

TD2 POO / Java : héritage

Objectif : Manipuler le concept d'héritage et savoir concevoir une classe dérivée.

Matériel / Logiciel : Environnement de développement sur PC (sous Linux) : JDK1.7 (ou supérieur) : compilateur java (javac) et JVM (java) IDE Geany / éditeur gedit et console / Eclipse.

Acquisition : l'héritage.

1 La classe PointCouleur

Reprendre l'exemple du CM et réaliser les classes « PointCouleur » et « TestPointCouleur »

Remarque : pour cette séance et pour la suite, on pourra utiliser l'IDE¹ Eclipse.

2 La classe Compteur

On souhaite concevoir puis implémenter une classe représentant un compteur entier. Un tel objet se caractérise par une valeur entière, positive ou nulle, en tenant compte du fait qu'elle ne peut varier que par pas de 1 (incréméntation ou décréméntation). On convient ici qu'une décréméntation d'un compteur nul est sans effet.

Définir le diagramme de classes UML de ce problème.

2.1 La classe Compteur

On veut créer un simple compteur ayant une valeur initiale nulle. Il s'agit donc de créer une classe Compteur pour rendre le service demandé :

rédiger en particulier les méthodes `incrémenter()` et `decrémenter()`. On écrira en outre une petite classe de test qui :

- a) créera un compteur et affichera sa valeur;
- b) l'incrémentera 10 fois, puis affichera à nouveau sa valeur;
- c) le décrémentera 20 fois, puis affichera une troisième fois sa valeur

On pourra si nécessaire, dans un premier temps ne pas se préoccuper des paquetages mais néanmoins écrire chaque classe (publique) dans des fichiers sources (.java) distincts : `Compteur.java` et `TestCompteur.java`. Les classes créées seront alors dans le paquetage par défaut.

L'affichage de ce programme doit donner (quelque chose comme) "0 10 0".

2.2 finalisation de la classe Compteur

Modifier la classe Compteur en ajoutant un deuxième constructeur de Compteur initialisé à une valeur positive de départ. Ajouter si ce n'est déjà fait une méthode de réinitialisation.

Organiser les classes créées dans des paquetages : la classe Compteur appartient au paquetage "comptage", la classe TestCompteur appartient au paquetage "test".

Apporter les améliorations utiles à vos codes : méthodes `toString()`, accesseurs et mutateurs, usage de « `this.` » et « `this()` » et éventuellement « `this` ».

¹ IDE : Integrated Development Environment.

3 classes dérivées de la classe Compteur

3.1 La classe *CompteurBorne*

On souhaite à présent définir une classe *CompteurBorne*, liée à la classe *Compteur* précédente par une relation d'héritage.

Un compteur borné est limité en valeur haute (il ne peut excéder un maximum défini lors de sa création).

Définir le modèle UML d'une telle classe (en complétant le diagramme de l'exercice du compteur simple).

Écrire les classes *CompteurBorne* et *TestCompteurBorne*

3.2 La classe *CompteurCyclique*

On souhaite maintenant concevoir une classe *CompteurCyclique* qui retournera à zéro dans le cas d'une incrémentation lorsque la valeur du compteur cyclique a atteint son maximum, et réciproquement retournera à son maximum lors d'une décrémentation d'un compteur cyclique dont la valeur est zéro.

Définir le modèle UML de cette nouvelle classe (en complétant le diagramme de classes précédant).

Écrire les classes *CompteurCyclique* et *TestCompteurCyclique*.