Design Pattern – TD n°1

Dans ce premier TD, nous allons voir comment un projet doit être attentivement réfléchi avant de passer à l'écriture du code. Le but ici sera de penser une architecture de programme sur papier, pour chacun des exercices. Il faudra s'interroger sur ce qui peut évoluer, ce qui restera figé en posant des questions au client (le professeur) et en anticipant ses changements d'avis.

VIDÉO CLUB 24/24

Vous êtes responsable du développement du logiciel d'un distributeur de DVD.

.Grâce à un écran vidéo et un clavier, un client pourra venir louer une cassette pour 24h pour 2 euros. Il s'identifiera grâce à son numéro de client (unique) et son mot de passe.

.Le client doit rendre sa cassette à temps sous peine d'avoir une pénalité de 0,5 euro par jour de retard.

.Un film peut être disponible en plusieurs exemplaires. Lorsque l'on affiche la liste des films disponibles, on doit montrer le nombre d'exemplaires disponibles à l'instant, et le nombre d'exemplaires en tout qu'il y avait à l'origine.

.Pour payer, le client devra entrer un numéro de carte de crédit (qui sera toujours bon !).

.Les films (caractérisés par un titre, une date de sortie en salle, et un type), doivent pouvoir être listés par titre (ordre alphabétique).

1. Par binômes, réfléchissez à l'architecture du programme à l'intérieur de la machine de location. Le programme doit comporter un menu général d'où le client peut voir la liste des films disponibles, demander à louer un film (dans ce cas, il devra entrer son numéro de carte de crédit), ou rendre un film (et dans ce cas, on lui affichera les pénalités de retard qui seront prélevées sur sa carte).

Utilisez les cartes données en classe afin d'élaborer votre structure de classe.

2. Par binômes, codez la solution d'architecture retenue et testez la (30 mn en tout disponibles, mettez cinq films en exemple dans votre base de films, et trois clients).

3. Échangez votre programme avec un autre binôme, et commentez leur architecture de classe. Attendez les instructions supplémentaires.

4. Récupérez votre programme. Attendez les instructions supplémentaires.

LES FOURMIS

Vous devez simuler le comportement d'une fourmilière dans un environnement représenté sous forme d'un tableau 2d, comportant une case fourmilière et certaines cases (choisies au hasard) nourriture. La fourmilière commence avec 10 fourmis. Chaque fourmi peut se déplacer, à chaque tour, d'une case (choisie au hasard), ou ramasser de la nourriture . Dès qu'une fourmi ramasse de la nourriture, elle rentre directement à la fourmilière (une case spécifique du tableau) et y reste. Sinon, elle continue de chercher.

Au bout de 20 tours, la nuit tombe. Quatre chauves souris apparaissent sur le plateau (sur des emplacements choisis au hasard) et se déplacent au hasard (une case par tour). Si elles tombent sur une case avec une ou des fourmis (et qui n'est pas la case fourmilière), la ou les fourmis meurent. Ceci dure pendant 5 tours.

Au bout de 5 tours, le jour revient, les chauves souris disparaissent, et c'est reparti. De la nouvelle nourriture apparaît, et l'on compte fait les comptes suivants :

Fourmis qui meurent = MAX((Nombre de fourmis sur le plateau - Quantité de nourriture ramenée dans la fourmilière) , 0).

Fourmis qui naissent dans la fourmilière = MAX((Quantité de nourriture ramenée dans la fourmilière - Nombre de fourmis sur le plateau) *2 , 0).

Nouveau stock de nourriture dans la fourmilière = MAX((Quantité de nourriture ramenée dans la fourmilière - Nombre de fourmis sur le plateau) , 0).

Réfléchissez à une bonne architecture de programme, souple, qui puisse s'ajuster à des changements de la part du client.