IUT d'Orsay - BUT Informatique - Jean-Claude MARTIN – jean-claude.martin@universite-paris-saclay.fr

S2 R2.01 - Développement Orienté Objet

EX 5 : Classes Abstraites

Objectifs

- Comprendre l'utilité d'une classe abstraite
- Programmer des classes abstraites

---

1. Classe abstraite comme lien entre 2 programmeurs

ATTENTION : POUR CET EXERCICE, METTEZ VOUS EN BINÔME : ne faites pas cet exercice tout seul : cela n'aurait aucun intérêt !

(si vous êtes en avance, commencez par l'exercice suivant (banque et classes abstraites) et revenez à cet exercice 1 quand votre binôme est disponible)

un.e étudiant.e jouera le rôle de **Pasquier**, l'autre étudiant.e jouera le rôle de **Dupont**.

Pasquier est chef(fe) de projet et développeur.
Le but du projet est de développer un éditeur de dessin sur SmartPhone en Java.
Pasquier ne peut pas tout developper et doit collaborer avec Dupont.
Pasquier va prendre en charge la partie affichage graphique et stockage en mémoire de toutes les formes graphiques dans un tableau. Pasquier devra notament écrire un programme affichant des statistiques avec toutes les formes créées classées par surface decroissante.
Dupont va prendre en charge la partie implémentation de différentes formes graphiques : des cercles, des rectangles ... Cette liste pourra évoluer au fur et à mesure du projet.

A FAIRE PAR PASQUIER

Pasquier a écrit une classe générale **Editeur** qui inclut une méthode afficherStats. Cette méthode affiche une fenetre graphique avec les informations sur une forme graphique passée en parametre à cette méthode (notamment surface). Cette classe vous est fournie ci-dessous. Elle comporte des instructions liées aux interfaces graphiques que vous verrez lors d'un prochain cours et que vous n'avez pas besoin de comprendre dès maintenant.

```
import java.awt.* ;
import javax.swing.* ;

public class Editeur {

      public void afficherStats (FormeGeometrique f) {

                  // Declarer une reference vers une fenetre independante
                  Frame frame ;

                  // Creer un objet de type fenetre independante
                  frame = new Frame ("Statistiques sur une forme graphique");

                  // Declarer une reference vers une etiquette
                  Label labelNom ;

                  // Creer une etiquette avec toutes les informations sur la
forme
                  String classe = f.getClass().getName();

                  labelNom = new Label (classe + ", x = " + f.getX() + ", y = " +
f.getY() + ", Surface = " + f.surface () + ", Perimetre = " + f.perimetre ());

                  // Ajouter l'etiquette dans la frame
                  frame.add(labelNom);
                  // Fixer la taille de la fenetre
                  frame.setSize (300, 200);

                  // Afficher la fenetre
                  frame.setVisible (true);
        }
}
```

a) Créez un projet et copiez y la classe Editeur.

b) Vous devez programmer la classe abstraite **FormeGeometrique** qui comporte
- deux attributs x, y (la position de la forme géométrique)
- deux méthodes implémentées : void déplacer () et void afficher () (vous prévoierez des paramètres si besoin)
- deux méthodes abstraites : perimetre () et surface ()

c) Est-ce que votre programme peut s'exécuter avec uniquement les classes Editeur et la classe abstraite FormeGeometrique ? ……..
Que faut-il d'autre ? ……

Pasquier ne s'est pas préoccupé du type de FormeGeometrique *.*

Pasquier sait que pour utiliser ses classes, Dupont devra passer en paramètre à la méthode afficherStats() une instance d'une classe concrète qui implémente obligatoirement les méthodes spécifiées comme abstraites dans la classe FormeGeometrique .

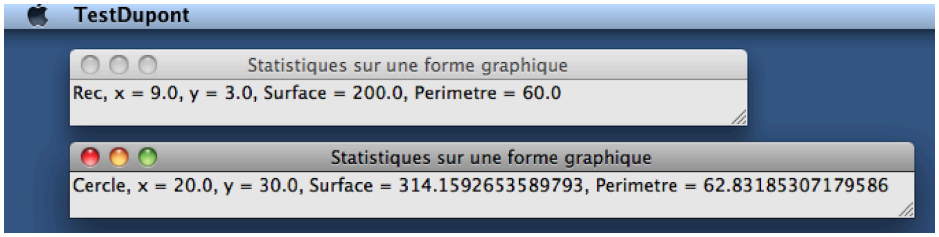
A FAIRE PAR DUPONT

a) Faites le travail de Dupont : programmez une classe **Rec** et une classe **Cercle** qui implémentent les méthodes abstraites de la classe abstraite FormeGeometrique.
Rappel : Le périmètre d'un cercle de rayon R est P = 2 π R. Sa surface est S = π R^2.
La valeur de PI est fournie par la classe Math (cherchez dans la javadoc).
Programmez aussi d'autres classes, par exemple TriangleRectangle.

b) Complétez ensuite la classe suivante pour tester le programme

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

    // Dupont cree un rectangle
      …
     // Dupont appelle les methodes developpees par Pasquier
     // Il est *oblige* de passer a la methode afficherStats un objet instance
d'une classe implementant la classe abstraite FormeGeometrique
     Editeur ed = new Editeur ();
      …
      // Afficher dans la console en appelant la méthode afficher sur cette
forme

