Programmation Objet Avancée - TD2

—o000o——o000o—

TP1 : Exercices de base sur les classes

1 Classe Polyligne

Une ligne polygonale, ou ligne brisée est une figure géométrique formée d'une suite de segments, la seconde extrémité de chacun d'entre eux étant la première du suivant. On appelle alors ligne polygonale la figure notée $A_1A_2A_3 \dots A_n$ et constituée par la suite des n˘1 segments $[A_1A_2]$, $[A_2A_3]$, …, $[A_{n˘1}A_n]$ comme montre la figure. Les points A_i sont appelés les sommets successifs de la ligne polygonale. De même, Les segments $[A_iA_{i+1}]sontlessegmentssuccessifsdelalignepolygonale.Lepoint A_i$ est nommé sommet commun des deux segments successifs $[A_{i-1}A_i]$ et $[A_iA_{i+1}]$.

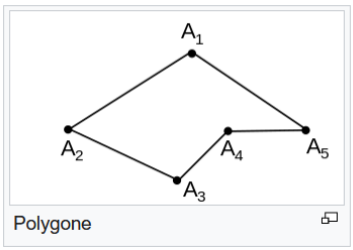

Figure 1: Polygone

La ligne polygonale est dite "fermée" si $A_1 = A_n$; on parle alors de polygone (voir la Figure 1). En se basant sur la classe Point qui est déjà définie :

1. Créer la classe Polyligne qui est définie par un ensemble de Point (les sommets de la polyligne) en implémentant ses attributs, constructeur et une méthode d'affichage.
2. Ajouter une méthode qui renvoie le nombre de sommets de la polyligne (i.e. le nombre de points).
3. Ajouter une méthode qui translate la polyligne selon un vecteur passé en paramètre.
4. Ajouter une méthode qui teste si la polyligne délimite un polygone.
5. Ajouter une méthode qui renvoie la longueur de la polyligne.
6. Ajouter une méthode qui insère des points au milieu de chaque couple de points. Cela réalise un sur-échantillonage de la polyligne.

Solution :

```
1
2
3  public class Polyligne {
4
5    private Point[] tab;
6
7    public Polyligne (Point[] points)
8    {
9      tab = points;
```

```
10    }
11
12    public void affiche()
13    {
14      for(int i=0; i<tab.length; i++)
15        tab[i].affiche();
16    }
17
18    public int nbPoints()
19    {
20      return tab.length;
21    }
22
23    public Polyligne translater(Vecteur v)
24    {
25      Point[] newtab = new Point[tab.length] ;
26      for(int i=0; i<tab.length; i++)
27        newtab[i] = tab[i].translater(v);
28      Polyligne resu = new Polyligne(newtab);
29      return resu;
30    }
31
32    public boolean polygone()
33    {
34      return  tab[0].egale(tab[tab.length-1]);
35    }
36
37    public double longueur()
38    {
39      double resu=0;
40      for(int i=0; i<tab.length-1; i++)
41        resu += tab[i].distance(tab[i+1]);
42      return resu;
43    }
44
45    public void surechantillonage()
46    {
47      Point[] newtab = new Point [(tab.length*2)-1];
48      int posi = 0;
49      for(int i=0; i<tab.length-1; i++) // pour tout les couples i i+1
50      {
51        newtab[posi]=tab[i]; // ajout du point
52        posi++;
53        newtab[posi]=tab[i].moyen1(tab[i+1]);
54        posi++;
55      }
56      newtab[posi]=tab[tab.length-1]; //ajout du dernier
57      tab = newtab; // changement du tableau
58    }
59
60    public Rectangle rectangleEnglobant()
61    {
62      double minx, miny,maxx, maxy;
```

```
63      // initialisation     l'aide des X et Y du premier point de tab
64      minx = tab[0].getAbscisse();
65      maxx = tab[0].getAbscisse();
66      miny = tab[0].getOrdonnee();
67      maxy = tab[0].getOrdonnee();
68      // recherche des max et des min     l'aide des autres points
69      for(int i=1; i<tab.length; i++)
70      {
71        if (tab[i].getAbscisse()<minx)
72          minx = tab[i].getAbscisse();
73        if (tab[i].getAbscisse()>maxx)
74          maxx = tab[i].getAbscisse();
75        if (tab[i].getOrdonnee()<miny)
76          miny = tab[i].getOrdonnee();
77        if (tab[i].getOrdonnee()>maxy)
78          maxy = tab[i].getOrdonnee();
79      }
80      //  c r ation  des deux points
81      Point bg = new Point(minx,miny);
82      Point hd = new Point(maxx,maxy);
83      //  c r ation  du rectangle
84      Rectangle resu = new Rectangle(bg,hd);
85      return resu;
86    }
87
88
89  }
```

