# Object Oriented Analysis And Design

## Assignment 3 Write Up

Amith Gopal, Akriti Kapur and Prashanth Thipparthi,

**Problem Statement:**

Write an object-oriented program that implements the problem domain and does the following:

Simulates the activity of the rental store for 35 days (34 nights). On each day, a random number of customers will visit the store as long as there are tools to rent. Each customer will create one Rental that follows the rules of their associated type before they leave the store. That is, no customer will show up and then leave without making a rental. Note: if the store has less than 3 tools, then a Business customer will NOT arrive (as they wouldn't be able to create a Rental that follows their rules). As soon as the store has zero tools, customers will magically stop arriving until tools are once again available. At the end of the simulation, the program will produce a report that includes the following information:

- the number of tools currently in the store along with a list of their names
- the amount of money the store made during the 35 days (including any rentals that occurred on the 35th day)
- a list of all the completed rentals including which tools were rented by which customer for how many days

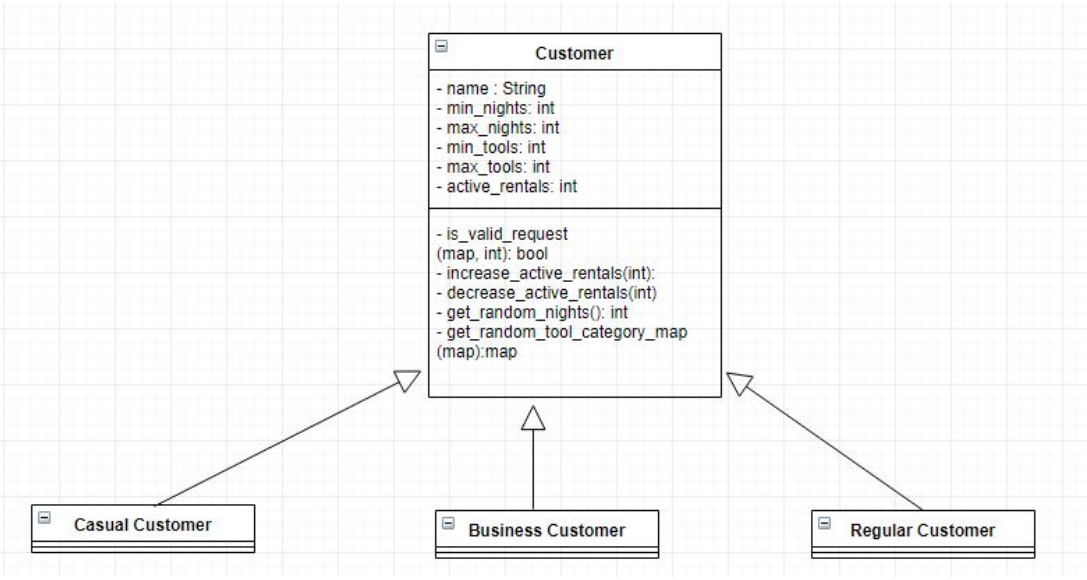
  along with the total amount of that rental
- a list of all the active rentals that includes all of the information listed in the previous bullet

- A customer can have more than one active rental. That is, they can show up on day 1 and rent 1 tool for 5 nights. They can then show up on day 2 and rent another tool for 4 nights. As long as they do not have more than 3 tools rented, they are allowed to have multiple rentals.
- Returns occur at the beginning of the day before the store opens for business. A tool rented for one night is available to customers the very next day; that's because the customer rented the tools for one night, used it, and got it back to the store early the next morning.
- Your program should be single-threaded; you do not need to handle the case of multiple customers trying to rent tools concurrently.

**Design:**

The project design can be divided into two modules, **Customer** and **Rental Store,** which have following classes:

**Customer:**




Customer Base class has different attributes like the name of the customer, minimum and maximum number of tools he can rent, minimum and maximum number of nights he can rent and also his active rentals.

Each type of customer derives from the Base Customer class and implements the **is_valid_request method** to check whether according to the customer type whether it is a valid rental request or not.

Main responsibilities of the customer class include:

1. Checking for validity of a request
2. Creating a random request consisting of choosing random categories and

   random nights
3. Maintaining the active rentals per customer

**Rental Store Class:**

Rental Store has an inventory in which different categories (sections) of tools are maintained. The reason for including an inventory in rental store is that a rental store can have/deal with multiple inventories to know/rent the tools availability.

