DAT159 Refactoring - Oblig 01

By Kristoffer-Andre Kalliainen (181192)

Part 1 - Refactoring steps

Extracting the switch case to its own method `determineAmount()`, but before extracting the method I extracted the `each.getMovie().getPricecode()` and `each.getDaysRented()` into separate variables.

> *from*

```
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;
}
```

> *to*

```
int priceCode = movie.getPriceCode();
int daysRented = each.getDaysRented();
double thisAmount = determineAmount(priceCode, daysRented);


private double determineAmount(int priceCode, int daysRented) {
    double thisAmount = 0;
    switch (priceCode) {
        case Movie.REGULAR:
            thisAmount += 2;
            if (daysRented > 2)
                thisAmount += (daysRented - 2) * 1.5;
            break;
        case Movie.NEW_RELEASE:
            thisAmount += daysRented * 3;
            break;
        case Movie.CHILDRENS:
            thisAmount += 1.5;
            if (daysRented > 3)
```

```
                thisAmount += (daysRented - 3) * 1.5;
            break;
    }
    return thisAmount;
}
```

Then I moved the method over to the Movie class and created three subclasses that was each case in the switchcase `Children`, `Regular` and `NewRelease`. I made the method abstract in Movie class and then override the method in childclasses. In each class I added the code corresponding to the code inside each case in the switch.

```
Movie movie = each.getMovie();
String title = movie.getTitle();
int priceCode = movie.getPriceCode();
double thisAmount = movie.determineAmount(daysRented);


public abstract class Movie {

    [...]

    abstract double determineAmount(int daysRented);

    class Children extends Movie {

        public Children(String title, int priceCode) {
            super(title, priceCode);
        }

        @Override
        double determineAmount(int daysRented) {
            double thisAmount = 1.5;
            if (daysRented > 3)
                thisAmount += (daysRented - 3) * 1.5;
            return thisAmount;
        }
    }

    class Regular extends Movie {

        public Regular(String title, int priceCode) {
            super(title, priceCode);
        }

        @Override
        double determineAmount(int daysRented) {
            double thisAmount = 2;
            if (daysRented > 2)
                thisAmount += (daysRented - 2) * 1.5;
```

```
            return thisAmount;
        }
    }

    class NewRelease extends Movie {

        public NewRelease(String title, int priceCode) {
            super(title, priceCode);
        }

        @Override
        double determineAmount(int daysRented) {
            return daysRented * 3;
        }
    }
}
```

Extracted the frequent renterpoints lines into its own method called `getFrequentRenterPoints()`.

> *from*

```
// add frequent renter points
frequentRenterPoints ++;
// add bonus for a two day new release rental
if ((priceCode == Movie.NEW_RELEASE) &&
        daysRented > 1) frequentRenterPoints ++;
```

> *to*

```
frequentRenterPoints = getFrequentRenterPoints(frequentRenterPoints, priceCode, daysRented);


private int getFrequentRenterPoints(int frequentRenterPoints, int priceCode, int daysRented) {
    // add frequent renter points
    frequentRenterPoints ++;
    // add bonus for a two day new release rental
    if ((priceCode == Movie.NEW_RELEASE) &&
            daysRented > 1) frequentRenterPoints ++;
    return frequentRenterPoints;
}
```

Extracting the movie variable `each.getMovie()`

```
Movie movie = each.getMovie();
String title = movie.getTitle();
```

```
int priceCode = movie.getPriceCode();
frequentRenterPoints += getFrequentRenterPoints(frequentRenterPoints, priceCode, daysRented);
```

Moving the getFrequentRenterPoints from Customer class to Movie. For the special case when its an Rew Release,

Im doing a override of the method and check for the two-days rented bonus.

> *Customer.class*

```
 frequentRenterPoints += movie.getFrequentRenterPoints(frequentRenterPoints, priceCode, daysRented);
```

> *Movie.class*

```
public int getFrequentRenterPoints(int frequentRenterPoints, int priceCode, int daysRented) {
    return ++frequentRenterPoints;
}
```

> *NewRelease.class*

```
@Override
public int getFrequentRenterPoints(int frequentRenterPoints, int priceCode, int daysRented) {
    // add frequent renter points
    frequentRenterPoints++;
    // add bonus for a two day new release rental
    if (daysRented > 1) frequentRenterPoints++;
    return frequentRenterPoints;
}
```

Then removing the constant in the top of the class

> *Deleting*

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

Extracting the footer lines to its own method.

