Final Exam

Collaboration – not permitted

For this test, you are NOT permitted to work with or communicate with others in any form. This includes sharing your code or helping classmates troubleshoot code. Violation of this will result in an automatic grade of 0 and an academic misconduct being filed.

GitHub

At the start of the test, you must create a private GitHub repository and make **ProfBlanc** a collaborator. You need to submit your code to a private GitHub repo upon completion of every question.

Once you have submitted the GitHub link in Blackboard you cannot make any changes to your GitHub repo. The test is considered submitted at that time of submission to Blackboard. If the timestamp on GitHub is later than the timestamp in Blackboard, your paper will not be marked. Tests with a GitHub timestamp after due date will not be accepted.

Coding Best Practices

This section is designed to ensure students adhere to coding best practices. Marks will be removed for code that matches any of the items below:

- 1 mark will be removed for each line of code I need to change to make the code compile
- 1 mark will be removed for each duplicated method.
- 1 mark will be removed for each redundant file or directory in the project. For example, if the GitHub repo has multiple "main" methods to start the program and the entire program is in more than 1 directory. ONLY submit the code that addresses this test.

Files for the test

You are given a "skeleton" program in a zip file that contains all the scenes and java files for your program. Extract the files to your local computer and open it with IntelliJ.

The objects in the view are already connected to the controller class methods. You need to supply the code in the methods to create the desired behavior.

JSON file

You are given a file called customers.json. It is your job to look at the structure of the json file and update the ApiResponse, Customer and Product classes to match the structure of the json file. The classes do not require constructors, set methods or Javadoc.

1. The Customer class (3 marks)

For the customer class, your java program must track (at minimum) the id, first name, last name, phone and purchased products. (2 marks for selecting the correct variable names/datatypes, 1 mark for incorporating a variable that can hold the product purchases)

When complete, ***commit and push your code to GitHub with the message question 1 complete.***

2. The Product class (3 marks)

This class should track the sku, name, sale price, regular price and a String to hold the url information for the image. This class should also have a toString() method.

The toString() method should format the response as "[name]-$[sale price]".

For example "T-shirt with Logo-$17.00".

(3 marks for selecting the correct variable names/datatypes and creating a toString() method)

When complete, make a commit and push your code to GitHub with the message "Question 2 complete".

3. Reading the json file (3 marks)

Create a class/method that will parse the JSON file and allow you to access an array list of Customer objects.

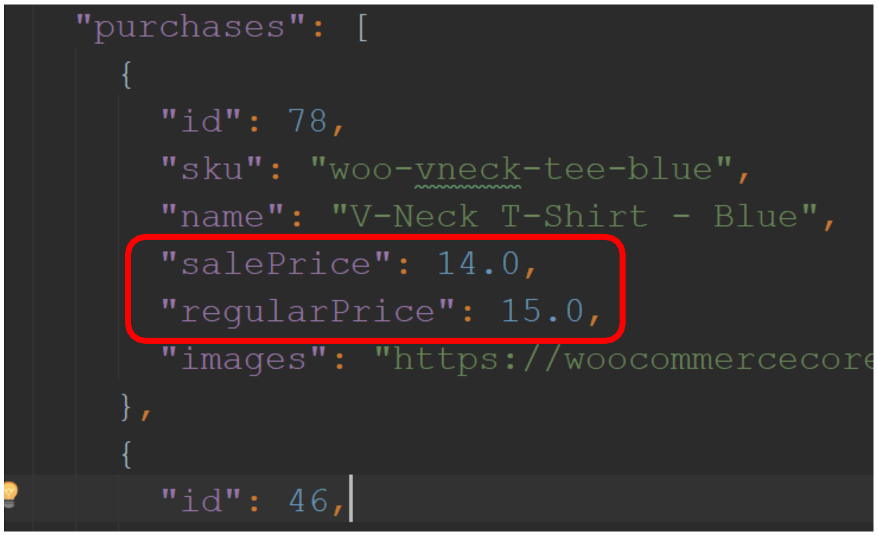
(1 mark for properly opening closing objects to read the json file, 1 mark for calling the gson methods correctly, 1 mark for being able to access an ArrayList of Customer objects

Commit and push your code to GitHub with the message "Question 3 complete"

4. Customer class – special methods (6 marks)

4.1 Create a method that returns the total purchases as a double. Each customer will have 0 or more products they have purchased. This method should use a stream to add up all the purchases and return the sum of their "salePrice" (2 marks). If you cannot get the stream working, but can get the correct total (1 mark). ***Now commit and push your code to GitHub with the message question 4.1 complete***.

4.2 Create a method that returns the total amount the customer has saved as a double. Each product has a regular price and a sale price. See the area highlighted in red in the diagram on the next page.



```
"purchases": [
  {
    "id": 78,
    "sku": "woo-vneck-tee-blue",
    "name": "V-Neck T-Shirt - Blue",
    "salePrice": 14.0,
    "regularPrice": 15.0,
    "images": "https://woocommercecore
  },
  {
    "id": 46,
```

In this example, the customer saved $1 on the first item that he/she purchased. Add up the savings across all the purchases and return a double that is the sum of all the savings. ***Commit and push your code to GitHub with the message "Question 4.2 complete"*** (2 marks)

4.3 Create a method that returns true if the customer saved $5 or more on all their purchases. (1 mark) ***Commit and push your code to GitHub with the message "Question 4.3 complete".***

5. Add your name and student number as comments at the top of all java files (2 marks)

Commit and push your code to GitHub with the message "Question 5 complete".