     // Changer les coordonnées de la forme en appelant la méthode déplacer

      // Afficher dans la console en appelant la méthode afficher sur cette
forme

_     // Demander de nouveau d'afficher des statistiques sur cette forme

     // Idem avec un cercle
  }
}
```

Exemple d'affichage :



c)
Que se passe-t-il si Dupont n'implémente pas les méthodes abstraites de la classe abstraite FormeGeometrique ?
……

Enlevez le « extends FormeGeometrique “ dans une des classes.
Est-ce qu'il y a une erreur à la compilation ou à l’exécution ?
Quelle est l’instruction provoquant une erreur ?
……

---

2. Banque et classes abstraites

Dans votre projet banqueHeritee est-ce logique d’avoir une classe concrète Compte ? …....

Est-ce que vous aurez à un moment donné besoin de créer un objet de type Compte qui ne soit ni un compte courant ni un compte d’épargne ?……
……

Dupliquer votre projet banqueHeritee en un projet banqueAbstraite.
Modifier les différentes classes de manière à gérer de manière plus générale une classe abstraite Compte et ses sous-classes. On supposera maintenant que les manières de créditer et débiter un compte ne doivent pas être détaillées dès la classe Compte.

---

3. Salariés de la banque

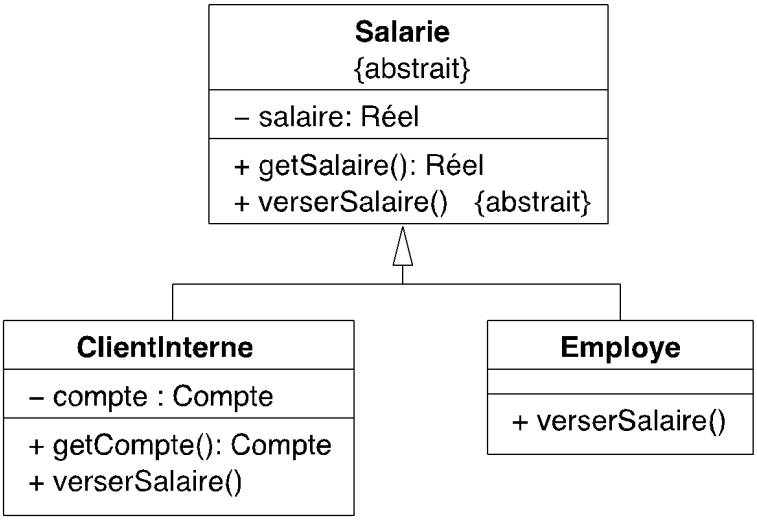
On distingue deux catégories de salariés dans une banque :

1) ceux qui ont́ un compte dans la banque (les « clients internes »)
2) ceux qui sont clients d’une autre banque (les « employés »)

Suivant le cas, la méthode verserSalaire doit effectuer un traitement différent :

1) pour les clients internes : un virement sur le compte interne
2) pour les employés : l’envoi d’un chèque que l’on simulera par un affichage à l’écran de la phrase « Versement de salaire sur compte externe » accompagné du montant de son salaire.

Pour représenter les attributs et méthodes communs à ces deux classes, on les fait hériter d’une classe abstraite Salarie. Cette classe est abstraite car elle n’implémente pas la méthode verserSalaire () mais la déclare comme méthode abstraite pour obliger toute classe dérivée à la redéfinir.



Définissez ces 3 classes.

Dans le main d'une classe TestSalarie, ajouter des instructions créant plusieurs salariés et appelant leur méthodes verserSalaire.

---

4. OPTIONNEL : Exercice sur l'héritage : « Les limites de A ''est un'' B... »

(Martin Morterol)

Instructions : lire l'énoncé en entier.

1) définir une classe Rectangle ayant une longueur, une largeur ainsi que des getteurs / setteurs et une méthode de calcul de l'aire du rectangle.

2) Définir une classe **TestRectangle** qui vérifie que les rectangles ont bien un comportement cohérent. Elle aura en attribut un rectangle. La classe aura autant de méthodes que vous jugerez nécessaire. Ces tests affecteront différentes valeurs aux attributs du rectangle et vérifieront que les méthodes retournent une valeur correcte ou modifie comme attendu les attributs du rectangle.

Un exemple (incomplet) de test peut être le suivant :

```
private boolean testAire()
{
        rect.setLongueur (5);
        rect.setLargeur (10);
        return (rect.calculAire()==50) ;
}
public boolean isRectangleValide()
{
        return testAire() && testQch() && …. && … ;
}
```

3) Implémenter une classe Carré qui hérite de Rectangle. Redéfinissez ses setters afin d'assurer un comportement correct d'un carré.

4) Puisqu'un Carré est-un rectangle, vous devez pouvoir tester votre carré grâce à la classe « TestRectangle ». Que constater vous ?……

5) Proposer une architecture permettant de ne plus avoir de comportement incohérent. N'hésitez pas à définir une classe abstraite.

Commentaire : **Un objet B peut/doit hériter de A si « B se comporte comme un A »** ou encore si « toutes méthodes qui utilisent un objet d'une classe mère doivent pouvoir utiliser toute instance d'une classe dérivée sans avoir à le savoir ». Cela s'appelle le « principe de substitution de liskov ». C'est un principe fondamental de la programmation orientée objet. Pour plus d'information, rechercher sur Google « S.O.L.I.D »

Section Exemple de violation du LSP dans
https://fr.wikipedia.org/wiki/Principe_de_substitution_de_Liskov

SOLID https://fr.wikipedia.org/wiki/SOLID_(informatique)