ASSIGNMENT 2

Team: Jibin Prince, Michael Muruthi, Richard Padilla, Karl Dill, Francisco E. Alderete

Software Refactoring:

Improve design of existing code without changing what it does
Making code easier to change
More loosely coupled
More cohesive modules
More comprehensive

1. At least 3 method extraction operations:
   a. Switch statement extracted from Customer to setPriceCode() method in Movie class
   b. Price calculation extracted from switch statement into Price subclasses
   c. frequentRenterPoints becomes own method: getFrequentRenterPoints()
   d. calculateFrequent has been extracted out of the statement() method into the Price classes
   e. Extract print statement into htmlRentalReciept() method
   f. Extract bonus rental points calculation out into a getCharge() method within Price.java

2. At least 3 creations of new classes:
   Rationale: Want to have separation of concerns on 3 types of movies. If there is another Movie type we want to add, we can just add a new subclass to handle it. Furthermore, we also wanted to separate the concern of the Price from the Movie, so we chose to make an Abstract Price class with 3 concrete Price subclasses: Price.java, ChildrensPrice.java, RegularPrice.java, NewReleasePrice.java.

3. At least 3 moving operations:
   a. Moved switch statement code to Movie class, since that calculation shouldn't be done by the customer
   b. Moved rental price calculation in switch statement code to the Price superclass, since the price calculation is unique for each of the different movie categories, but very similar to the point that it only makes sense to have once
   c. Bonus renter point calculations got moved out of the Customer.java class into specific Price subclasses.

4. At least 3 renaming operations
   a. Renamed statement() method to htmlRentalReciept(). Name was ambiguous and did not communicate what it actually does or how to use it. Method names should comply with a naming convention and should be clear about what it does.
   b. Renamed several magic numbers from Movie.java. Using the magic numbers directly in the code can lead to an anti-pattern that makes the code difficult to understand and maintain.
   c. Renamed getFrequentRenterPoints() to getRentalPointValue(). This better communicates what it actually does or how to use it.

    d. Renamed the variable names "each" to rental.

5. 1-2 replacements of data types
    a. Changed Vector to List<Rentals>. Vector class can be considered as obsolete or legacy class which is deprecated, less performant due to overhead of locking whether synchronization is required or not. It's slow as it is synchronized, and only a single thread can obtain a lock on an operation, making other threads wait until that lock is released. ArrayList on the other hand is much faster as it is not synchronized, and multiple threads can operate on it at the same time.
    b. Changed Enumeration data type into Rental type

6. A Main Method was implemented to test the program.
7. New Print statement added was htmlRentalReceipt()