Project 3

Team Member:

- Yutaka Urakami
- Hao Wu
- Mengyu Wu

Used language: Java 8

Used Test Framework: JUnit 5

Design patterns

- Simple Factory

    When initializing concrete customers (BusinessCustomer, CasualCustomer, and RegularCustomer) and tools (Concrete, Painting, Woodwork, Plumbing, and Yardwork), CustomerFactory and ToolFactory are used, respectively.

- Decorator

    We used decorator pattern for adding option(s) to concrete tools. Component is Tool class, concrete components are Concrete, Painting, Woodwork, Plumbing, and Yardwok classes, Decorator is OptionalTool class, and concrete decorators are AccessoryKit, ExtensionCord, and GearPackage classes.

- Observer

    Subject is Subject class, concrete subject is HardwareRentalStore class, observer is Observer class, and concrete observers are BusinessCustomer, CasualCustomer, and RegularCustomer classes. Every time the day is changed, subject sends a notification to concrete observers with random #observers and random order. The subject stops a notification when

    - there are no available rentals in the store
    - subject finishes notification

Renting operation

    When an observer receives the notification, the observer (customer) checks #available tools in the store. And, if there are enough tools, the customer rents a tool with or without option(s) and initializes it [rentTool(Subject subject, int nights) method in rentToolCustomer.jave]. The number of rents is determined in an update(Subject subject) method in each concrete customer class.

Returning operation

Every time the day is changed, HardwareRentalStore decreases the number in remainingDays attribute in the class. And, check the value equals to 0. If the value is 0, remove it from active rentals (activeRentals attribute in the class).

Constant values

We put all constant values in Constants.java. The values:
- Tool type name (CONCRETE, PAINTING, and so on) which are used for initializing and ToolFactory
- Both lists use the same index (About the index, please look at HarwareRentalStore/active and left tools)
    - List of concrete tool names (TOOL_NAME)
    - List of concrete tool's type names (TOOL_TYPE)

- Customer type name (BUSINESS, CASUAL, and so on) which are used for initializing and CustomerFactory
- Both lists use the same index (About the index, please look at HarwareRentalStore/active and left tools)
    - List of concrete customer names (CUSTOMER_NAME)
    - List of concrete customer's type names (CUSTOMER_TYPE)

HardwareRentalStore (Concrete Subject)

Manage:
- active and left tools
    Using availability (boolean list) in HardwareRentalStore.java. The size is #all tools. True means left, and False means active rentals. Each index corresponds to TOOL_NAME, TOOL_TYPE, CUSTOMER_NAME, and CUSTOMER_TYPE. activeRentals is an array of active rentals
- day
    Change the day, and finish when the day is 35.
- output
    reportSales(), reportActiveAndLeftRentals(), and reportFinal()
- returning operation (Please look at Returning operation)

UML

UML

Concrete
+ getCategory(): String
+ setDays(int): void

Painting
+ getCategory(): String
+ setDays(int): void

Woodwork
+ getCategory(): String
+ setDays(int): void

Plumbing
+ getCategory(): String
+ setDays(int): void

Yardwork
+ getCategory(): String
+ setDays(int): void

ToolFactory
+ getTool(String, String): Tool

Tool
productName: String
price: double
day: int
rentingPersonName: String

+ getCategory(): String
+ setDays(int): void
+ getName(): String
+ getProductName(): String
+ getPrice(): double
+ getDay(): int
+ getRentingPersonName(): String
+ setRentingPersonName(String): void
+ toString(): String

ArrayList<>

OptionalTool

AccessoryKit
tool: Tool
+ getPrice(): double
+ getName(): String
+ getCategory(): String
+ setDays(int): void
+ getDay(): int
+ getProductName(): String
+ getRentingPersonName(): String

ExtensionCord
tool: Tool
+ getPrice(): double
+ getName(): String
+ getCategory(): String
+ setDays(int): void
+ getDay(): int
+ getProductName(): String
+ getRentingPersonName(): String

GearPackage
tool: Tool
+ getPrice(): double
+ getName(): String
+ getCategory(): String
+ setDays(int): void
+ getDay(): int
+ getProductName(): String
+ getRentingPersonName(): String

Subject
isNotification: boolean
- observers: ArrayList<Observer>

+ registerObserver(Observer): void
+ removeObserver(Observer): void
+ notifyObservers(): void
+ getAvailability(): boolean[]
+ setAvailability(int, Boolean): void
+ getNumAvailableTools(): int
+ announce(String): void
+ addRentedTool(Tool): void
+ stopStore(): void

<<interface>>
Observer
+ update(Subject): void

CustomerFactory
+ getCustomer(String, String): Observer

BussinesCustomer
+ update(Subject): void

RegularCustomer
+ update(Subject): void

CasualCustomer
+ update(Subject): void

HardwareeReentalStore
- availability: boolean[]
- day: int
- otalSales: double
- salesFormat: DecimalFormat
- todayRentals: ArrayList<Tool>
- activeRentals: ArrayList<Tool>
- completedRentals: ArrayList<Tool>
- remainingDays: ArrayList<Integer>

+ openStore(): void
+ getAvailability(): boolean[]
+ setAvailability(int, Boolean): void
+ getNumAvailableTools(): int
+ announce(String): void
+ addRentedTool(Tool): void
+ stopStore(): void
- reportSales(): void
- reportActiveAndLeftRentals(): void
- prepareForNextDay(): void
- prepareForOpening(): void
- reportFinal():: void

Customer
- name: String
+ getName(): String
+ rentTool(): void
- generateRentedTool(): Tool
- randomlyAddOption(): Tool

Simulator
+ main(String[]): void

ArrayList<>

1

component

0..6

observers

subject