétres a et b, ainsi que de son constructeur par d´efaut.
- Implémentez le constructeur par défaut en attribuant à a et b la valeur 0.
- Ajoutez un constructeur permettant d'initialiser les valeurs de a et b (pouvant être différents de 0) en les passants comme paramétres.
- Définissez une fonction qui calcule la somme de deux nombres complexes.
- Définissez une fonction qui détermine le produit de deux nombres complexes.

Encapsulation

- Un des quatre principes de la programmation objet orienté.

  - Utiliser le modificateur d'accès private.

  - Setter: Public void setXXX(valeur)
  - Point d'accès pour modifier une variable d'objet.

  - Getter: public typeDeRetour getXXX()
  - Point d'accès pour lire une variable d'objet.

Interfaces

- Une interface est une collection d'opérations utilisée pour spécifier un service offert par une classe,
- C’est un contrat spécifié par une classe par le mot clé implements pour dire que l’on respecte cette interface,
- Toutes les méthodes sont abstraites (pas de corps),
- Elles sont implicitement publiques .

Exemple

- Class Person
- Class Car
- Interface ShowType

Utilisation d’interface

- On peut utiliser interface comme une référence aux objets des classes qui implémentent l'interface.

- ✓ Dans ce cas, les seules méthodes disponibles sont celles définies dans l'interface.

- On peut passer des objets d’interface comme paramètre à une méthode.

- ✓ Dans ce cas, les objets des classes qui implémentent l’interface sont disponible dans la méthode.

Static Method

```java
public class Dog {
    int age;
    static int numberOfInstances;
    public Dog(int age) {
        this.age = age;
        numberOfInstances++;
    }

    public static void display() {
        System.out.println("The number of instances are " + numberOfInstances);
    }

    public void about() {
        System.out.println("This dog is"+ this.age +".");
    }
}
```

```java
public class Main {

    public static void main(String[] args) {
    Dog.display();

    Dog dog1 = new Dog(3);
    dog1.about();

    Dog.display();

    }
}
```

Output:

The dog is 3.

The number of instances are 1

Static Methodes ne pouvent pas utiliser instance variable.

Instance Methodes peuvent utiliser static variable.

Utilisation Méthode Static

- La méthode est indépendant des objets.
- Un morceau de code particulier doit être partagé par toutes les méthodes d'instance.
- Vous écrivez des classes utilitaires qui ne doivent pas être modifiées.

Exercice

- Certain animaux peuvent crier, d'autres sont muets. On représentera le fait de crier au moyen d'une méthode affichant à l'écran le cri de l'animal.

1. Ecrivez une interface contenant la méthode permettant de crier.

2. Ecrivez les classes des chats, des chiens et des lapins (qui sont muets)

3. Ecrivez un programme avec un tableau pour les animaux qui savent crier, le remplissez avec des chiens et des chats, puis faire crier tous ces animaux. Décrivez ce qui s'affiche à l'écran à l'exécution de ce programme.

Exercice

- Pour ecrire un logiciel bancaire, il faudra représenter dans notre langage de programmation l'ensemble des informations caractérisant un compte et coder les actions qui s'effectuent sur les comptes (retrait, dépot). L' état d'un compte, pourra être défini par son numéro, le nom de son titulaire, son solde ; son comportement est caractérisé par les opérations de dépot, de retrait et d'affichage du solde.

1. Ecrivez la classe Compte().
2. Ecrivez une méthode afficher(), pour afficher le solde.
3. Codez les méthodes retirer() et déposer().

Exercice

- Considérons une classe Personne par la donnée des deux variables d'instances, l'une contenant la date de naissance de la personne et l'autre son nom. La date de naissance n'est pas un type pré défini. Il faut donc aussi définir une classe Date.

1. Ecrivez la classe Personne().
2. Ecrivez la classe Date().

Exercice

- On propose de pouvoir comparer des objets de différentes classes au moyen d'une conversion vers les nombres entiers. Pour cela on va utiliser une interface avec la méthode de conversion.

```
interface Convertible{
int toInt();
}
```

1. Modifiez les classes Compte() et Date() vues en cours pourqu'ells implémentent cette interface.

2. Ecrivez une classe proposant des méthodes statiques pour comparer deux objets convertibles : une pour le test plus grand strict, une pour le test plus petit strict, une pour le test d'égalité, en comparant les entiers obtenus par conversion.

3. Ajoutez à la classe précédente une méthode statique permettant de trier en ordre croissant un tableau d'objets convertibles.

Instanceof et Héritage

```java
public class Personnage { ... }
```

```java
public class Gaulois extends Personnage { ... }
```

```java
public class IrreductibleGaulois extends Gaulois { ... }
```

```java
public class Romain extends Personnage { ... }
...
  public static void main(String[] args){
    IrreductibleGaulois asterix = new IrreductibleGaulois();
    System.out.println( asterix instanceof Personnage);
    System.out.println( asterix instanceof Gaulois);
    System.out.println( asterix instanceof Romain);
```

Polymorphisme

```java
public class Machine {
    public void start() {
        System.out.println("Machine Start");
    }

    public void stop() {
        System.out.println("Machine Stop");
    }
}
```

```java
public class Car extends Machine {

    @Override
    public void start() {
        System.out.println("Car Start");
    }

    @Override
    public void stop() {
        System.out.println("Car Stop");

    }

}
```

```java
public class Main {

    public static void main(String[] args) {
        Machine machine1 = new Machine();
        machine1.start();
        machine1.stop();
    }
}
```

Output:

Machine Start

Machine Stop

Polymorphism

```java
public class Machine {
    public void start() {
        System.out.println("Machine Start");
    }

    public void stop() {
        System.out.println("Machine Stop");
    }

}
```

```java
public class Car extends Machine {

    @Override
    public void start() {
        System.out.println("Car Start");
    }

    @Override
    public void stop() {
        System.out.println("Car Stop");

    }

}
```

```java
public class Main {

    public static void main(String[] args) {
        Machine machine1 = new Car();
        machine1.start();
        machine1.stop();
    }
}
```

Output:

Car Start

Car Stop

Polymorphism

```java
public class Machine {
      public void start() {
          System.out.println("Machine Start");
      }

      public void stop() {
          System.out.println("Machine Stop");
      }

}
```

```java
public class Car extends Machine {

    @Override
    public void start() {
        System.out.println("Car Start");
    }

    @Override
    public void stop() {
        System.out.println("Car Stop");

    }

    public void windShield() {
        System.out.println("Windshield Start");
    }

}
```

```java
public class Main {

    public static void main(String[] args) {
        Machine machine1 = new Car();
        machine1.start();
        machine1.stop();
        machine1.windshield;
    }
}
```

Erreur de Compilation