CS 2336 PROJECT 1 – Disneyland Dining Rewards

Pseudocode Due: 9/10 by 10:00 PM

Project Due: 9/25 by 10:00 PM

KEY ITEMS: Key items are marked in red. Failure to include or complete key items will incur additional deductions as noted beside the item.

Submission and Grading:

- **The file containing `main` must be named Main.java. (-5 points)**
- **The project files must be in a package named `DrinkRewards`. (-5 points)**
- All project deliverables are to be submitted in eLearning.
- Zip the contents of the `src` directory into a single zipped file
  - Make sure the zipped file has a .zip extension (not .tar, .rar, .7z, etc.) (-5 points)
  - Please review the submission testing information in eLearning on the Course Homepage
- Projects submitted after the due date are subject to the late penalties described in the syllabus.
- Programs must compile and run with JDK 8.
- Each submitted program will be graded with the rubric provided in eLearning as well as a set of test cases. These test cases will be posted in eLearning after the due date. Each student is responsible for developing sample test cases to ensure the program works as expected.
- **Type your name and netID in the comments at the top of all files submitted.** (-5 points)

Objectives:

- Use inheritance to create base and child classes.
- Utilize multiple classes in the same program.
- Perform standard input validation
- Implement a solution that uses polymorphism

Problem: Disneyland has begun a service that allows guests at the park to pre-order drinks and pick them up at a designated time so as to avoid long concession lines. Guests that spend more than $50 (cumulatively) become preferred customers and are awarded discounts on their orders. A program is needed to track the amount spent by each customer and promote them to preferred customer when they have accumulated $50 in purchases.

Pseudocode: Your pseudocode should describe the following items

- Main.java
  - List functions you plan to create
    - Determine the parameters
    - Determine the return type
    - Detail the step-by-step logic that the function will perform
  - Detail the step-by-step logic of the main function

Classes:

- **Customer (base class)**
  - Members
    - First name (string)
    - Last name (string)
    - Guest ID (string)
    - Amount spent (float)
  - Methods
    - Overloaded Constructor
    - Accessors
    - Mutators
- **Preferred Customer - Gold (derived from customer)**
  - Member
    - Discount Percentage (float)
  - Methods
    - Overloaded Constructor (chained to base class constructor)
    - Accessor
    - Mutator
- **Preferred Customer - Platinum (derived from customer)**
  - Member
    - Bonus bucks (float)
  - Methods
    - Overloaded Constructor (chained to base class constructor)
    - Accessor
    - Mutator

Details:

- The price of the drink is determined by the type of drink and the amount of ounces ordered
  - Soda - $0.20
  - Tea - $0.12
  - Fruit punch - $0.15
- Each drink will come in a cylindrical container
  - Small
    - Diameter – 4 in.
    - Height – 4.5 in.
    - 12 oz.
  - Medium
    - Diameter – 4.5 in.
    - Height – 5.75 in.
    - 20 oz.
  - Large
    - Diameter – 5.5 in.
    - Height – 7 in.

