Problem Domain: Hardware Rental Store

For this homework, you will design an object-oriented program to model the following problem domain.

You are responsible for the point of sale system for a new hardware rental store chain. At present, the store has one location.

The hardware rental store has a catalog of 24 different tools to rent, spread across 5 different categories (Painting, Concrete, Plumbing, Woodwork, Yardwork). Each tool has a unique name (e.g. “Paint Tool 1”) and belongs to a specific category; the price per day to rent a tool varies by category. You may decide on the number of tools in each category (at least 2) and the pricing of daily rentals by category.

When a customer rents a tool, they may optionally add (1) an extension cord, (2) an accessory kit, and (3) a protective gear package. Each of these options has a different cost by category. The cost of an option is on a per rental basis (not per day). You may decide on the pricing of the options. You do not need to keep track of inventory for options, any number of options are always available for valid rentals.

Customers can rent a tool for up to 7 nights. (A tool rented for 7 nights on a Monday must be returned by the following Monday. A tool rented for "one day" would, for example, be rented on a Tuesday and returned the next morning before the rental store opens for business on Wednesday.) Customers can have at most three tools rented at any one time.

Presently, the store has 12 regular customers; each customer has a unique name and is associated with one of three types. *Casual* customers rent one or two tools for one or two nights. *Business* customers always rent three tools for seven nights. *Regular* customers will rent one to three tools each time they visit for 3 to 5 nights. You can decide the mix of customers but should ensure there is at least one of each type.

Each time a customer comes into the store, a rental record is created that will keep track of what tools they rented and how many nights they will keep the tools. A customer rents their set of tools and returns them all at the same time. They will NOT, for example, rent three tools and then return one after 2 days, the second after 5 days, and the third after seven days. They will instead return all of the tools they rented at the same specified time. (That means, for instance, that a customer will never be late in returning their tools.)

Also, for each tool a customer rents, they will randomly add zero to six options to the rental. For example, they may rent Paint Tool 1, and then randomly decide to add two extension cords and three protective gear packages.

The store keeps track of the existing rental records along with the current inventory of the store. As such, when it has zero active rentals, there will be 24 tools in its inventory. When it has zero tools in its inventory, it will have multiple active rentals that between them account for all 24 tools.

Finally, a customer pays up front for their rental. If, for example, a customer rents three Paint tools for three nights at a price of $3 per night, they will pay the store $27 dollars before they leave the store with their three tools. If they decide to add any options to the tools, that cost will be added to the rental fee.

Assignment

Write an object-oriented program in Java 8 or later that implements the problem domain and does the following:

- Simulate 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 Record 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.
- The program should print the following information (which you should capture to a text out file):
    - At the beginning of each day, print the following (in a concise format):
        - Day number
        - A count and list of all completed rentals including which tools and options were rented by which customer, for how many days, along with the total fee for that rental
        - A count and list of all active rentals, including which tools are rented by which customer
        - A count and list of all tools left in the store and their names
        - The amount of money the store made that day
    - At the end of the simulation, print (in a concise format):
        - The total number of completed rentals (including any rentals that occurred on the 35th day), overall and by type of customer (casual, business, regular)
        - The total money the store made for the 35 day period
- 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.
- The purpose of this assignment is NOT to meet the requirements by any means necessary. A program that does the simulation above and produces the requested output but makes use of

only structured programming techniques (i.e. no objects, just data structures and a main program) will receive zero points (for the whole assignment). Only object-oriented programs that show good use of abstraction, encapsulation and polymorphism and meet the above requirements will be able to get full credit for the program.

- Further, you should select and use at least three distinct OO patterns that we have discussed in your design. Typical uses might include factory (for instantiating tools), decorator (for adding options), observer (for recording events that will go to output). You may see other opportunities for other OO pattern use. Document and comment clearly where patterns are applied.

Grading Rubric

As always, your OO code should be well structured and commented, and citations and/or URLs should be present for any code taken from external sources. Comments should focus on what is being done in the code, not on how, unless the method in question is unclear or obscure. You may not directly use code developed by other student teams, although you may discuss approaches to the problems, and may wish to credit other teams (or class staff) for ideas or direction.

The submission is a GitHub Repository URL. The Repo must contain:

- Running, commented OO code for the simulation in Java 8 or later
- A README Markdown file with the names of team members and any special instructions to run the code (graders may request demonstrations)
- A text out file containing a capture of complete output from a typical simulation run
- A PDF containing
    - Program title and names of team members
    - Language and environment used for development
    - Text description of program design, including any assumptions you have made about specifications that may not be complete
    - A full UML Class diagram that shows classes and relationships from your design (include data attributes and methods in the class diagram)

Homework/Project 3 is worth **75 points** as follows. There is also an additional **20 point** extra credit opportunity (see below).

- 5 Points – README file (-1 or -2 for incomplete or missing elements)
- 10 Points – PDF with design and assumptions discussion and UML class diagram (-1 to -2 for poor quality in elements, -5 for missing UML diagram or discussion)
- 15 Points – Use of at least 3 patterns in simulation design (-1 or -2 for implementation issues or poor commenting per pattern, -5 if a pattern is missing)
- 15 Points – Execution as evidenced by text-based out file (-2 or -4 for missing expected functionality or output)
- 30 Points – Code quality (-1, -2, -3 for issues including poor commenting, procedural style code, or logic errors; -30 if not an object-oriented solution as requested).

Any UML tool can be used to create the UML class diagram. If done on paper/pencil or whiteboard, please be sure the diagrams are readable and clear.

Homework/Project 2 is Due at 10 AM on Friday 10/18.

Late submissions for this homework project suffer a 15% penalty in grade, and after a week, the homework may no longer be submitted.

E-mail or Slack me or the class staff regarding homework questions or assistance.

Extra Credit Opportunity – JUnit tests – up to 20 points

Using JUnit, create a set of ten test methods (methods with the @Test decorator) in a single test class called MyUnitTest. These unit tests should be applied appropriately to classes and methods in your simulation code to ensure their proper behavior. You should be able to instantiate a MyUnitTest object before your simulation run and then execute each of the ten test methods at some point before, during, or after the simulation, with output from each test indicating the identification of the test and success or failure of each test method. A single test method may contain single or multiple assert statements as you feel are needed or are appropriate.

I suggest using a simple test class containing test methods using a variety of assert statements as outlined here (http://tutorials.jenkov.com/java-unit-testing/simple-test.html), here (http://tutorials.jenkov.com/java-unit-testing/asserts.html), and here (http://tutorials.jenkov.com/java-unit-testing/matchers.html). I further suggest using a later version of JUnit 4.

If you are more experienced with JUnit, you may want to implement the tests in a more sophisticated fashion. As long as ten test methods are created and clearly used, you may use an alternate implementation.

You will be awarded 2 extra credit points for each functioning and relevant test case demonstrated as being executed in the simulation run out file, to a maximum of 20 points.