Benjamin Bourgeaux
Lucas Hélaine
Cours : 8INF957

TP2 Rapport

Question 2 :

1) Exemple de **cohésion entre classe** : Dans le projet TP2, pour la question 1, nous avons séparé les pigeons et leur affichage, de la nourriture. Dans le code suivant :

```
class SalesReport {
      void connectToDb(){ }
      void generateSalesReport() { }
      void saveAsFile() { }
      void print() { }
}
```

Cette classe peut être divisée en plusieurs classes dans différents fichiers :

```
class SalesReport {
      Options getReportingOptions() { }
      void generateSalesReport(Options o) { }
}
class ConnectToDb {
      DBconnection getDb() { }
}
class PrintStuff {
      PrintOptions getPrintOptions() { }
}
class FileSaver {
      SaveOptions getFileSaveOptions() { }
}
```

Exemple de **cohésion entre méthode** : soit la méthode suivante,

```
public static void CirconferenceCercle(){
      // Lecture du rayon
      System.out.print("Entrer le rayon du cercle: ");
      double rayon = Keyboard.readDouble();
      while(Keyboard.error() || rayon < 0){
            System.out.print("Entrer le rayon du cercle: ");
            rayon = Keyboard.readDouble();
      }

      // Calcul de la circonference
```

```
      double circonference = 2 * Math.PI * rayon;

      // Affichage de la circonference
      System.out.println("La circonference est " + circonference);
```

Elle peut être divisée en plusieurs méthodes séparées augmentant la réutilisation et facilitant la maintenance :

```
// Lecture du rayon
public static double lireRayon(){
      System.out.print("Entrer le rayon du cercle: ");
      double rayon = Keyboard.readDouble();
      while(Keyboard.error() || rayon < 0){
            System.out.print("Entrer le rayon du cercle: ");
            rayon = Keyboard.readDouble();
      }
      return rayon;
}

// Calcul de la circonference
public static double circonferenceCercle(double rayon){
      return 2 * Math.PI * rayon;
}

// Affichage de la circonference
public static void afficheCirconference(double circonference){
      System.out.println("La circonference est " + circonference);
}
```

2) méthode CashRegister() : purchase et payment
méthode recordPurchase() : purchase et amount
méthode receivePayment() : payment, dollars, quarters, dimes, nickels, pennies
méthode giveChange() : payment, purchase, change

CashRegister et recordPurchase ont un attribut en commun
CashRegister et receivePayment ont un attribut en commun
CashRegister et giveChange ont deux attributs en commun
recordPurchase et giveChange ont un attribut en commun
receivePayment et giveChange ont un attribut en commun
donc Q = 5

recordPurchase et receivePayment n'ont pas d'attribut en commun
donc P = 1
LCOM = |P| - |Q| = 1 - 5 = -4
Or si LCOM < 0, LCOM = 0, donc LCOM = 0.

3) La classe CashRegister n'est pas cohésive car "une classe est cohésive si ses méthodes agissent sur le même ensemble de données. Les méthodes sont donc reliées entre elles." [Toure, Indicateur de qualité pour les systèmes orientés objet : Vers un modèle unifiant plusieurs métriques]

Les variables final sont "static" et disponibles pour toutes les méthodes alors qu'elles ne sont utilisées que par une méthode.

4) Voici le code modifié

```
/* ---------------------------- CashRegister.java
------------------------------------------------ */
/**
A cash register totals up sales and computes change due.
 */
public class CashRegister
{
    private double purchase;
    private double payment;
    /**
Constructs a cash register with no money in it.
     */
    public CashRegister()
    {
      purchase = 0;
      payment = 0;
    }
    /**
Records the purchase price of an item.
@param amount the price of the purchased item
     */
    public void recordPurchase(double amount)
    {
      purchase = purchase + amount;
    }
    /**
Processes the payment received from the customer.
@param dollars the number of dollars in the payment
@param quarters the number of quarters in the payment
@param dimes the number of dimes in the payment
@param nickels the number of nickels in the payment
@param pennies the number of pennies in the payment
     */
    public void receivePayment(int dollars, int quarters,
            int dimes, int nickels, int pennies)
    {
      final double QUARTER_VALUE = 0.25;
```

```
      final double DIME_VALUE = 0.1;
      final double NICKEL_VALUE = 0.05;
      final double PENNY_VALUE = 0.01;
      payment = dollars + quarters * QUARTER_VALUE + dimes *
DIME_VALUE
                  + nickels * NICKEL_VALUE + pennies * PENNY_VALUE;
   }
   /**
Computes the change due and resets the machine for the next
customer.
@return the change due to the customer
    */
   public double giveChange()
   {
      double change = payment - purchase;
      purchase = 0;
      payment = 0;
      return change;
   }
}
```

SOURCES

http://www.iro.umontreal.ca/~pift1015/cours/art2_print.pdf
http://depot-e.uqtr.ca/1529/1/030043133.pdf
http://savoirs.usherbrooke.ca/bitstream/handle/11143/1606/MR83692.pdf?sequence=1