Rental Store is also responsible for maintaining the inventory which includes checking for the availability of a tool when a rental request arrives, renting the tools and also updating the tools availability when

the tools are returned,  maintaining a list of “active rentals”, “completed rentals”, “customer list” and total amount of revenue.

Main responsibilities of the Store class include:

1. Processing the Renting tools request
2. Updating the rentals on the return date of the rental
3. Holding information regarding active rentals and completed rentals
4. Creating a rental object and storing it in Rental Object list

**Inventory Class:**

Inventory class has different sections(Categories) of tools in it. It has APIs to get the tool information and update the tool information while rental is done.

Main responsibilities of the Inventory class include:

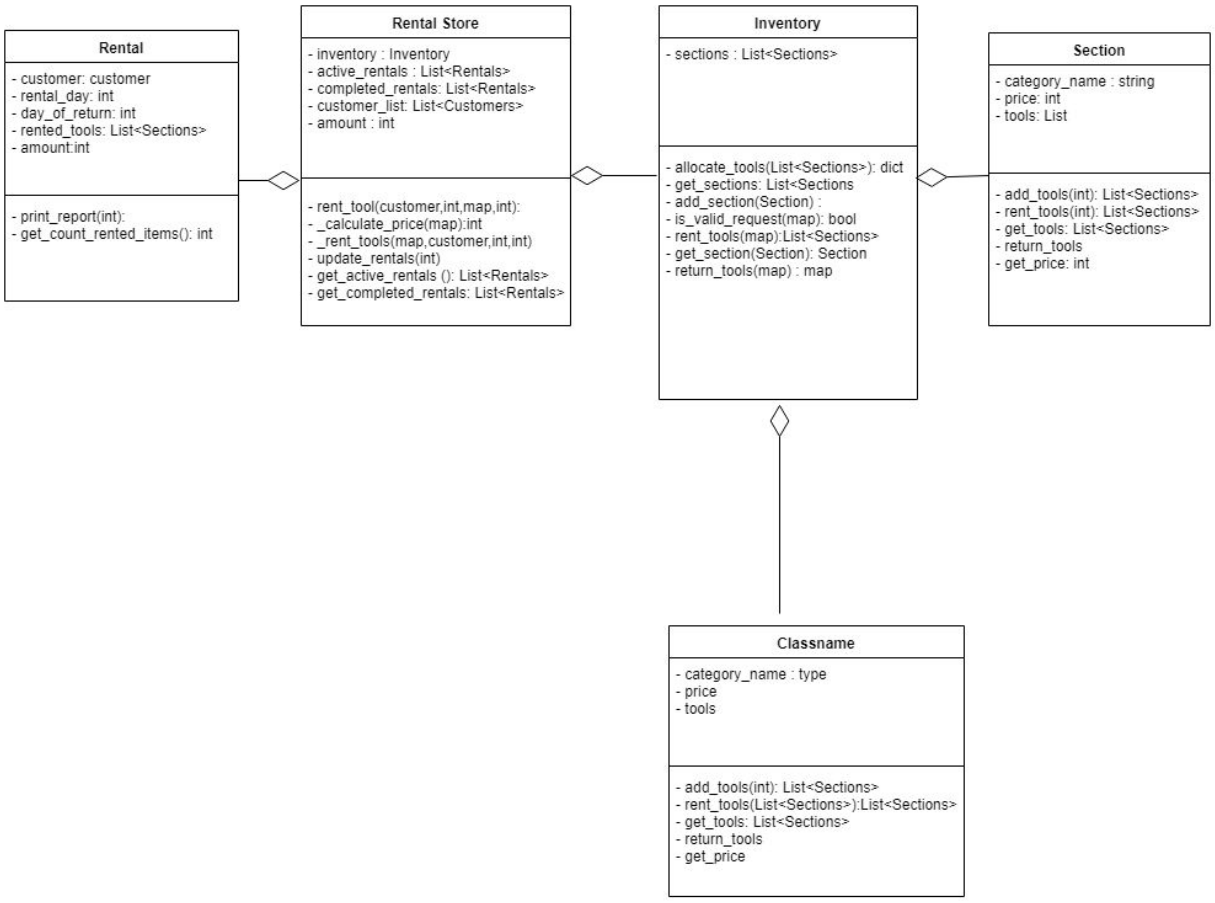
1. Verifying the availability of tools before processing request.
2. Retrieving the tools information
3. Updating the tool information.

**Rental Class**

In the **Rental** class, we have attributes to store all the information related to a particular rental like the customer involved in the rental, rental day, day of return, tools rented in the rental, total amount of the rental.

