1 Simulation type SimCity

Nous nous intéressons à une problématique de simulation d'une ville (*jeu de type Sim City*). Il s'agit pour l'instant de proposer une modélisation de quelques comportements des habitants.

Pour se déplacer dans la ville, les habitants doivent pouvoir emprunter différents modes de locomotion : marche, vélo ou transports en commun.

D'autres part, des zones pour les habitations sont localisées dans la ville et plusieurs facteurs peuvent intervenir dans le choix d'un lieu d'habitation par un habitant : le temps de transport domicile-travail, la proximité de la nature, des commerces ou encore des écoles, etc. Trois combinaisons de ces critères, donnant plus ou moins d'importance à l'un ou à l'autre, doivent être observables chez les habitants :

- moins de transports possible ;
- le plus proche possible de la nature (parcs) ;
- importance égale accordée à tous les facteurs.

La simulation doit pouvoir faire intervenir différents types d'habitants aux comportements prédéfinis. Par exemple :

- l'amoureux de la nature se déplace en marchant et habite proche des parcs
- le sportif se déplace en vélo et habite proche des parcs (pour pouvoir courir)
- la famille nombreuse se déplace en transports en commun et tient compte de tout les facteurs pour le choix de son lieu d'habitation.

2 Montre digitale

Nous désirons modéliser le fonctionnement d'une montre digitale. La montre comporte un affichage et quatre boutons. Trois modes de fonctionnement sont possibles : affichage de l'heure, chronomètre et réglages.

Le bouton A est utilisé pour changer de mode, ce qui s'effectue de manière cyclique : heure => chronomètre => réglages => heure...

Le bouton B est utilisé pour éclairer la montre. Le fonctionnement de ce bouton est identique quel que soit le mode de fonctionnement.

En mode affichage de l'heure, le bouton C permet de passer d'un affichage de l'heure sur 12 ou 24h. Le bouton D quant à lui est inactif.

En mode chronomètre, le bouton C permet de lancer le chronomètre et de prendre des temps intermédiaires. Le bouton D est utilisé pour arrêter le chronomètre. Une pression longue sur le bouton D permet de remettre le chronomètre à zéro. Lorsque la montre est en mode chronomètre, l'affichage est divisé en deux. La partie supérieure montre le temps écoulé depuis le déclanchement tandis que la partie inférieure s'arrête sur le dernier temps intermédiaire. Le chronomètre doit pouvoir continuer de fonctionner lorsque l'on change de mode.

Enfin, en mode réglage, le bouton C permet d'avancer l'heure d'une heure, tandis que le bouton D l'avance d'une minute.

3 Télécommande programmable

On souhaite modéliser des télécommandes programmables destinées à commander les appareils électrique d'une maison.

Ces télécommandes disposent d'un certain nombre de boutons dont on peut choisir et modifier l'action. Il peut s'agir d'allumer ou d'éteindre un appareil électrique ou encore d'en monter le son ou bien de passer à la chaîne suivante d'un téléviseur, etc. On souhaite donc disposer d'une classe `Telecommande` dont le constructeur sera paramétré par le nombre de boutons de la télécommande. Les appareils électriques connectés possèdent tous à minima des méthodes `allumer()` et `eteindre()`. En fonction de l'appareil, d'autres actions peuvent être possibles (exemple : `augmenter_volume()` pour une chaîne hifi)

Chaque bouton d'une télécommande est numéroté et la classe `Telecommande` dispose d'une méthode publique `appuyer_bouton(int i)` dont l'invocation correspond à l'appui sur le bouton numéro i de la télécommande. Elle doit donc déclencher l'action associée.

Il doit donc être possible d'associer à chaque bouton "i" une action à déclencher.

Q1 : modéliser le problème sous la forme d'un diagramme UML

Q2 : . . .

4 GeoSport

L'objectif de se projet est de réaliser une application pour téléphone portable qui enregistre périodiquement (la période est un paramètre définit par l'utilisateur) les positions GPS d'un coureur. Elle affiche en temps réel la durée de l'effort et la vitesse instantanée. A la fin de l'exercice, elle doit pouvoir afficher la distance totale parcourue, les vitesses moyenne/minimale/maximale. Elle doit également pouvoir représenter le parcours sur un fond de carte Google Maps.

Enfin, l'application conserve l'historique des différents parcours, ce qui permet de calculer et afficher la progression du sportif.

5 Le logger

Nous souhaitons écrire une classe permettant à des utilisateurs de tracer l'exécution d'un programme en affichant des messages dans la console.

Un `Logger` est un objet possédant trois méthodes :

- `logInfo(String msg)`
- `logDebug(String msg)`
- `logError(String msg)`

L'état de l'objet `Logger`, stocké dans un attribut `logLevel`, nous indique quels messages doivent être affichés. `logLevel` peut ainsi prendre trois valeurs : `INFO`, `DEBUG` ou `ERROR`.

Par exemple, si niveau est à `INFO` les trois méthodes affichent leur messages dans la console. En revanche, si le niveau est à `ERROR` seule la méthode `logError()` affichera quelquechose, les autres méthodes n'affichant rien.

Q1 : proposez une modélisation de la classe `Logger`.

Q2 : quels sont les défauts de cette modélisation ?

Q3 : fournissez une nouvelle solution qui permette de s'affranchir des défauts constatés.

Nous désirons maintenant écrire des logs sur différents supports et plus uniquement dans la console (par exemple, dans des fichiers XML ou dans une base de données).

Q4 : faites évoluer votre modèle pour prendre en compte cette modification.

6 Application de filtres sur des images

Nous cherchons à réaliser un programme destiné à appliquer des filtres sur des images. Une image est caractérisée par son nom, son format, sa taille et le répertoire dans lequel elle se trouve. Le filtre doit permettre de modifier la couleur et/ou d'appliquer une rotation et/ou de flouter l'image.

L'application du filtre ne modifie pas l'image d'origine mais en crée une nouvelle dans le même répertoire.

Q1 : proposez une modélisation des classes impliquées.

Le filtre dépend en fait du format de l'image (jpg, png, tiff, etc.).

Q2 : modifier la structure du programme pour que le filtre s'adapte au type de l'image.

7 Fichiers et arborescences

Vous venez d'arriver dans une nouvelle entreprise et vous récupérez un programme, mis en place par votre prédécesseur, qui étant donné un chemin complet vers un fichier Windows, en affiche juste le nom.

Le code est composé d'une seule classe écrite en Java :

```
public class Main {
    public static void parse_filename(String path) {
        //index est l'endroit où se situe, dans la String path, la dernière
        //apparition du caractère \
        int index = path.lastIndexOf("\\");
        //On construit une String qui ne contient que la partie située à
            droite
        //du dernier caractère \
        String r = path.substring(index+1);
        System.out.println(r);
    }
}
```

Vous souhaitez faire évoluer le programme pour qu'il puisse prendre en compte les chemins Linux.

Q1 : proposez une modélisation de la nouvelle mouture du programme.

Le programme étant fonctionnel, nous allons tenter d'ajouter de nouvelles fonctionnalités. Nous voulons compter le nombre de dossiers qu'il faut parcourir depuis la racine pour trouver le fichier.

Q2 : ajoutez cette fonctionnalité à votre modèle en n'oubliant pas que là encore, la méthode sera différente pour les chemins Windows et Linux.

Q3 : adaptez votre code pour le rendre modulable à souhait (pouvoir ajouter autant de fonctionnalités que souhaité).