Object Oriented Analysis and Design:

Description:

Designed a program to support the management of a movie rental place by refactoring and applying various design patterns. The program supports different types of items including movies, video games, music CDs, xbox, ps4 and books. New type of items can also be included without making changes to the existing code. The program also supports renting, selling of items and provides the final transaction amount to be paid after applying discounts to eligible customers.

Classes:

- Driver(main class)
- ItemOptionFactory
- ItemRenter
- Buyer
- BookingDetails
- ChooseItem
- Customer
- Rental
- Items:
    - Book – FantasyBook, MysteryBook, ScienceFictionBook
    - Movie- ActionMovie, ClassicMovie, AnimationMovie
    - MusicCD – JazzMusic, PopMusic
    - Xbox
    - PS4
    - VideoGame

Class flow:

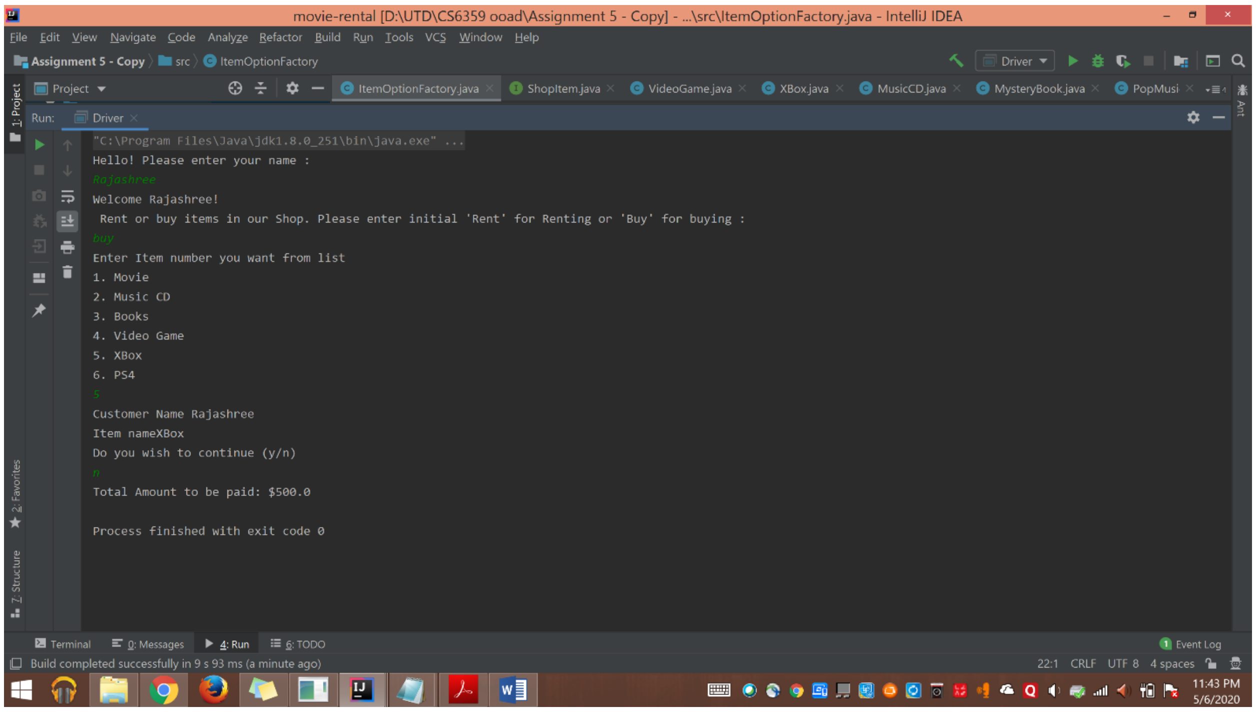
Driver will be the main file which should be executed first and it gives the option for customer to rent or buy items and then customer should choose the item desired. Based on the option, it gets redirected to ItemOptionFactory class where the corresponding child class is instantiated.

If rent is chosen, then the number of days rented will be inquired. Then based on the items and buy or rent option, the final amount will be calculated and given back to the Driver class. Then based on the type of customer, the discount value is applied to the final amount.

Refactoring in the large:

- Created the ItemOptionFactory class which acts as a **Factory Design Pattern** to instantiate different child class objects during run time.
- Buying and selling items takes place using different carts
- **Strategy design pattern** has been used along with simple factory design pattern to instantiate and run different strategies at run time. Each of those subclasses(in movie, book, cd items) will be the strategy classes.
- Additional items can be separately added as classes which inherit the ChooseItem interface and its concrete instantiation should be put in the Factory class. This helps us to add items without changing the base code.

Sample Output:


movie-rental [D:\UTD\CS6359 ooad\Assignment 5 - Copy] - ...\src\ItemOptionFactory.java - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Assignment 5 - Copy 〉 src 〉 ItemOptionFactory
Project
ItemOptionFactory.java
ShopItem.java
VideoGame.java
XBox.java
MusicCD.java
MysteryBook.java
PopMusi
Run: Driver
Terminal
0: Messages
4: Run
6: TODO
Event Log

```
"C:\Program Files\Java\jdk1.8.0_251\bin\java.exe" ...
Hello! Please enter your name :
Rajashree
Welcome Rajashree!
 Rent or buy items in our Shop. Please enter initial 'Rent' for Renting or 'Buy' for buying :
buy
Enter Item number you want from list
1. Movie
2. Music CD
3. Books
4. Video Game
5. XBox
6. PS4
5
Customer Name Rajashree
Item nameXBox
Do you wish to continue (y/n)
n
Total Amount to be paid: $500.0

Process finished with exit code 0
```

Build completed successfully in 9 s 93 ms (a minute ago)

Scope:

- In future, Discount and Payment methods can be added by implementing Decorator design pattern & State Design pattern. This is a way to improvise the existing code without changing the base code