6. Customers to the TableView (4 marks)

The TableView object should be updated such that that it will display each customers' id, first name, last name, phone number and a column that shows a sum of their total purchases. Their total purchases should be the sum of all sale prices in their purchases.

When setting up the PropertyValueFactory for each column, you need a corresponding "get" method in the customer class. If the propertyValueFactory refers to firstName, you will need a method in the Customer class called "getFirstName()".

The TableView should be populated with the 1000 customers from the json file. (1 mark for configuring each of the columns, 1 mark for loading 1000 customers from the json file, 1 mark for displaying the total $ purchases per customer, 1 mark for displaying the total purchases as a String with a $ and exactly 2 decimal places).

Commit and push your code to GitHub with the message "Question 6 complete".

7. Update the Label with total number of rows (1 mark)

There is a label called "rowsInTableLabel", update it to show the total number of rows in the tableview object (1 mark for the label being set dynamically – in other words, do not simply label.setText("Rows returned: 1000"); it should use the tableview to calculate the # of rows returned. ***Commit and push your code to GitHub with the message "Question 7 complete".***

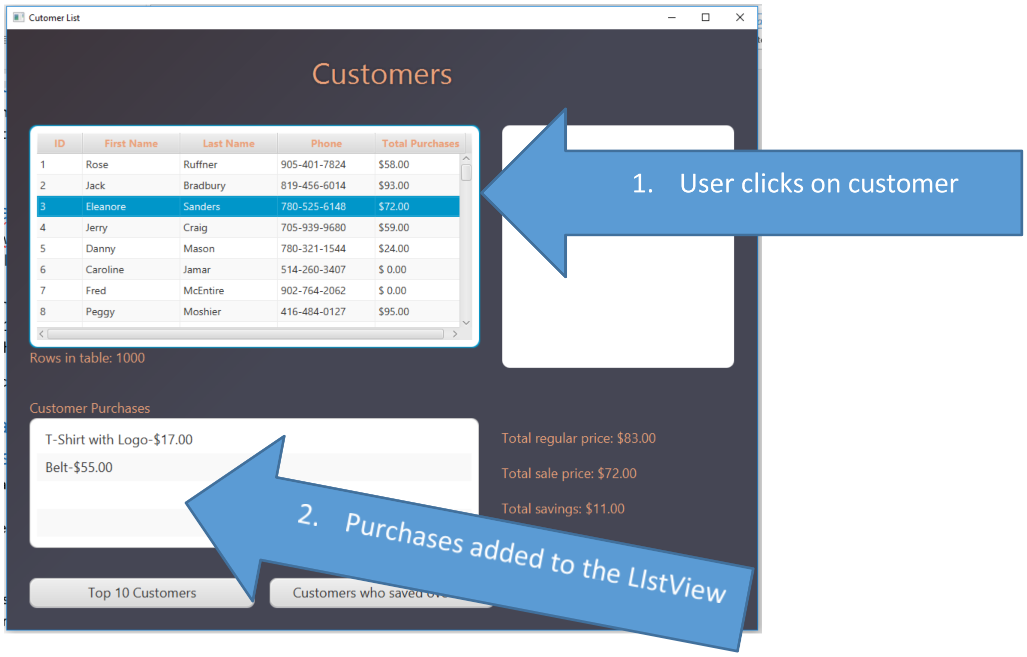
8. Listener added to the TableView (2 marks)

Add a listener added to the tableview object such that if a customer is selected (left click mouse on a customer), it will put all of their purchases in the purchaseListView. (1 mark for the listener, 1 mark for correctly displaying the purchases in the listview). The products should be displayed as "[name]-$[price]" If you did not get the purchases to import, at least create a fake product and add it to list view.

Remember you can add listeners to almost any Javafx object that holds a list by using

getSelectionModel().selectedItemProperty().addListener(<< enter your change listener code>>)

Commit and push your code to GitHub saying question 8 complete.



Cutomer List
Customers
1. User clicks on customer
2. Purchases added to the LIstView
Top 10 Customers
Customers who saved ov

ID	First Name	Last Name	Phone	Total Purchases
1	Rose	Ruffner	905-401-7824	$58.00
2	Jack	Bradbury	819-456-6014	$93.00
3	Eleanore	Sanders	780-525-6148	$72.00
4	Jerry	Craig	705-939-9680	$59.00
5	Danny	Mason	780-321-1544	$24.00
6	Caroline	Jamar	514-260-3407	$ 0.00
7	Fred	McEntire	902-764-2062	$ 0.00
8	Peggy	Moshier	416-484-0127	$95.00

Rows in table: 1000

Customer Purchases

T-Shirt with Logo-$17.00

Belt-$55.00

Total regular price: $83.00

Total sale price: $72.00

Total savings: $11.00

9. Update purchases labels (4 marks)

Add labels that display the sum of the regular prices (msrpLabel) for all purchases, the sum of the sale prices (saleLabel) and the total savings for that customer (savingsLabel). The savings should be calculated by subtracting the total sale price from the total regular price. (1 mark for calculating / displaying the correct total regular price, 1 mark for calculating / displaying the correct sale price, 1 mark for calculating the correct total savings, 1 mark for updating when a different customer is selected)

Commit and push your code to GitHub with the message "Question 9 complete".

