TP C++/OOP

Des évènements et des écouteurs

TP évalué

Le but de ce TP est d'utiliser les notions de polymorphisme et de collection. Nous allons simuler un système d'alarme à base de détecteurs positionnés dans différentes zones d'un bâtiment en créant les objets nécessaires, puis créer un système de commande qui va permettre via la ligne de commande d'interagir avec le système d'alarme.

Evaluation du TP

Ce TP doit être rendu au plus tard le **08/12**. Les différentes classes seront créées dans des fichiers .h et .cpp séparés. Vous êtes libres d'utiliser l'arborescence des dossiers qui vous convient, la seule contrainte est que la compilation du programme fonctionne en incluant tous les fichiers .cpp présents dans l'arborescence.
Les travaux seront rendus par emails sous la forme d'un zip nommé précisement **PRENOM_NOM-detecteurs.zip**.

Partie 1 : zones et détecteurs

Les détecteurs

Un détecteur est un élément dont le rôle est très simple : il surveille une zone, et se déclenche dès qu'il détecte un mouvement. Le déclenchement peut provoquer l'émission d'un signal si le détecteur est activé.
Créer une classe **Detecteur** qui contient un attribut boolean qui permet de savoir s'il est activé ou non, et trois méthode :

- void activer() : active le capteur
- void desactiver() : désactive le capteur
- void detecter() : déclenche le signal si le capteur est activé. Pour l'instant, cette méthode ne fera qu'afficher le message "signal envoyé"

La gestion des zones

Les bâtiments à sécuriser sont découpés en zones. Chaque zone est équipée d'un détecteur. Quand on entre dans une zone, le détecteur associé à cette zone se déclenche.
Créer une classe **Batiment** qui va servir à gérer ces zones. Elle contient :

- Un attribut de type map qui stocke les détecteurs de chacunes des zones. La clef est le nom de la zone, la valeur associée une instance de *Detecteur*.
- Des méthodes :

- void creerZone(string nom) : crée un détecteur et l'associe à la zone indiqué
- void activerZone(string nom) : active le détecteur présent dans la zone indiqué
- void desactiverZone(string nom) : désactive le détecteur de la zone indiqué
- void entrerZone(string nom) : déclenche le détecteur de la zone indiqué

Test des classes

Créer une fonction testZonage(), appelée directement dans la fonction main, et qui réalise les opérations suivantes :

1. Création d'un Batiment
2. Création d'une zone "livraison"
3. Création d'une zone "accueil"
4. Création d'une zone "magasin"
5. Activation des zones livraison et magasin, et desactivation de la zone accueil
6. Pénétrer dans la zone accueil -> le détecteur ne doit pas réagir
7. Pénétrer dans la zone magasin -> le détecteur doit le signaler

Partie 2 : Les Actions et la télécommande

La télécommande va permettre de gérer le bâtiment à partir de la ligne de commande en spécifiant les actions que l'on souhaite faire.
Le déroulé du programme sera le suivant :

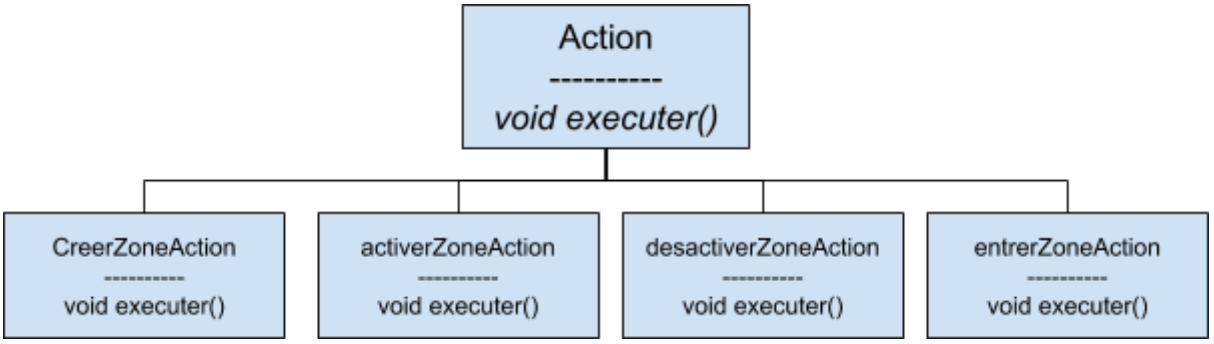
- Le programme attend un mot clef de l'utilisateur, qui indique la commande à effectuer, par exemple : *activer, creer, désactiver*
- En fonction de la commande entrée, le programme exécutera l'action qui correspond

Exemple de déroulé du programme :

```
> creer
Créer une zone. Entrez le nom de la zone: _(on tape magasin)
OK, zone magasin créée
> activer
Activer une zone. Entrez le nom de la zone : _(on tape magasin)
Zone magasin activée
> entrer
Dans quelle zone voulez vous entrer : _(on tape magasin)
Vous êtes maintenant dans la zone magasin
SIGNAL DU DETECTEUR
```

Cette télécommande pourrait être implémentée avec un *switch* et en mettant le code qui va bien dans chaque section, mais nous allons utiliser une approche plus modulaire à l'aide du polymorphisme. Une classe **Action** va représenter une action quelconque. Le code qui permet de gérer l'action est implémenté dans autant de classes filles que d'action possibles.

Les classes filles sont ensuite associée à leur mot clef pour être exécutée quand on les demande :



La classe Action

Créer une classe **Action** qui ne contient qu'une seule méthode : void executer(). Cette méthode ne sera pas implémentée dans la classe Action, elle le sera dans les classes filles.

Les classes Action

Créer les 4 classes d'action telles que sur le schéma ci dessus. Chacune de ces actions aura comme attribut un pointeur vers le Batiment à gérer, et implémenterons le code de la méthode *executer()* en fonction de l'action à exécuter :

1. Afficher sur la console l'action demandée
2. Demander le nom de la zone sur laquelle porte l'action
3. Déclencher la méthode qui va bien sur la classe Batiment sur la zone demandée

La classe télécommande

Il faut maintenant faire le lien entre la demande utilisateur et l'action à exécuter. Ce sera le rôle de la classe **Telecommande**.
Elle va contenir comme attribut :

- Une map<string, action> qui va faire le lien entre le code de l'action et la classe Action qui va avec

Et comme méthodes :

- void ajouterAction(string nom, Action action) qui ajoute l'action dans la map des actions
- void demarrer() qui démarre les interactions avec l'utilisateur. Elle boucle sur les opérations suivantes :
    a. Demander le code de l'action que l'utilisateur souhaite faire
    b. Aller chercher dans la map la classe Action associée
    c. Exécuter la méthode *execute()* de l'action correspondante

Passer dans le constructeur de la classe Telecommande un pointeur vers un objet Batiment, et créer dans ce constructeur l'ensemble des classes actions qui ont été implémentées précédement pour les codes *creer, activer, desactiver* et *entrer*.

La fonction main() définitive

Conserver dans la fonction main() l'appel à la fonction *testZonage()* pour l'évaluation du TP, mais ajouter à la suite :

- La création d'un nouveau **Batiment** vide
- La création d'une **Telecommande** pour gérer ce nouveau batiment
- L'exécution de la Telecommande