> *from*

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

> *to*

```
result += getFooterLines(totalAmount, frequentRenterPoints, result);

private String getFooterLines(double totalAmount, int frequentRenterPoints, String result) {
    //add footer lines
    result += "Amount owed is " + String.valueOf(totalAmount) + "\n";
    result += "You earned " + String.valueOf(frequentRenterPoints) +
            " frequent renter points";
    return result;
}
```

Extracting the result string to its own method

> *from*

```
result += ("\t" + title + "\t" + String.valueOf(thisAmount) + "\n");
```

> *to*

```
result += printFiguresForRental(result, title, thisAmount);


private String printFiguresForRental(String result, String title, double thisAmount) {
    return result + ("\t" + title + "\t" + String.valueOf(thisAmount) + "\n");
}
```

Final Result

> `Customer.java`

```
package net.jeremykendall.refactoring.videostore;

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

public class Customer {
    private String _name;
    private Vector _rentals = new Vector();

    public Customer(String name) {
        _name = name;
    }

    public String statement() {
        double totalAmount = 0;
        int frequentRenterPoints = 0;
```

```
        Enumeration rentals = _rentals.elements();
        String result = "Rental Record for " + getName() + "\n";
        while (rentals.hasMoreElements()) {
            Rental each = (Rental) rentals.nextElement();
            int daysRented = each.getDaysRented();
            Movie movie = each.getMovie();

            int priceCode = movie.getPriceCode();
            frequentRenterPoints += movie.getFrequentRenterPoints(frequentRenterPoints, priceCode, daysRente
d);

            String title = movie.getTitle();
            double thisAmount = movie.determineAmount(daysRented);
            result += printFiguresForRental(result, title, thisAmount);
            totalAmount += thisAmount;
        }
        result += getFooterLines(totalAmount, frequentRenterPoints, result);
        return result;
    }

    private String printFiguresForRental(String result, String title, double thisAmount) {
        return result + ("\t" + title + "\t" + String.valueOf(thisAmount) + "\n");
    }

    private String getFooterLines(double totalAmount, int frequentRenterPoints, String result) {
        return result
                + "Amount owed is " + String.valueOf(totalAmount) + "\n"
                + "You earned " + String.valueOf(frequentRenterPoints)
                + " frequent renter points";
    }

    public void addRental(Rental arg) {
        _rentals.addElement(arg);
    }

    public String getName() {
        return _name;
    }
}
```

Movie.java

```
package net.jeremykendall.refactoring.videostore;

public abstract class Movie {

    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 _priceCode) {
        this._priceCode = _priceCode;
    }

    public String getTitle() {
        return _title;
    }

    public abstract double determineAmount(int daysRented);

    public int getFrequentRenterPoints(int frequentRenterPoints, int priceCode, int daysRented) {
        return ++frequentRenterPoints;
    }


    class Children extends Movie {

        public Children(String title, int priceCode) {
            super(title, priceCode);
        }

        @Override
        public double determineAmount(int daysRented) {
            double thisAmount = 1.5;
            if (daysRented > 3)
                thisAmount += (daysRented - 3) * 1.5;
            return thisAmount;
        }
    }

    class Regular extends Movie {

        public Regular(String title, int priceCode) {
            super(title, priceCode);
        }

        @Override
```

```
        public double determineAmount(int daysRented) {
            double thisAmount = 2;
            if (daysRented > 2)
                thisAmount += (daysRented - 2) * 1.5;
            return thisAmount;
        }
    }

    class NewRelease extends Movie {

        public NewRelease(String title, int priceCode) {
            super(title, priceCode);
        }

        @Override
        public double determineAmount(int daysRented) {
            return daysRented * 3;
        }

        @Override
        public int getFrequentRenterPoints(int frequentRenterPoints, int priceCode, int daysRented) {
            // add frequent renter points
            frequentRenterPoints++;
            // add bonus for a two day new release rental
            if (daysRented > 1) frequentRenterPoints++;
            return frequentRenterPoints;
        }
    }
}
```

Rental.java

```
package net.jeremykendall.refactoring.videostore;

public class Rental {
    private Movie _movie;
    private int _daysRented;

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

    public Movie getMovie() {
        return _movie;
    }
```

```
    public int getDaysRented() {
        return _daysRented;
    }
}
```

Part 2 - Measure SLOC and McCabe's Cyclomatic Complexity on `Customer.statement()` method

The SLOC - Source lines of code before was 43 lines. After the refactoring the number of lines is now 21.

So by refactoring we reduced the lines of code with 49%.

The Cyclomatic Complexity is measured by the number of linearly independent paths through a program's source code.

