Program title and names of team members: **Marissa Bueno**
Language and environment used for development: **Java, JDK 11, Intellj**
Text description of program design, including any assumptions you have made about specifications that may not be complete:

The program begins by setting up the simulation by creating all 24 tools using the factory method. The program then creates 12 customers, each of different types. The program then enters the simulation. At the beginning of each day, customers who have a rental with 0 days left on the record, return the tools back to the store and the inventory is updated. If the record still has more days on their rental, the day tracker is reduced by one. Over each of the 35 days, a random number (n) is selected to determine how many customers are to visit the store that day. A random sample of size n is then selected. If the customer already has three active rentals, they will be unable to visit the store. If this isn't the case, a check is done to ensure that there are tools in stock. If there are no tools, then no rental transaction is made. If there are tools, then a number of tools and number of days are randomly selected based on the type of customer is visiting the store. After these numbers are obtained, a random selection of available tools are selected from the hardware store and rented to the customer. A random number of extensions is generated for each rental and applied on top of the rental transaction which is facilitated by the decorator pattern. After the transaction setup has completed, an update() method is triggered which sets off updates for the customer object and the hardwarestore object. This is done by using a facade design pattern; it reduces complexity by lowering the number of methods needed to be called and the number of classes that need to be dealt with. At the end of the day, an inventory of the store is displayed as well as how much money was made for that specific day.

Each day follows the same pattern described above. The design pattern of Iterator is sprinkled in the program to help facilitate the iteration of customers dealt with each day. At the end of the simulation, a customer history record is displayed to show how many of each customer type visited over the month as well as how much money the store made over the time period. Before the simulation begins, a total of 10 JUnit tests are ran and their results are displayed.

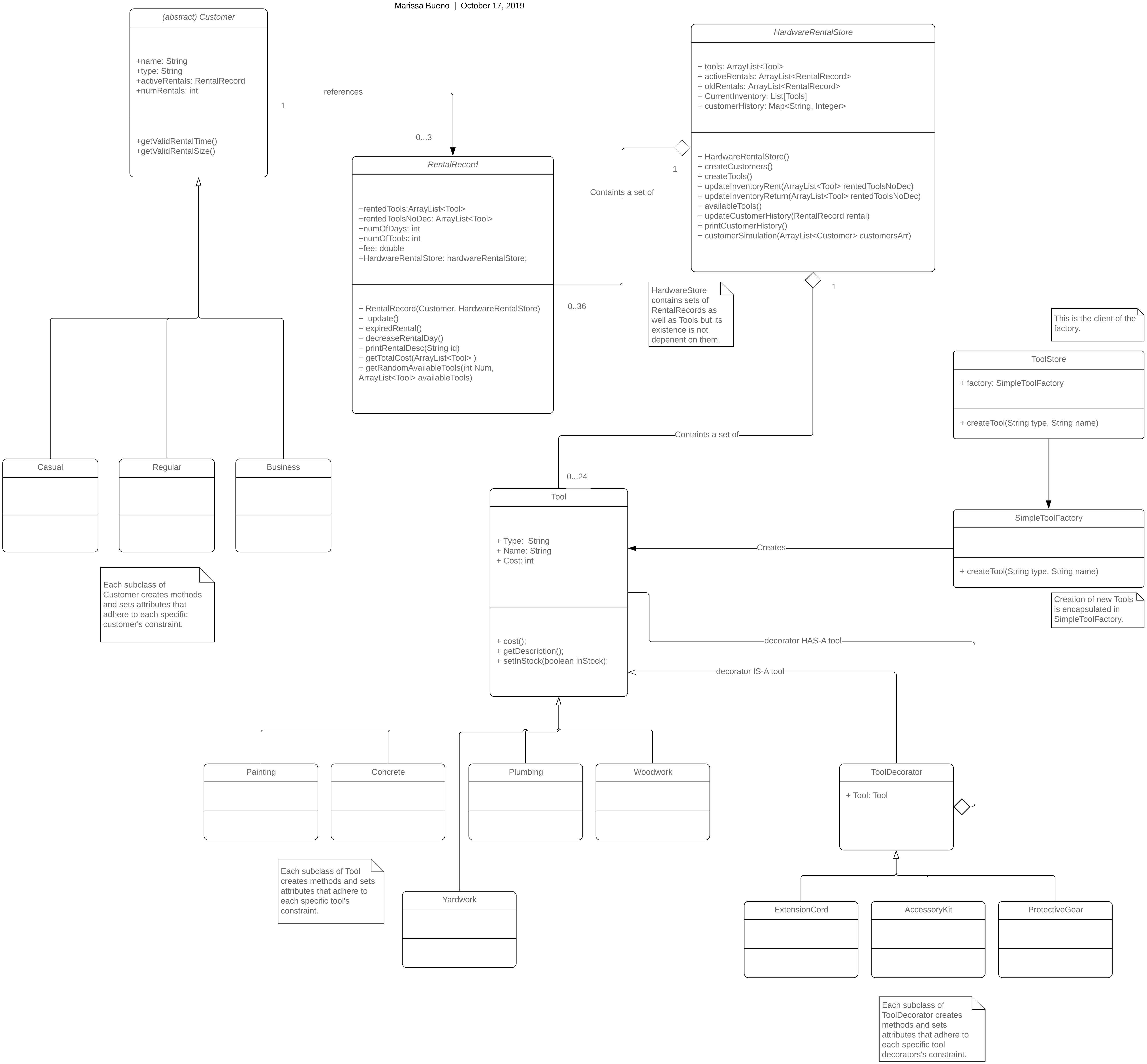
This is the general idea of how the program operates.

