UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Licence INFO
Info0201

Travaux dirigés n° 6

Héritage

Exercice 1 (Simulation de trafic routier)

1°) Dans un système de simulation de trafic routier on définit les éléments suivants : **autobus**, **autoroute**, **axesRoutiers**, **bicyclette**, **camion**, **camionCiterne**, **couleur**, **motocyclette**, **rouleAvecMoteur**, **rouleSansMoteur**, **route**, **traficRoutier**, **vehicule**, **vehicules**, **voiture**.

- Pour chaque élément, déterminez son type (classe, interface, paquetage ou énumération) : vous pourrez modifier les noms en ajoutant des majuscules, par exemple.
- Proposez un diagramme de classes simplifié décrivant les relations entre ces éléments.

2°) Voici un début de réalisation :

```
public enum Couleur {
  BLEU, BLANC, ROUGE, NOIR, GRIS
}
```

```
public class Vehicule{
    private static int numGlobal = 0;
    protected int numPerso;
    private String proprietaire = "inconnu";
    private Couleur couleur = Couleur.BLANC;

    public Vehicule(){
        numPerso = numGlobal;
        numGlobal ++;
        System.out.println(
            "Je suis un vehicule (numero "
            + numPerso + ").");
    }

    public Vehicule(String nom, Couleur couleur){
        this();
        proprietaire = nom;
        this.couleur = couleur;
    }

    public String toString(){
        return "\tVehicule " + numPerso
          + ", de proprietaire " + proprietaire
          + ", de couleur " + couleur;
    }
}
```

```
public class Voiture extends Vehicule{
    private int puissance = 0;

    public Voiture(){
        super();
        System.out.println(
                "Je suis une voiture");
    }

    public Voiture(String nom, Couleur couleur,
                int puissance){
        super(nom, couleur);
        this.puissance = puissance;
    }

    public String toString(){
        return super.toString()
            + ", de puissance "
            + this.puissance;
    }

    public void afficher(){
        System.out.println(super.toString());
    }
}
```

Donnez le diagramme de classes détaillé liant les classes **Vehicule** et **Voiture** et l'énumération **Couleur**.

3°) Décrivez, pas à pas, l'effet de chacune des instructions du programme de test suivant (donnez la représentation mémoire et précisez l'affichage) :

```
public class TestHeritage{

  public static void main(String[] args){
    Vehicule v1 = new Vehicule();
    Voiture v2 = new Voiture();
    System.out.println(v1.toString());
    System.out.println(v2.toString());

    v1 = v2;
    System.out.println(v1.toString());
    // v2 = v1;
    v2.afficher();
    // v1.afficher();

    // suite

    Vehicule[] v = new Vehicule[2];
    v[0] = new Vehicule("Martin", Couleur.BLANC);
    v[1] = new Voiture("Dupont",
                Couleur.ROUGE, 76);
    for(int i = 0; i < v.length; i++)
      System.out.println(v[i]);

    } // main

} // class TestHeritage
```

4°) Quels seraient les messages d'erreur engendrés par les deux instructions mises en commentaire ?

Exercice 2 (Keskesafait)

1°) On considère les trois classes suivantes. Aucune erreur ne s'est glissée dans celles-ci. Pour chacun des affichages de la méthode principale de la classe de test, précisez ses effets.

```
public class C1 {
  public C1() {
    // Rien a faire
  }

  public void affiche() {
    System.out.println("Ici C1");
  }
}
```

```
public class C2 extends C1 {
  public C2() {
    super();
  }

  public void affiche() {
    System.out.println("ici C2");
  }
}
```

```
public class Keskesafait {
  public void main(String[] d){
    C1 c11 = new C1();
    c11.affiche();
    C2 c22 = new C2();
    c22.affiche();
    C1 c12 = new C2();
    c12.affiche();
  }
}
```

2°) Même question avec les classes suivantes :

```
public class D1 {
  protected int n;

  public D1(int v) {
    this.n = v;
  }
  public D1() {
    System.out.println("?");
    n = 0;
  }

  public void affiche() {
    System.out.println("ici D1"
        + " n=" + n);
  }
  public void setN(int y) {
    this.n = y;
  }
  public void setVal(int u) {
    this.n = u;
  }
}
```

```
public class D2 extends D1 {
  private int p;

  public D2(int p) {
    super(0);
    this.p = p;
  }
  public D2(int n, int p) {
    this(p);
    super.setN(n);
  }
  public D2() {
    //  super();
    p = 1;
  }
  public void setVal(int u) {
    this.p = u;
  }
  public void affiche() {
    System.out.println("ici D2"
        + " p=" + p + ", n=" + n);
  }
}
```

```
public class KeskesafaitBis {
  public void main(String[] d){
    D1 d11 = new D1(3);
    D2 d22 = new D2(2,5);
    d22.affiche();
    d11.affiche();

    D1 d12 = new D2(1,2);
    d12.setN(-1);
    d12.affiche();

    d12.setVal(21);
    d12.affiche();
  }
}
```

Exercice 3 (Les bâtiments)

Nous envisageons une application destinée à effectuer le calcul de l'impôt foncier sur des bâtiments. Pour cela, nous réalisons les classes **Batiment**, **Commerce**, **Habitation**, **Industrie**, **Proprietaire**, **Locataire** et **Adresse**. Elles sont organisées dans des packages **batiments** et **personnes**. Le calcul de l'impôt d'un bâtiment repose sur sa superficie et dépend de la valeur d'un coefficient, liée à sa nature (maison particulière, commerce...). De plus, chaque bâtiment dispose systématiquement d'une adresse et d'un propriétaire, et les propriétaires doivent avoir une adresse (celle de leur habitation). Ils ne possèdent pas forcément de locataire.

1°) Proposez un diagramme de classes simplifié.

2°) Écrivez la classe **Batiment** : constructeur(s), *getter(s)* et *setter(s)* (quand ils ont un sens), conversion en chaîne de caractères, test d'égalité, ..., la méthode `calculerImpot()`.

3°) Écrivez les autres classes de l'application, en redéfinissant les méthodes nécessaires.

4°) Commentez le code suivant (complétez les "..." suivant votre implémentation) :

```
Batiment a = new Commerce(...);
Batiment b = new Industrie(...);
Batiment c = new Habitation(...);

System.out.println(a.toString() + "montant : " + a.calcImpot());
System.out.println(b + "montant : " + b.calcImpot());
System.out.println(c + "montant : " + c.calcImpot());
```

5°) D'après vous, est-il cohérent de pouvoir créer des instances de **Batiment** ?

6°) Aurait-on pu utiliser une énumération en lieu et place des classes **Habitation**, **Commerce** et **Industrie** ?