R2.01-TD1-Prog

Semaine 4

Objectifs du TD/TP

Examiner la définition d'une classe.
Apprendre à écrire une classe simple et sa classe de test.

1- Examen d'une définition de classe

L'exemple utilisé est extrait de BlueJ. Il s'agit d'un distributeur de ticket du genre ticket de parking mais avec un prix unique.

Nous allons examiner ligne par ligne le code la classe donnée en annexe.

1. L'entête de la classe et les attributs (lignes 1 à 16)

   Quelles sont les deux formes de commentaire utilisées ? Leurs différences ?

   Expliquer l'utilisation des majuscules et des minuscules dans les noms ?

   La différence entre une visibilité publique et privée ?

2. Le constructeur (lignes 18 à 26)

   Par rapport à ce qui a été vu en cours, à quoi sert le constructeur ?
   Quelle syntaxe doit respecter le constructeur ?

   A quoi correspond « int ticketCost » ?

   L'utilisation de la pseudo-variable « this ».

3. Les autres méthodes

   Donner la liste des noms des méthodes

   Quelles sont les méthodes qui modifient les attributs ?

   Quelles sont les méthodes qui ne modifient pas les attributs ?

   Quelles sont les méthodes qui retournent quelque chose ?

   Dans la méthode insertMoney (ligne 55) pourquoi n'a-t-on pas écrit « this.amount » ?

2- Tester et exécuter la classe TicketMachine

Ecrire les envois de messages correspondants à l'utilisation de chacune des méthodes de la classe.

En déduire une classe de test pour la classe.

3- Amélioration de la classe TicketMachine

La première version de la classe était assez naïve et faisait confiance à l'utilisateur. Maintenant il faut faire des vérifications pour garantir que l'utilisateur entre des données qui ont du sens et le ticket ne sera imprimer que si suffisamment d'argent a été donné (la somme collectée est supérieure ou égal au prix du ticket).

Par ailleurs l'utilisateur peut avoir inséré plus que le prix du ticket, la machine doit donc lui rendre la

monnaie.

- ajouter des tests dans toutes les méthodes qui prennent un paramètre. On testera les valeurs négatives et on imprimera un message explicite.
- Modifier la méthode printTicket() pour
  - vérifier la somme collectée (amount) et imprimer un message indiquant la somme manquante,
  - n'augmenter le total que du prix du ticket et diminuer la balance que du prix.
- Ajouter une méthode `int refundBalance()` qui retourne la monnaie à rendre et remet la balance à 0.

Code de la classe TicketMachine

```
1. /**
2.  * TicketMachine models a ticket machine that issues tickets.
3.  * The price of a ticket is specified via the constructor.
4.  * It is a naive machine in the sense that it trusts its users
5.  $ to insert enough money before tying to print a ticket.
6.  * It also assumes that users enter sensible amounts.
7.  * @author David J. Barnes and Michael Kolling
8.  * modified by IB
9.  */
10.       public class TicketMachine {
11.    // The price of a ticket from this machine.
12.    private int price;
13.    // The amount of money entered by a customer so far.
14.    private int balance;
15.    // The total amount of money collected by this machine.
16.    private int total;
17.
18.    /**
19.     * Create a machine that issues tickets of the given price.
20.     * @param ticketCost : the price of the ticket
21.     */
22.     public TicketMachine(int ticketCost) {
23.          this.price = ticketCost;
24.          this.balance = 0;
25.          this.total = 0;
26.     }
27.
28.     /**
29.      * Get the ticket price
30.      * @return the price.
31.      */
32.      public int getPrice() {
33.           return this.price;
34.      }
35.
36.      /**
```

```
37.       * Get the amount of money already inserted for the next ticket.
38.       * @return the amount inserted
39.       */
40.       public int getBalance() {
41.           return this.balance;
42.       }
43.      /**
44.       * Get the total collected with all the inserted amounts
45.       * @return the total collected
46.       */
47.       public int getTotal() {
48.           return this.total;
49.       }
50.      /**
51.       * Receive an amount of money in cents from a customer.
52.       * @param  amount : the amount inserted by a customer
53.       */
54.       public void insertMoney(int amount)  {
55.             this.balance = this.balance + amount;
56.       }
57.
58.      /**
59.       * Print a ticket
60.       * Update the total collected and reduce the balance to zero.
61.       */
62.      public void printTicket()   {
63.          System.out.println("###################");
64.          System.out.println("# IUT de Vannes");
65.          System.out.println("# Ticket");
66.          System.out.println("# " + this.price + " euros.");
67.          System.out.println("###################");
68.          System.out.println();
69.
70.          // Update the total collected with the balance.
71.          this.total = this.total + this.balance;
72.          // Clear the balance.
73.          this.balance = 0;
74.       }
75.    } // end TicketMachine
```