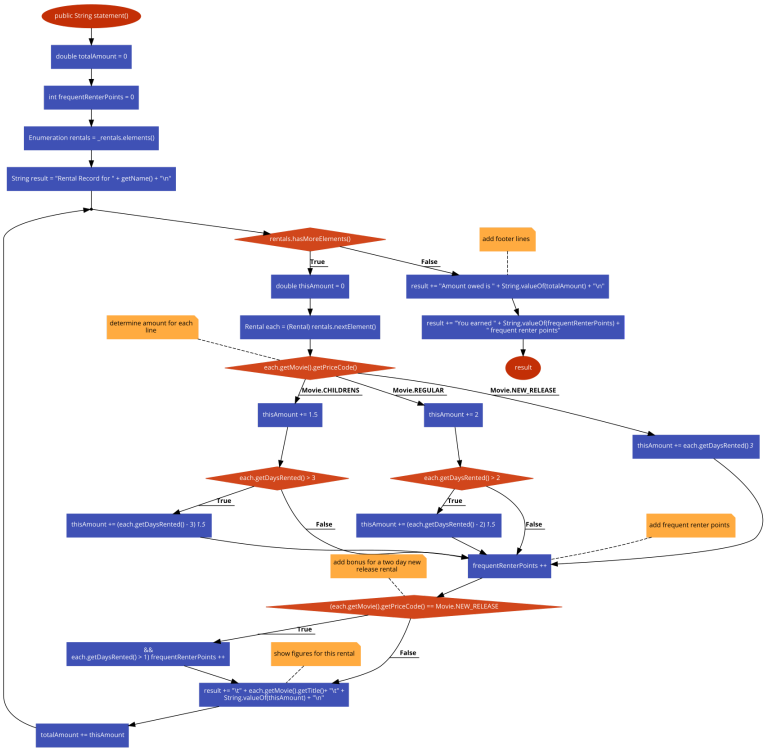
The complexity **M** is defined as `M = E − N + 2` for a single method.

Where

- E = the number of edges of the graph
- N = the number of nodes of the graph

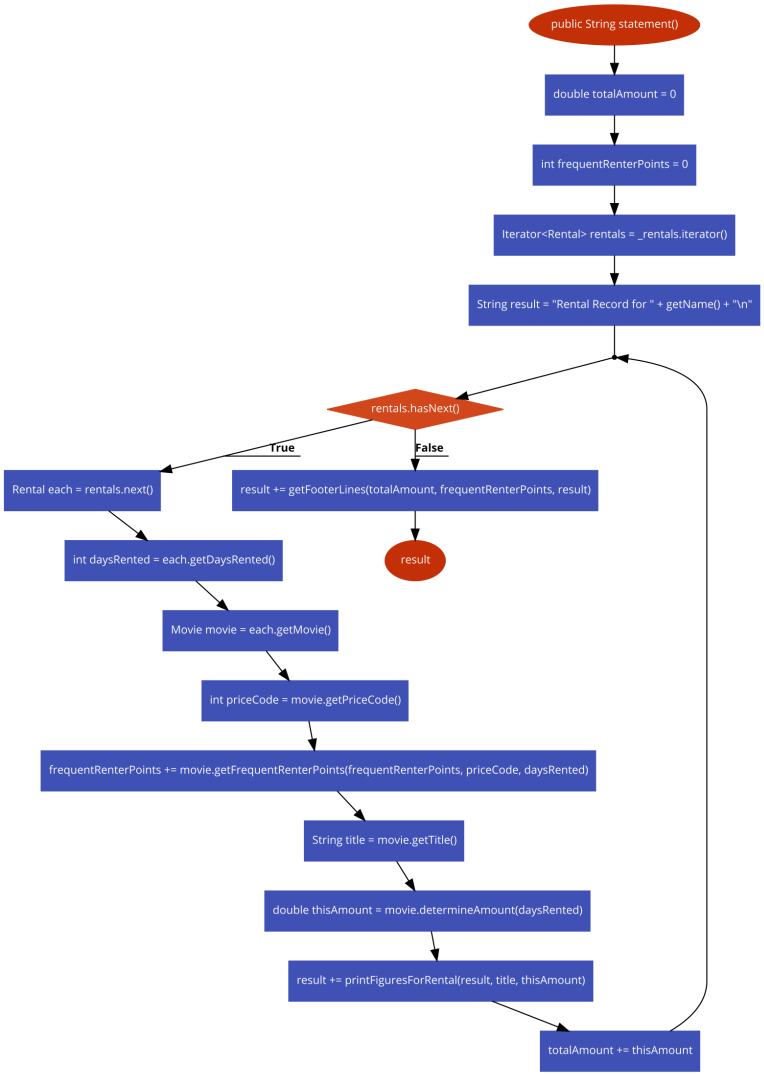
Before the refactoring the Complexity was 9.

30 number of edges - 23 number of nodes + 2 = 9



After the refactoring the complexity was 2.

17 number of edges - 17 number of nodes + 2 = 2


public String statement()
double totalAmount = 0
int frequentRenterPoints = 0
Iterator<Rental> rentals = _rentals.iterator()
String result = "Rental Record for " + getName() + "\n"
rentals.hasNext()
True
False
Rental each = rentals.next()
result += getFooterLines(totalAmount, frequentRenterPoints, result)
int daysRented = each.getDaysRented()
result
Movie movie = each.getMovie()
int priceCode = movie.getPriceCode()
frequentRenterPoints += movie.getFrequentRenterPoints(frequentRenterPoints, priceCode, daysRented)
String title = movie.getTitle()
double thisAmount = movie.determineAmount(daysRented)
result += printFiguresForRental(result, title, thisAmount)
totalAmount += thisAmount