Java Midterm Exam

This exam is due Wednesday, October 19th @ 11:59pm. You may take as long as you need on the exam up until the deadline.

YOU CANNOT:

- Work with someone else on this exam.
- Copy from someone else's exam.
- Use StackOverflow or any search engine, such as Google.com, for any reason (other than to go to the course exam or approved resources).
- Use an internet search for keywords in the exam. For example, do not Google "loading a .csv file in Java".
- Discuss the exam with other students until cleared by the instructor to do so. Other students have not yet taken the exam, and you CANNOT provide academically dishonest assistance to them.

YOU CAN:

- Reference any material from the course or recitation that can be accessed in Canvas. This includes videos, readings, slides, code samples, homework assignments, quizzes, and practice coding exercises.
- Reference approved online Java documentation such as:
    - https://docs.oracle.com/en/java/
    - https://www.w3resource.com/java-tutorial/
    - https://www.baeldung.com/

Use of Ed Discussion/Office Hours as it Relates to the Exam

If you have any clarification questions regarding the exam, please make a **private post** on Ed Discussion. We will keep a pinned thread going of commonly asked questions, so please be sure to check that before posting your question

The Assignment

This Java exam will involve implementing a *Vending Machine program*. You will be provided with the skeleton of the program and you will have to implement the methods and write the unit tests. The design of the program has been set up for you.

Steps for Completing the Exam

- Complete all of the required methods
    - Implement all of the methods defined in the provided classes
    - Javadocs have already been provided
    - Add comments to your code
    - You can create any number of helper methods (with Javadocs)
    - The *main* method has already been implemented for you. DO NOT CHANGE THE CODE IN MAIN.
- Test your code by running (and passing) all of the provided test cases in the given test files. **Write additional test cases as noted** and make sure they pass as expected. Your test cases should be distinct.
- Make sure your entire program and the unit testing file run without errors!
- To complete the assignment, submit all the classes in your entire Java project. This will include everything in your "src" folder.

Required Classes & Methods

You have been provided four classes that implement the functionality for a Vending Machine from which a user can enter a product number and some payment and have the machine dispense the requested product.

The classes are:

Item.java:

- A simple class associating a price, an item ID, and an item name together to represent an Item. The only things here are fields, getters, and a constructor. This class is provided for you in its entirety.
- An Item has three fields, one for an Item's price, one for its unique id, and one for its name
- Item.java has a single simple constructor and three basic "getters" for its fields

Payment.java:

- A Payment represents a sum of money. Its fields track how many dollars ($1.00), quarters ($0.25), and dimes ($0.10) the Payment consists of.
- Payment.java has two methods:
    - One for calculating the value of the dollars and coins in the Payment
    - One for returning a new Payment with enough dollars, quarters, and dimes to represent the difference in value between the current Payment and some other amount of money

Transaction.java

- Represents the way that money and items are returned from the Vending Machine to the user. This class is provided for you in its entirety.

- Note that a Transaction will always have some Payment and some Item associated with it, as well as a boolean flag that indicates whether or not the Transaction resulted in a successful purchase.
- A successful Transaction represents a completed purchase, and should contain the change that the user receives from their purchase as a Payment, the Item they bought, and have the isSuccessful flag set to true.
- An unsuccessful Transaction represents a purchase attempt where the user did not provide enough money in their Payment. The Item in the Transaction should still be the Item that the user attempted to purchase, and the isSuccessful flag should be set to false.

VendingMachine.java

- Represents the Vending Machine itself
- A VendingMachine has one field: an array of Items that represents their inventory. VendingMachine.java has five crucial methods
    - *addItem* adds the given item to the vending machine inventory
    - *removeItem* removes the item with the given ID from the vending machine inventory
    - *selectItemByNumber* searches through the inventory, checking to see which Item matches the provided ID and returns that item
    - *selectItemByName* searches through the inventory, checking to see which Item matches the provided name and returns that item
    - *attemptPurchase* performs the function of having a customer enter an Item id, provide some money as a Payment, and return a Transaction representing what happens next. If the customer gave enough money, the Transaction should contain the customer's change, their requested item, and be marked as successful. Otherise, it should contain the insufficient payment the customer provided, the requested item, and be marked as not successful.

Fill in the specified methods. Your code must compile, and the methods that you fill in should conform to the specification outlined above and repeated in the method headers in the files you receive. Do not change any fields or method signatures whatsoever.

METHODS TO FILL IN:

Payment.java

- *public double calculateValue()*
- *public Payment makeChangeForPurchase(double itemCost) throws Exception*

VendingMachine.java

- *public boolean addItem(Item item)*

- *public boolean removeItem(int itemID)*
- *public Item selectItemByNumber(int itemID)*
- *public Item selectItemByName(String itemName)*
- *public Transaction attemptPurchase(int itemID, Payment currencyProvided)*

Testing

In addition, you will notice two test classes - PaymentTest.java and VendingMachineTest.java. We have provided unit tests for you to run your code against. Ensure your code passes all provided test cases. **In addition you must add your own test cases to each test method.** The comments in the provided file specify whether you should add 1 or 2 test cases to the specific test method.

METHODS TO ADD TESTS TO:

PaymentTest.java
- *void testCalculateValue()*
- *void testMakeChangeForPurchase()*

VendingMachineTest.java
- *void testAddItem()*
- *void testRemoveItem()*
- *void testSelectItemByNumber()*
- *void testSelectItemByName()*
- *void testAttemptPurchase()*

Submission

Please submit all the classes provided in the starter zip file (Item.java, Payment.java, PaymentTest.java, Transaction.Java, VendingMachine.java, VendingMachineTest.java). We will accept a zip submission or an upload of each of the individual Java files on Canvas. Again, take care that you have not changed any function signatures or changed anything about the other classes that were provided to you.

Grading

You will be graded on the following criteria:

Part 1
- Did you implement the methods correctly? **14 pts (2 pts per method)**

- Did you write and pass your own test cases? **11pts (1 pt per required test case)**
- Did you adhere to good style (camelcase variables and commenting non-trivial lines of code)? **2pts**