TD1 : La programmation orientée objet et Java
V1.1.1

Document en ligne : www.mickael-martin-nevot.com

1 Généralités

Écrivez les applications ci-dessous en Java et en respectant la norme de programmation donnée en cours ; puis testez-les.

2 Géométrie

2.1 Point

2.1.1 Classe Point

La classe `Point` permet de manipuler un point. Cette classe contient :
- les constantes de classe correspondant aux coordonnées par défaut : `DFL_X` et `DFL_Y` ;
- les variables d'instance correspondant aux coordonnées : `x` et `y` ;
- les accesseurs et mutateurs associés à ces variables d'instance ;
- plusieurs constructeurs (avec ou sans coordonnées par défaut) ;
- la méthode `move(double x, double y)` qui effectue une translation ;
- la méthode `toString()` qui retourne une chaîne de caractères au format : `"(x,y)"`.

2.1.2 Classe TestPoint

La classe `TestPoint` contient une méthode principale qui crée plusieurs objets de type `Point` (avec ou sans coordonnées par défaut), les déplace et les affiche (avant et après les déplacements).

2.2 Cercle

2.2.1 Classe Circle

La classe `Circle` permet de représenter un cercle. Un `Circle` est caractérisé par son centre (un `Point`) et son rayon. Outre les accesseurs et mutateurs associés à ses variables d'instance, un `Circle` doit avoir des méthodes permettant de :
- calculer sa surface ;
- calculer son périmètre ;
- tester sa superposition avec un autre `Circle` ;

- tester l'appartenance d'un `Point` à l'objet ;
- renvoyer une chaîne de caractère représentant le `Circle` de manière textuelle.

2.2.2 Classe TestCircle

La classe `TestCircle` doit réaliser les instructions suivantes (dans l'ordre) :

- création d'un `Circle` de centre (5,5) et de rayon 2 ;
- affichage de son périmètre ;
- affichage de sa surface ;
- multiplication par trois de son rayon ;
- nouvel affichage de son périmètre
- nouvel affichage de sa surface ;
- translation du centre du `Circle` à l'origine ;
- vérification que le `Point` (1,1) est bien à l'intérieur du `Circle` ;
- vérification que le `Point` (9,9) est bien à l'extérieur du `Circle` ;
- test de superposition du `Circle` avec un autre `Circle` de centre (0,0) et de rayon 2 ;
- test de superposition du `Circle` avec un autre `Circle` de centre (0,0) et de rayon 6.

3 Gestion d'une bibliothèque

Vous devez créer une application de gestion d'une bibliothèque.

3.1 Classe Book

Cette classe contient :

- les variables d'instance : `title`, `author`, `editor` et `pageNb` ;
- un constructeur ;
- les accesseurs et mutateurs associés à ces variables d'instance ;
- une méthode permettant d'afficher le livre ;
- une méthode permettant de vérifier si deux instances de `Book` sont égales.

3.2 Classe Library

Cette classe contient :

- la constante de classe : `MAX_BOOKS`
- les variables d'instance : `name`, `address`, `max` et un tableau de `Book` d'au plus `MAX_BOOKS` ;
- un constructeur ;
- les accesseurs et mutateurs associés à ces variables d'instance ;
- une méthode permettant d'afficher les livres de la bibliothèque ;
- une méthode permettant d'ajouter un livre à la bibliothèque ;
- une méthode permettant de retirer un livre de la bibliothèque ;
- une méthode permettant d'éliminer les doublons de la bibliothèque ;
- une méthode qui prend comme argument une bibliothèque et affiche les livres présents dans les deux bibliothèques ;
- une méthode permettant de trier les livres de la bibliothèque par auteur.

3.3 Classe LibraryUI

Cette classe propose à l'utilisateur un menu complet (pas nécessairement graphique) pour interagir

avec la bibliothèque.

4 Gestion d'une mairie

Vous devez créer une application pour l'automatisation du calcul des impôts locaux d'une mairie. Il y a deux types d'habitation (le calcul d'impôt n'étant pas le même dans les deux cas) :

- les maisons personnelles ;
- les habitations professionnelles.

Ces deux types d'habitation ont des caractéristiques communes.

4.1 Classe Home

La classe `Home` comprend :

- les attributs `owner` (propriétaire), `address`, `surface` ;
- un constructeur ;
- la méthode `tax()` qui calcule le montant de l'impôt général en fonction de la surface (**2 € /m²**) ;
- la méthode `toString()`.

4.2 Classes PersonalHome et ProHome

4.2.1 PersonalHome

La classe `PersonalHome` hérite de la classe `Home`.

Le calcul de l'impôt d'une maison personnelle se calcule en fonction de la surface habitable, du nombre de pièces (**15 € par pièce**) et de la présence ou non d'une piscine (**80 €**).

4.2.2 ProHome

La classe `ProHome` hérite de la classe `Home`.

Le calcul de l'impôt d'une habitation professionnelle se calcule en fonction de la surface et du nombre d'employés travaillant dans l'entreprise (**150 € par dizaine d'employés**).