        - 32 oz.
- The container can be personalized with different Disney graphics
    - The total price of the personalization is determined by the price per square inch
    - The area covered by the graphic will be just the outer surface area of the cylinder
    - The surface area of the cylinder will not include the top and bottom of the cylinder
- Preferred customer information and regular customer information will be stored in files and will be read into memory before the program processes orders.
    - The preferred customer file may be empty or may not exist.
    - In both cases, the respective customer array should not be created until a customer reaches preferred status.
- Regular customers and preferred customers will be held in 2 separate arrays. (-10 points if not)
    - Both gold and platinum customers will be stored in the same array (preferred)
    - Do not use an array list for this project
- The arrays will never be larger than the actual number of customers in each category.
- Orders will be read from a file.
- After processing an order, if a regular customer has spent at least $50, promote the customer to preferred status (gold).
    - The preferred customer array will be resized to add one more element.
    - Add the new preferred customer into the open element at the end of the preferred customer array.
    - Resize the regular customer array and remove the customer data that was promoted to preferred status.
    - Do not rearrange the order of the data in either array.
    - Remember that each array should be no bigger than the number of customers at that level.
- Gold preferred customers get a discount based on how much they have spent overall
    - $50 = 5% discount
    - $100 = 10% discount
    - $150 = 15% discount
    - This discount applies as soon as the above target value is obtained.
        - Example 1: customer has spent $40 previously. The customer places an order for drinks that amounts to $10. The customer is immediately placed in preferred status and given a 5% discount. The new amount spent is $49.50
        - Example 2: customer has spent $90 previously. The customer places an order for drinks that amounts to $10. However, the customer has a 5% discount as a preferred customer. The customer has a total of $99.50 after the discount is applied and is not moved to the next bracket for preferred customers
            - If the customer placed an order for $11, the new total with the discount would be $100.45. The customer is immediately placed into the new bracket and a 10% discount is given instead. The new amount would be $99.90 and the customer would be at the 10% discount level.
- After processing an order, if a gold preferred customer has spent at least $200, promote the customer to platinum status.
    - Replace the gold member object in the array with a platinum member object
    - Update the bonus bucks if necessary (see below)

- Platinum preferred customers receive bonus bucks instead of a discount.
    - For every $5 over $200, the customer receives 1 bonus buck (worth a dollar)
        - When an order is placed, bonus bucks are only earned for every multiple of $5 above the beginning total
            - For example, if a customer has spent $217, and makes a purchase of $12, the customer would earn 2 bonus bucks (one for $220 and one for $225)
    - The bonus buck is applied to the next order
    - The total spent does not include any bonus bucks used
        - If an order costs $5 and a bonus buck is used, the total sales increases by $4.
    - Platinum status is applied when the total amount spent is $200 after gold member discounts are applied.
    - If the gold discount puts the customer under the $200 threshold, the customer remains at gold status.

Input: Input data will be stored in three files: preferred.dat, customers.dat and orders.dat. The total number of lines in each file will be unknown.

Preferred.dat will hold the data for preferred customers and customers.dat will hold the data for regular customers. Both files should be read at the beginning of the program to establish the preferred customer and regular customer arrays respectively. Preferred.dat may be empty or may not exist. In such cases, there should not be a preferred customer array created before reading the orders.

Each line of preferred.dat will be formatted as follows (except for the last line of the file which may not have a newline character). The data will be listed in the following order on each line with a space between each field. There will be no invalid data on any line.

- `customer ID`
- `first name`
- `last name`
- `amount spent`
- `discount or bonus bucks`
- `newline`

Amount spent will indicate if the customer is gold or platinum. Discount will be a floating-point number for gold members or an integer value for platinum members.

Each line of customer.dat will be formatted as follows (except for the last line of the file which may not have a newline character). The data will be listed in the following order on each line with a space between each field. There will be no invalid data on any line.

- `customer ID`
- `first name`
- `last name`
- `amount spent`
- `newline`

Orders.dat will hold all orders to process. Each valid line of the file will be formatted as follows (except for the last line of the file which may not have a newline character). A valid line will consist of data listed in the following order with a space between each field. **There may be invalid data on any line. If a line contains invalid data, ignore everything on the line.**

- `customer ID (string)`
- `size (character) – S, M, or L`
- `drink type (string) – soda, tea or punch`
- `square inch price (float)`
- `quantity (integer)`
- `newline`

Invalid data: Invalid data can consist of any of the following:

- incorrect number of fields (more or less than 5)
- incorrect ID – doesn't match an existing ID of a customer
- incorrect value option for size or drink type
- garbage characters – character present in value that is outside the data type's acceptable character set
    - example: a letter character in a value for a field with a numeric data type

Output: At the end of the program, write the regular customer data and preferred customer data to their respective files. Use the proper format as listed above in the Input section. There will be no output to the screen.