M1 bio-info

Java **2011-2012**

TP 6

Classes abstraites et un peu de dessin

Téléchargez sur Didel ou sur `http://igm.univ-mlv.fr/~fagnot/` le fichier `exo2.zip`. Compilez la classe `TestDessin` et testez-la.

Pour la suite, vous aurez besoin de regarder l'API de la classe `Polygon` (page `http://download.oracle.com/javase/6/docs/api/java/awt/Polygon.html`)

Exercice 1 On va définir une classe `Figure` dont voici le début :

```
import java.awt.Polygon;

public abstract class Figure{
    // coordonnées du centre approximatif de la figure
    private int posX;
    private int posY;

    public Figure(int x, int y){
        posX = x;
        posY = y;
    }
    ............
}
```

On définira aussi dans `Figure` les méthodes concrètes :

- `public int getPosX()` qui donnera la position horizontale du centre de la figure (abscisse) ;
- `public int getPosY()` qui donnera la position verticale du centre de la figure (ordonnée) ;

Ainsi que la méthode abstraite

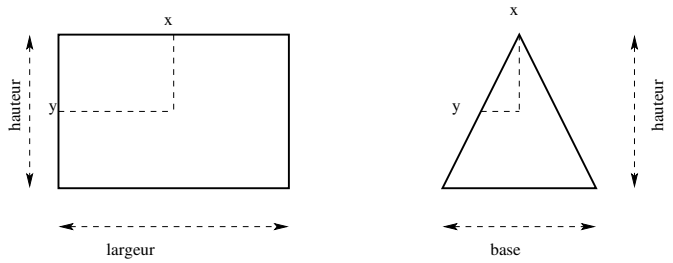
- `public abstract Polygon creePolygon();` qui créera un polygone correspondant à la figure.

Écrivez le code de `Figure`.

On définira aussi les classes concrètes suivantes : `Rectangle`, `Carre` et `Triangle`.

Donnez la hiérarchie de classes que vous choisiriez. Pour l'instant, n'écrivez pas de code pour ces classes.

Exercice 2 Écrivez le code de la classe `Rectangle`. On doit passer en paramètres du constructeur la position du centre, la largeur et la hauteur. Dans la figure ci-dessous, `x` et `y` représente la position du centre.



Exercice 3 Écrivez le code de la classe `Carre`. On doit passer en paramètres du constructeur la position du centre et la longueur d'un côté.

Exercice 4 Écrivez le code de la classe `Triangle`. On considère qu'un triangle est toujours isocèle et positionné comme sur le dessin ci-dessus. On doit passer en paramètres du constructeur la position du centre, la base et la hauteur (cf. dessin). Dans la figure ci-dessus, *x* et *y* représente la position du centre, *y* est à la moitié de la hauteur.

Si vous avez fini

Exercice 5 Les méthodes des questions suivantes sont à ajouter (sauf indication contraire) dans `Figure`. Pour chacune d'entre elles, demandez-vous si elle être abstraite ou non, et dans quelles classes il convient de la définir ou la redéfinir. Vous pouvez faire ces questions dans l'ordre qui vous convient.

- Écrivez la méthode `double estDistantDe(Figure fig)` qui calculera la distance entre le centre de la figure sur laquelle est appelée la méthode et le centre de `fig` ;
- Écrivez une méthode `déplacement(int x, int y)` qui déplace une figure (et la modifie donc) dans la direction indiquez par `x` et `y`.
- Écrivez une méthode `changeTaille(double coeff)` qui réduit ou agrandit la figure selon la valeur de `coeff`, par exemple si `coeff` vaut deux la figure sera deux fois plus grande, s'il vaut 0.5 elle sera deux fois plus petite.
- En regardant la classe `Dessin`, trouvez comment les couleurs sont définies. changez vos classes de telle manière qu'une figure ait une couleur, puis surchargez la méthode `dessine()` de `Dessin` de façon à ce qu'elle prenne une figure en paramètre et la dessine dans la couleur voulue.

La galaxie Blonk

Dans la galaxie Blonk, vivent les Schtarks. Les Schtarks sont de notre point de vue humain difformes et ceci de manière variée.

Les Schtarks sont constitués de trois grandes familles :

- les Zorglubs qui changent de forme comme ils veulent. Ils ont une couleur fixe (bleu, violet ou rose). Ils ont entre 4 et 11 yeux.
- Les Fgrongs qui ont tous deux yeux exactement et sont en forme de cube, de boule ou de pyramide. Leur couleur varient en fonction du temps qu'il fait.
- Les Slurps sont tous jaunes. Ils ont entre 5 et 8 mains. Ils se divisent en deux catégories, les Slurp rampants qui ont deux fois plus d'yeux que de mains et ont la forme d'un ver de terre et les Slurps sauteurs qui ont trois fois plus d'yeux que de mains et ressemblent à un gros ressort.

On a défini l'interface Schtark comme ci-dessous. Elle n'est pas modifiable.

```
public interface  Schtark{

      public abstract int nombreYeux();

      public abstract String quelleForme();
            // retourne "indefini" si forme changeante

      public abstract String quelleCouleur();
            // retourne "indefini" si couleur changeante
}
```

Avant de programmer les 3 premiers exercices, vous devez dessiner la hiérarchie en indiquant pour chaque classe ou interface, ce qui y est déclaré ou implémenté.

Exercice 1

Les classes `Zorglub` et `Fgrong` sont des classes concrètes implémentant `Schtark`. Si un constructeur est appelé avec une valeur non conforme, une valeur par défaut sera attribuée. Écrivez les classes `Zorglub` et `Fgrong`.

Exercice 2

La classe Slurp est une classe abstraite implémentant `Schtark`. Quelle(s) méthode(s) de `Schtark` peut-on concrétiser dans Slurp ? Écrivez la classe Slurp.

Écrivez également le code de `SlurpRampant` et de `SlurpSauteur`.

Exercice 3

Dans l'interface `Schtark`, on ajoute la méthode suivante

```
public abstract int vitesseDeDeplacement() throws NeSeDeplacePasException;
```

qui retourne en kilomètres par heure la vitesse de déplacement maximale du `Schtark` en question et où `NeSeDeplacePasException` est une exception qui n'a qu'un constructeur vide.

Sachant que

- Un Zorglub a une vitesse proportionnelle à son nombre d'yeux, soit 5 kilomètres/heure * nombre d'yeux.
- Les Fgrongs en forme de cube ou de pyramide ne se déplacent pas. Ceux en forme de boule se déplacent de 25 kilomètres par heure.
- Les Slurps se déplacent tous à 37 kilomètres/heure.

Écrivez le code de la classe `NeSeDeplacePasException`.

Écrivez pour chaque classe pour lesquelles cela s'avère nécessaire la méthode `vitesseDeDeplacement()`.

Exercice 5

Reprenez la définition du constructeur de `Zorglub` pour que si la couleur donnée en argument n'est pas correcte l'exception `IllegalArgumentException()` est lancée. L'exception `IllegalArgumentException` existe dans l'API est est une sous classe de `RuntimeException`.