M1 informatique : Design Patterns
TD n°1

On considère les classes suivantes :

```
public class Movie {

        public static final int CHILDRENS = 2;
        public static final int REGULAR = 0;
        public static final int NEW_RELEASE = 1;

        private String _title;
        private int _priceCode;

        public Movie(String title, int priceCode) {
                _title = title;
                _priceCode = priceCode;
        }

        public int getPriceCode() {
                return _priceCode;
        }

        public void setPriceCode(int arg) {
                _priceCode = arg;
        }

        public String getTitle() {
                return _title;
        }
}

public class Rental {

        private Movie _movie;
        private int _daysRented;

        public Rental(Movie movie, int daysRented) {
                _movie = movie;
                _daysRented = daysRented;
        }
        public int getDaysRented() {
                return _daysRented;
        }
        public Movie getMovie() {
                return _movie;
        }
}

import java.util.Vector;
import java.util.Enumeration;

public class Customer {

        private String _name;
        private Vector<Rental> _rentals = new Vector<Rental>();

        public Customer(String name) {
                _name = name;
        }
        public String getName() {
                return _name;
```

```
    }
    public void addRental(Rental arg) {
            _rentals.addElement(arg);
    }
    public String statement() {
       double totalAmount = 0;
       int frequentRenterPoints = 0;
       Enumeration<Rental> rentals = _rentals.elements();
       String result = "--\nRental Record for " +  getName() + "\n";
       while (rentals.hasMoreElements()) {
          double thisAmount = 0;
          Rental each =  rentals.nextElement();

          //determine amounts for each line

          switch (each.getMovie().getPriceCode()) {
             case Movie.REGULAR:
                thisAmount += 2;
                if (each.getDaysRented() > 2)
                   thisAmount += (each.getDaysRented() - 2) * 1.5;
                   break;
             case Movie.NEW_RELEASE:
                thisAmount += each.getDaysRented() * 3;
                break;
             case Movie.CHILDRENS:
                thisAmount += 1.5;
                if (each.getDaysRented() > 3)
                   thisAmount += (each.getDaysRented() - 3) * 1.5;
                break;
          }

          // add frequent renter points
          frequentRenterPoints ++;
          // add bonus for a two day new release rental
          if ((each.getMovie().getPriceCode() == Movie.NEW_RELEASE) &&
             each.getDaysRented() > 1) frequentRenterPoints ++;

          //show figures for this rental
          result += "\t" + each.getDaysRented() +
                    "\t" + each.getMovie().getTitle() +
                    "\t" + String.valueOf(thisAmount) + "\n";
          totalAmount += thisAmount;
       }

       //add footer lines
       result += "Amount owed is " + String.valueOf(totalAmount) + "\n";
       result += "You earned " +  String.valueOf(frequentRenterPoints) +
                 " frequent renter points\n";
       return result;

    }
}
```

— Il s'agit de réécrire le code proprement dans le but de pouvoir générer un état du compte client en html sans dupliquer le code qui génère l'état en ascii en suivant les opérations dans le fichier `Refactoring-Presentation-from-JavaOne.pdf`