Main responsibilities include:

1. Storing of Transaction information
2. Displaying the Transaction information




**Section Class:**

**Section** class has various attributes like the name of the category,  price of the tools in the category, Number of tools in the category

Main responsibilities of this class includes:

1. Adding tools to the section - called during init time by the Inventory object
2. Rent tools from this section
3. Returning tools to this section
4. Returning the price of the tools in this section.

**Relationship between classes:**

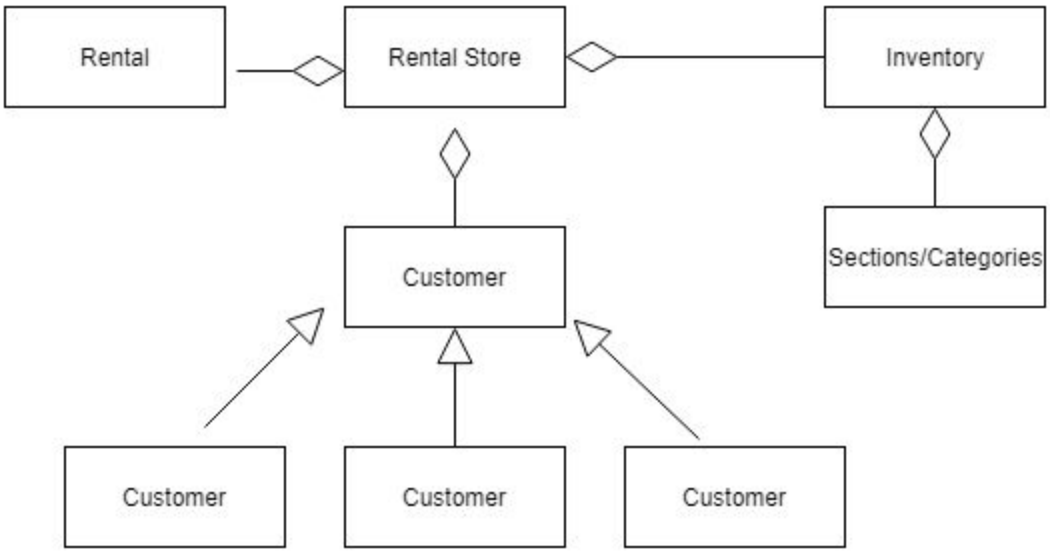
**Rental Workflow:**

1. when a customer wants to create a rental, a customer object is randomly selected from the available customers list and the no of tools required and days within the customer's eligibility range.

2. A rental request is made with the customer object, tools required and number of days.

3. When a request arrives to the rental store, rental store will check the availability of the tool in the store and updates the inventory and rents the tool to the customer.

4.When the customer returns the tool on a particular day morning it updates the inventory with appropriate details and make the tools available for rental.




**Simulation workflow:**

We run the simulation in a while loop, where in each iteration we will be doing the following tasks.

1. Increment the day
2. Check the active rentals list to check if any rentals has to be closed.
3. Report all the rentals closed at the day
4. Select a random customer from the customer list
5. Select the random number of tools and days according to the customer class selected
6. Create a customer request
7. Process request by calling the rent method in the Rental store.
8. Print all the rentals for the day.

If simulation has to run we have execute the “simulation.py” file.

**Addressing the Object Oriented Design Principles:**

**Encapsulation:**

The data and the related methods are encapsulated in all the classes to achieve high cohesion and loose coupling.

**Data Abstraction:**

The access to attributes is restricted in various classes by making them as private so that we can access only by using the methods in the respective class.

**Inheritance:**

In the design various customer classes derive from the Base Customer class and override the variable behaviour such as the way each type of customer rent the tools.

**Polymorphism:**

The inheritance relationship between the Customer Base class and the various types of derived Customer classes helps us to deal with all the customer objects in the similar manner while processing the transaction in the rental Store Class.

**High Cohesion and loose coupling:**

In all the classes and the methods only the tasks which are closely related are encapsulated to acheive the high cohesion.For example, Rental Store class has methods only to renting the tools.

The way the Rental Store handles the tools management using the Inventory class shows us the loose coupling the design.

**Delegation:**

It is included in the design by including the Inventory class in the Section class, where adding tools to the repository and updating the tools information is delegated to the section class.

**Simulation Output:`**

**The simulation output is stored in the “simulation_output.txt” file in the github repository at the below link.**

**https://github.com/AkritiKapur/OOAD/blob/master/Assignment03/simulation_output.txt**