ING 1

TD sur le polymorphisme

Écriture de classe – Héritage – Compréhension

Soit le programme suivant :

```
package heritage;


public class Film {
    // attributs

    public static  int RECENT = 2019;

    private String titre, realisateur;
    private int annee;
    private boolean estRecent = false;
    private int duree;


    // constructeur
    public Film(String t, String r, int d, int a){
        titre = t;
        realisateur = r;
        duree = d;
        annee = a;
    }


    // constructeur
    public Film(String t, String r, int d){
        titre = t;
        realisateur = r;
        duree = d;
        annee = 1900;
```

```
    }

        // constructeur
    public Film(String t, String r){
        titre = t;
        realisateur = r;
        duree = 0;
    }



    public boolean estRecent(){
        if ( annee >= RECENT)
            return true;
        else
            return false;
    }

    public void setAnnee(int a){
        annee = a;
    }


    @Override
    public String toString(){
        String res = "Titre : " + titre + "\n";

        res += "Réalisateur : " + realisateur + "\n";
        res += "Durée : " + duree + "\n";
        if (this.estRecent())
            res += "film récent ";
        else
            res += "film non récent ou date inconnue";

        return res;
    }
}
```

```
package heritage;

public class Telefilm extends Film {

    private String chaine;

    public Telefilm(String t, String r, int d, int a, String c){
        super(t,r,d,a);
        chaine = c;
    }

}
```

```java
package heritage;


public class Serie extends Film {

    private int [] dureeEpisode;
    private int nbEpisode;

    public Serie(String t, String r, int nb){
        super(t,r);
        nbEpisode = nb;

        dureeEpisode = new int[nbEpisode];
        for (int i = 0; i < nbEpisode; i++)
            dureeEpisode[i] = 0;
    }

    public void setDuree(int i, int d){
        dureeEpisode[i - 1 ] = 0;
    }

}
```

```java
package heritage;


public class Test {

    public static void main(String[] args){

        Film f1 = new Film("Le cas Richard Jewell", "Clint Eastwood",
                           209,2020);

        Film f2 = new Film("Antoinette dans les Cévennes","Carole Vignal",
                           127,2020);

        System.out.println(f1);
        System.out.println(f2);


        Serie s1 = new Serie("Le jeu de la Dame","Scott Frank",7);
        System.out.println(s1);


    }

}
```

- Expliquer les informations que le programme va afficher lors de son exécution

- Expliquer comment le programme décide si un film est récent ou non
- Décrivez la hiérarchie des classes dans un diagramme UML
- Comment est géré la durée d'un épisode ? Rajouter une méthode pour que les épisodes aient tous la même durée
- Vous pouvez voir que le programme n'est pas satisfaisant au niveau de l'affichage. Pour quelle(s) raison(s) ? Modifier les routines d'affichage

Écriture de classe – Héritage et Polymorphisme

On veut construire un programme permettant de gérer des figures géométriques colorées. Nous pourrons également faire des déplacements à ces différentes figures.

Un graphique est un ensemble de figures **affichables** et **déplaçables** en réalisant une translation.

On veut pouvoir afficher des segments, des triangles, des rectangles, et des carrées. Ces éléments graphiques sont définis à l'aide de points cartésiens (*cf* TD n°1). On est dans le cadre de la délégation où ces points sont utilisés dans nos figures.

L'application sera modélisée de la manière suivante :

- la classe `PointCartesien` représentera les coordonnées qui serviront à créer nos figures
- la classe `Segment` sera définie par deux points et une couleur, codée par trois entiers positifs (une composante rouge, une composante verte et une composante bleue https://fr.wikipedia.org/wiki/Rouge_vert_bleu)
- la classe `Forme` qui sera définie par un ou plusieurs segments formant une figure fermée de couleur identique
- la classe `Triangle` sera définie par trois points et une couleur (cf ci-dessus).
- la classe `Quadrilatère` sera définie par quatre points et une couleur
- la classe `Rectangle` sera définie par quatre points et une couleur
- –la classe `Graphique` est la classe principale et permettra d'afficher un ensemble d'objets graphiques.
- On sera amené à calculer le périmètre, la surface et obtenir la couleur de toutes les figures que l'on manipulera
- 1. Après avoir étudie les différences et les points communs entre les cinq premières classes que nous avons distinguées, décrivez le diagramme UML et précisez la hiérarchie des classes. Notez bien que nous

souhaitons réutiliser le maximum de code. Seule la classe `Forme` sera amenée à stocker des segments ;
Voici un exemple de fonctionnement qui peut vous inspirer une modélisation :
ackage couleurs;

```
import java.util.List;
import java.util.ArrayList;


public class Test {

    public static void main(String [] args){

        List<Forme> formes = new ArrayList<Forme>();

        PointCartesien p1 = new PointCartesien(1,1);
        PointCartesien p2 = new PointCartesien(1,-1);
        PointCartesien p3 = new PointCartesien(-1,1);
        PointCartesien p4 = new PointCartesien(-1,-1);
        PointCartesien p5 = new PointCartesien(-2,-2);


        Triangle t1 = new Triangle(p1, p2, p3, 100, 150, 200);
        Quadrilatere q1 = new Quadrilatere(p1, p2, p3, p5, 0, 100, 150);
        Rectangle r1 =  new Rectangle(p1, p2, p3, p4, 50, 50, 125);

        formes.add(t1);
        formes.add(q1);
        formes.add(r1);

        for(int i = 0; i < formes.size(); i++){
            System.out.print(formes.get(i));
            System.out.println("périmètre  " + formes.get(i).getPerimetre());
        }

    }

}
```

java couleurs.Test

```
Liste des segments du triangle
segment[P1PointCartesien{1.0,1.0}, P2PointCartesien{1.0,-1.0}]
segment[P1PointCartesien{1.0,-1.0}, P2PointCartesien{-1.0,1.0}]
segment[P1PointCartesien{1.0,1.0}, P2PointCartesien{-1.0,1.0}]
```

```
périmètre  6.82842712474619
Liste des segments du Quadrilatere
 segment[P1PointCartesien{1.0,1.0}, P2PointCartesien{1.0,-1.0}]
segment[P1PointCartesien{1.0,-1.0}, P2PointCartesien{-1.0,1.0}]
segment[P1PointCartesien{-1.0,1.0}, P2PointCartesien{-2.0,-2.0}]
segment[P1PointCartesien{-2.0,-2.0}, P2PointCartesien{1.0,1.0}]
périmètre  12.233345472033854
Liste des segments du Quadrilatere
 segment[P1PointCartesien{1.0,1.0}, P2PointCartesien{1.0,-1.0}]
segment[P1PointCartesien{1.0,-1.0}, P2PointCartesien{-1.0,1.0}]
segment[P1PointCartesien{-1.0,1.0}, P2PointCartesien{-1.0,-1.0}]
segment[P1PointCartesien{-1.0,-1.0}, P2PointCartesien{1.0,1.0}]
périmètre  9.65685424949238
```

- 2. Donnez le code de la classe `Segment`, et la méthode qui effectuera une translation. Où la (les) placer ?
- 3. Décrire entièrement la classe `Triangle`.