TP2/java/polyligne.java

2 Compte bancaire

On va gérer les comptes bancaires du cette manière :

- Un compte bancaire possède à tout moment une donnée : son solde.
- Ce solde peut être positif (compte créditeur) ou négatif (compte débiteur).
- Chaque compte est caractérisé par un code incrémenté automatiquement.
- A sa création, un compte bancaire a un solde nul et un nouveau code lui est affecté (qui est le dernier code affecté à un compte incrémenté de 1).
- Il est aussi possible de créer un compte en précisant son solde initial.
- Utiliser son compte consiste à pouvoir y faire des dépôts et des retraits. Pour ces deux opérations, il faut connaître le montant de l'opération et voir si elle est réalisable avant de la faire.
- L'utilisateur peut aussi consulter le solde de son compte par la méthode `toString()`.
- Un compte Epargne est un compte bancaire qui possède en plus un champ "TauxInterêt=6" et une méthode `calculIntéret()` qui permet de mettre à jour le solde en tenant compte des interêts. Notant que les intérêts sont calculés selon l'équation suivante:

$$solde = solde * (1 + taux/100)$$

- Un ComptePayant est un compte bancaire pour lequel chaque opération de retrait et de versement est payante et vaut 5 euros.

Questions:

1. Définir les classes `CompteBancaire`, `CompteEpargne` et `ComptePayant`.
2. Définir une classe contenant la fonction `main()` permettant de tester les classes `CompteBancaire` et `CompteEpargne` avec les actions suivantes:
   - Créer une instance de la classe `CompteBancaire`, une autre de la classe `CompteEpargne` et une instance de la classe `ComptePayant`.
   - Faire appel à la méthode `deposer()` de chaque instance pour déposer une somme quelconque dans ses comptes.
   - Faire appel à la méthode `retirer()` de chaque instance pour retirer une somme quelconque de ses comptes.
   - Faire appel à la méthode `calculInteret()` du compte `Epargne`.
   - Afficher le solde des 3 comptes.

Solution :

```
public class CompteB {
int code;
float solde;
static int nbComptes=0;

public CompteB(float s){solde=s;++nbComptes;code=nbComptes;}
public CompteB(){this(0);}
public void verser(float mt){solde=solde+mt;}
public void retirer(float mt){solde-=mt;}
public String toString(){return ("code = "+code+ "  Solde= "+solde);}
}


public class CompteE extendsCompteB {
float taux=6;
public CompteE(float s){super(s);}
public CompteE(){super(0);}
public void calculInterets(){solde=solde*(1+taux/100);}
public String toString(){return "Compte Epargne "+super.toString()+ "
    taux= "+taux;}
}


public class CompteP extends CompteB {
public CompteP(float s){super(s);}
public CompteP(){super(0);}
public void verser(float mt){super.verser(mt);super.retirer(5);}
public void retirer(float mt){super.retirer(mt);super.retirer(5);}
public String toString(){return "Compte Payant "+super.toString();}
}


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

