Programmation orientée objet (C++) – ING2-GSI

TP3 : Héritage

Objectifs

- Appréhender la notion d'héritage par des exemples typiques d'utilisation.

1 - Compteur

Un compteur est défini par les propriétés suivantes :

- Il possède une valeur entière, positive ou nulle, à sa création.
- Il ne peut varier que par pas de 1 (incrémentation ou décrémentation). On admettra que décrémenter un compteur nul laisse ce dernier à 0.

Ecrire une classe Compteur sur ce cahier des charges simple.

Ensuite, écrire un programme permettant de tester ce compteur comme suit :

- D'abord, le programme crée un compteur et affiche sa valeur.
- Ensuite, il incrémente 10 fois ce compteur avant d'afficher à nouveau sa valeur.
- Enfin, il décrémente 20 fois ce compteur, et il affiche une dernière fois sa valeur.

Que doit rendre le programme en sortie ?

2 - Compteur borné

On va opérer une modification du cahier des charges de la classe Compteur : notre client désire maintenant pouvoir spécifier, lors de la création d'un compteur, une borne maximale que le compteur ne pourra pas dépasser. On va pour cela créer une classe dérivée de la classe précédente qu'on appellera `CompteurBorne`.
Définir un constructeur permettant de fournir une borne maximale éventuelle à un compteur, et mettre à jour la structure et les méthodes nécessaires pour prendre en compte cette notion de borne.
Modifier le programme de test pour qu'il effectue la même série d'opérations sur un compteur normal et sur un compteur borné.

3 - Le coursier

Nous allons simuler le travail d'un coursier dans une grande entreprise. Ce coursier passe tous les matins dans les bureaux et récupère le courrier à affranchir. Ce courrier est de deux types, avec une différence au niveau du processus d'affranchissement :

- les lettres, qui peuvent être ordinaires (0,5 euro) ou urgentes (+0,3 euro). On convient que la lettre représente 1 unité de volume.
- les colis, dont l'affranchissement dépend du volume (0,5 euro / unité de volume).

Le coursier place les courriers dans un grand sac (de capacité fixée à sa création). De retour dans son bureau, il place le sac dans une machine a affranchir (qui affranchit automatiquement

les courriers contenus dans le sac en fonction de leur type). On demande de modéliser la tournée du coursier en

- créant un sac et quelques courriers,
- plaçant les courriers dans le sac,
- affranchissant le sac,
- affichant le tarif des différents courriers.

Dans un premier temps, nous implémentons le sac à l'aide d'un tableau. On supposera qu'on dispose de la fonction `aleatoire()` qui renvoie un nombre réel aléatoire entre 0 et 1.

4 - Villes et capitales

Ecrire une classe Ville et une classe Capitale qui en hérite avec les instructions suivantes :

- Une ville est décrite par son nom et son nombre d'habitants.
- Le nom d'une ville ne peut pas varier ; Il doit être connu dès l'instanciation de l'objet.

Par la suite, ce nom servira de clé de recherche d'informations.

- Les villes sont classées en 3 catégories : grande (plus de 500 000 habitants), moyenne (entre 100 000 et 500 000) ou petite (moins de 100 000).
- La classe Ville doit fournir une méthode "`obtenirInformations()`" qui retourne les informations sous la forme d'une chaîne de caractères.
- Une capitale est une ville et contient l'information du pays où elle est située. Il faut donc surcharger la méthode obtenirInformations() pour rajouter cette dernière information.

Exemple :

```
V : Ville
v ← nouveau Ville("Les Pennes-Mirabeau",19022)
Ecrire( v.obtenirInformations() )
```

a pour effet d'afficher :

```
les pennes-mirabeau : 19022 habitants, petite ville.
```

Autre exemple:

```
c : Capitale
c ← nouveau Capitale("Paris",2153600,"France")
Ecrire( c.obtenirInformations() )
```

affiche :

```
paris : 2153600 habitants, grande ville. Capitale de France.
```