A full UML Class diagram that shows classes and relationships from your design (include data attributes and methods in the class diagram

See attached.

Hardware Rental Store

Marissa Bueno | October 17, 2019


(abstract) Customer
+name: String
+type: String
+activeRentals: RentalRecord
+numRentals: int
+getValidRentalTime()
+getValidRentalSize()
references
1
0...3
RentalRecord
+rentedTools:ArrayList<Tool>
+rentedToolsNoDec: ArrayList<Tool>
+numOfDays: int
+numOfTools: int
+fee: double
+HardwareRentalStore: hardwareRentalStore;
+ RentalRecord(Customer, HardwareRentalStore)
+ update()
+ expiredRental()
+ decreaseRentalDay()
+ printRentalDesc(String id)
+ getTotalCost(ArrayList<Tool>)
+ getRandomAvailableTools(int Num, ArrayList<Tool> availableTools)
Containts a set of
1
0..36
HardwareRentalStore
+ tools: ArrayList<Tool>
+ activeRentals: ArrayList<RentalRecord>
+ oldRentals: ArrayList<RentalRecord>
+ CurrentInventory: List[Tools]
+ customerHistory: Map<String, Integer>
+ HardwareRentalStore()
+ createCustomers()
+ createTools()
+ updateInventoryRent(ArrayList<Tool> rentedToolsNoDec)
+ updateInventoryReturn(ArrayList<Tool> rentedToolsNoDec)
+ availableTools()
+ updateCustomerHistory(RentalRecord rental)
+ printCustomerHistory()
+ customerSimulation(ArrayList<Customer> customersArr)
1
Containts a set of
0...24
Casual
Regular
Business
ToolStore
+ factory: SimpleToolFactory
+ createTool(String type, String name)
SimpleToolFactory
+ createTool(String type, String name)
Creates
Tool
+ Type: String
+ Name: String
+ Cost: int
+ cost();
+ getDescription();
+ setInStock(boolean inStock);
decorator HAS-A tool
decorator IS-A tool
Painting
Concrete
Plumbing
Woodwork
Yardwork
ToolDecorator
+ Tool: Tool
ExtensionCord
AccessoryKit
ProtectiveGear

HardwareStore contains sets of RentalRecords as well as Tools but its existence is not depenent on them.

This is the client of the factory.

Creation of new Tools is encapsulated in SimpleToolFactory.

Each subclass of Customer creates methods and sets attributes that adhere to each specific customer's constraint.

Each subclass of Tool creates methods and sets attributes that adhere to each specific tool's constraint.

Each subclass of ToolDecorator creates methods and sets attributes that adhere to each specific tool decorators's constraint.