```
CompteB c1=new CompteB(5000);
CompteE c2=new CompteE(55000);
CompteP c3=new CompteP();
c1.verser(6000);
c1.retirer(4000);
System.out.println(c1.toString());
c2.verser(3300);
c2.retirer(2000);
System.out.println(c2);
c2.calculInterets();
System.out.println(c2);
c3.verser(2000);
System.out.println(c3.toString());}
}
```

TP2/java/compte_bancaire.java

3 Classe Animal

On va créer un zoo avec différents animaux. Chaque animal a une race, un nom, un âge et fait quelque chose comme bruit (un chien aboie par exemple, en français en faisant "oua", en anglais "bark", etc.)

1. Créez la classe Animal suivant les indications précédentes. Attention, cet animal doit pouvoir représenter tout type d'animal. Il n'est donc pas possible de savoir en l'état quel bruit il peut produire. Autrement formulé, quel type de classe devez-vous utiliser ?
2. Créez une méthode pour la classe Animal qui écrit sur l'écran la présentation d'un animal : son nom, sa race ...
3. Créez les classes dérivées pour deux races : chien et chat dont les bruits produits sont respectivement un aboiement ou un miaulement.
4. Créez une classe ZooMain qui contient la méthode main pour créer les animaux du zoo dans une variable static de la classe ZooMain en forme d'un tableau. Les cases du tableau correspondent aux cages du zoo.
5. Créez un chien et deux chats dans votre zoo et faites les présenter.

Solution :

```
public abstract class Animal
{
    // variables
    public String nom;
    public String race;
    public int age;


    // m thode constructrice
    public Animal(String a, String b, int c) {
        nom = a;
        race = b;
        age=c;


    }
```

```java
    // m thodes
    public abstract void jedis();
    public void sePresenter() {
        System.out.println("Bonjour! Je suis un(e) "+race+".");
        System.out.println("Je m'appelle "+nom+".");
        System.out.println("J'ai  "+age+" ans.");

    }
}
public class ZooApp
{
    // les cages dans le zoo
    public static Animal[] Zoo = new Animal[3];

    public static void main(String[] args) {
        Zoo[0] = new Chien("Lia");
        Zoo[1] = new Chien("Toto");
        Zoo[2] = new Chien("Sisi");

        for (int j=0; j<3; j++) Zoo[j].sePresenter();


    }
}
public class Chien extends Animal
{
    // m thode constructrice
    public Chien(String n) {
        super(n, "chien",3);
    }

  @Override
  public void jedis() {
    // TODO Auto-generated method stub
    System.out.println("vau, vau");
  }
}
```

TP2/java/animal.java

4 Metiers

Écrire les classes nécessaires au fonctionnement du programme suivant (en ne fournissant que les méthodes nécessaires à ce fonctionnement).

```java
    public class TestMetiers {
      public static void main(String[] argv) {
        Personne[] personnes = new Personne[3];
        personnes[0] = new Menuisier("Paul");
        personnes[1] = new Plombier("Jean");
        personnes[2] = new Menuisier("Adrien");
        for (int i = 0; i < personnes.length; i++)
          personnes[i].affiche();
```

```
62        }
63      }
```

Solution :

```
1  public abstract class Personne {
2
3    protected String nom;
4
5    public Personne(String nom) {
6      this.nom = nom;
7    }
8
9    public abstract void affiche();
10
11 }
12 public class Menuisier extends Personne {
13
14   public Menuisier(String nom) {
15     super(nom);
16   }
17
18   @Override
19   public void affiche() {
20     System.out.println("Je suis " + this.nom + " Le "
21         + this.getClass().getSimpleName());
22   }
23
24 }
25 public class Plombier extends Personne {
26
27   public Plombier(String nom) {
28     super(nom);
29   }
30
31   @Override
32   public void affiche() {
33     System.out.println("Je suis " + this.nom + " Le "
34         + this.getClass().getSimpleName());
35   }
36
37 }
```

TP2/java/metier.java

5 Polymorphisme

Qu'affiche le programme suivant ?

```
38 class A{
```

```
39 public String f (B obj){ return "A et B" ;}
40 public String f (A obj){ return "A et A" ;}
41 }
42 class B extends A{
43  public String f (B obj){ return "B et B" ;}
44  public String f (A obj){ return "B et A" ;}
45 }
46 public class Test{
47 public static void main(String[] argv){
48  A    a1  =new A ( ) ;
49  A    a2  =new B ( ) ;
50  B    b   =new B( );
51
52 System.out.println (a1.f(a1) ) ;
53 System.out.println (a1.f(a2) ) ;
54 System.out.println (a2.f(a1) ) ;
55 System.out.println (a2.f(a2) ) ;
56 System.out.println (a2.f(b) ) ;
57 System.out.println (b.f(a2) ) ;
58 }
59
60 }
```