TD – M2103

Semaine 2

Objectifs du TD/TP

Examiner la définition f'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 13)

   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 15 à 13

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

   A quoi correspond « int ticketCost » ?

   L'utilisation de la pseuso-varable « 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 48) 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.

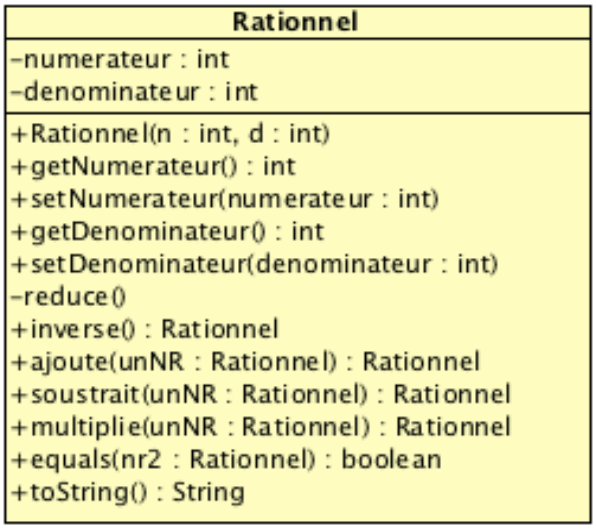
En déduire une classe de test pour la classe.

3- La classe Rationnel (à terminer en TP)

Il s'agit de définir en java une classe permettant de représenter et manipuler des nombres rationnels.
Un nombre rationnel est formé d'un numérateur et d'un dénominateur et correspond à une fraction

entière. Dans notre représentation on a décidé de mettre le signe du rationnel avec le numérateur.

Voici la spécification UML de la classe :



1) Classer les méthodes de la classe en deux catégories : celles qui modifient l'état des objets de la classe et celles qui ne modifient pas l'état des objets.

2) Définir en Java dans l'ordre :

- l'entête de la classe avec la déclaration des attributs d'instance
- le constructeur de la classe
  - Le constructeur devra vérifier que le dénominateur est >0, =0 et s'il est <0 alors faut changer le signe du numérateur.
- les méthodes d'accès aux attributs
- les méthodes de modifications des attributs
- les autres méthodes

✗ La méthode **`reduce`** est une méthode privée qui fait la réduction de la fraction en divisant le numérateur par le plus grand diviseur commun. Il faut donc savoir calculer un *pgcd*.

✗ Les méthodes **`inverse`**, **`ajoute`**, **`soustrait`**, **`multiplie`** retournent toutes en résultat un nouveau Rationnel.

✗ Il faudra également définir une classe **`TestRationnel`** qui testera la classe **`Rationnel`**.

TP – M2103

Semaine 2

Rappel pgcd par la méthode d'Euclide :

- On considère que pgcd(a, 0) = a et que pour b ≠ 0 pgcd(a,b) = pgcd(b, a mod b).
- On progresse dans l'algorithme en diminuant à chaque étape les nombres considérés par calcul du modulo.

Travail à réaliser :

- Ecrire le code Java complet de la classe `Rationnel` et la compiler
- Ecrire une classe de test
- Compiler et exécuter les tests
- Déposer le code source sur la zone de rendu (DUT INFO 1A- M2103)

```
1. /**
2.  * TicketMachine modélise un distributeur de ticket simplifié
3.  * le prix du ticket est fixé à la construction du distributeur.
4.  * Il est simplifié dans le sens où il fait confiance aux utilisateurs d'avoir
5.  * inséré suffisament d'argent avant d'essayer d'imprimer le ticket.
6.  */
7. public class TicketMachine {
8.           // le prix du ticket pour ce distributeur.
9.           private int price;
10.          // la somme qu'insèrera l'utilisateur
11.          private int balance;
12.          // la somme totale collectée par ce distributeur.
13.          private int total;
14.
15.          /**
16.           * Create a machine that issues tickets of the given price.
17.           * @param ticketCost le prix du ticket
18.           */
19.          public TicketMachine(int ticketCost)    {
20.              this.price = ticketCost;
21.              this.balance = 0;
22.              this.total = 0;
23.          }
24.
25.          /**
26.           * obtenir le prix du ticket
27.           * @return le prix du ticket.
28.           */
29.          public int getPrice()   {
30.              return this.price;
31.          }
32.
33.          /**
34.           * Obtenir la somme insérée
35.           * @return le somme inséré pour le ticket
36.           */
37.          public int getBalance()    {
38.              return this.balance;
39.          }
```

```
40.          /**
41.           * Obtenir la somme totale insérée
42.           * @return le somme totale insérée pour le ticket
43.           */
44.          public int getTotal()    {
45.              return this.total;
46.          }
47.
48.
49.          /**
50.           * Reçoit un montant de l'utilisateur.
51.           * @param amount la somme entière insérée
52.           */
53.          public void insertMoney(int amount)   {
54.              this.balance = this.balance + amount;
55.          }
56.
57.          /**
58.           * Impression du ticket
59.           * met à jour la somme totale collectée et remet la balance à 0.
60.           */
61.          public void printTicket()   {
62.              System.out.println("##################");
63.              System.out.println("# IUT de Vannes");
64.              System.out.println("# Ticket");
65.              System.out.println("# " + this.price + " euros.");
66.              System.out.println("##################");
67.              System.out.println();
68.
69.              this.total = this.total + this.balance;
70.              this.balance = 0;
71.          }
72.      } // fin TicketMachine
```