10. Customers who saved $5.00 or more (4 marks)

In the Customer class, you should have created a method that will return true if the user has saved $5.00 or more. Use this method to filter the TableView to only include customers that have saved at least $5.00.

The tableview should have 516 customers once filtered (1 mark for the logic to create a list of customers that have saved $5.00 or more, 1 mark for updating the tableview with only those customers, 1 mark for updating the rows returned label to dynamically show the rows in the table, 1 mark for not having any exceptions triggered. For example, if a customer is selected and the button is pushed to filter the table, no exceptions should be thrown).

Commit and push your code to GitHub with the message "Question 10 complete"

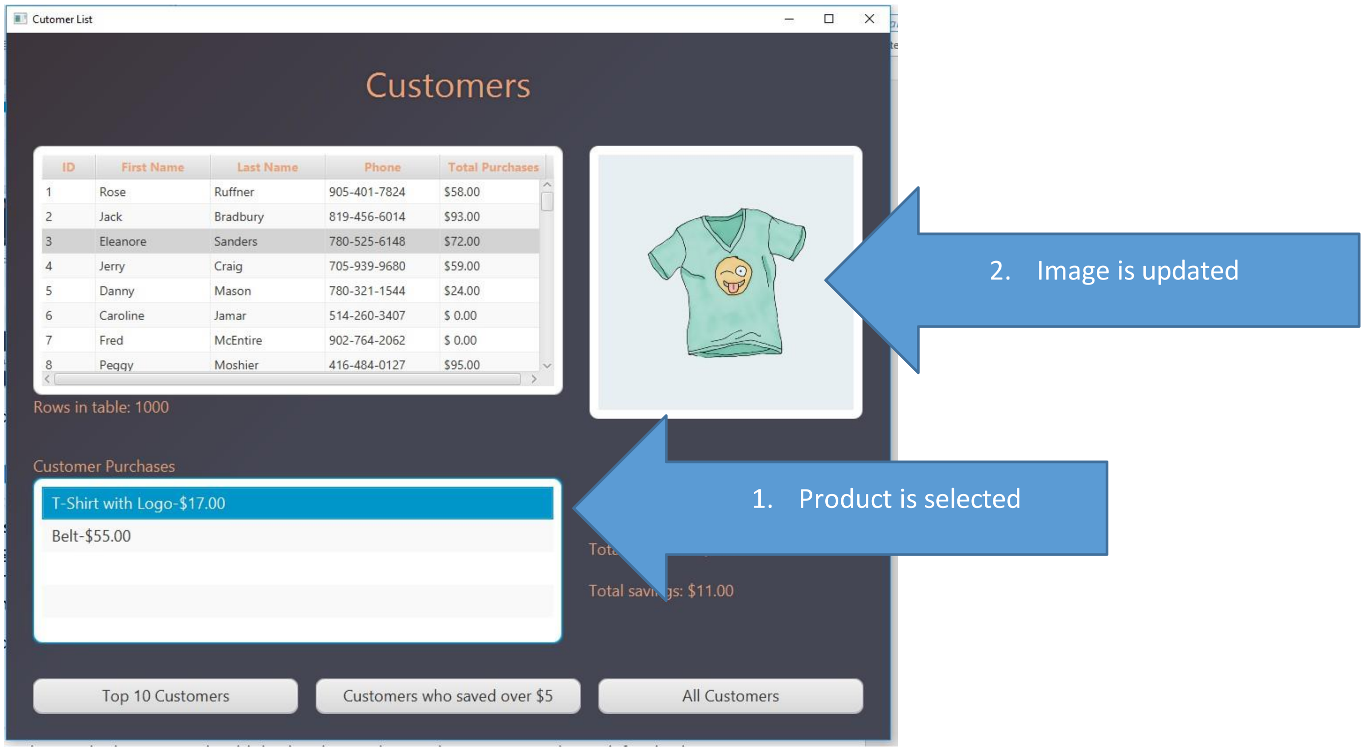
11. Load all customers (3 marks)

When the "All Customers" button is pushed, the tableView object is updated with all the customers from the JSon file (1000). (1 mark for updating the tableview with 1000 customers total, 1 mark for updating the row label, 1 mark for not triggering any exceptions if a customer is selected)

Commit and push your code to GitHub with the message "Question 11 complete"

12 Display image in ImageView (2 marks)

When a user clicks on a customer, it will display all of their purchases in the listview. If the user then clicks on a product in the listview, it should display the product image in the imageview. If a different user is selected, the image view should not be visible. If a different product is selected, the corresponding image view should displayed.



ID	First Name	Last Name	Phone	Total Purchases
1	Rose	Ruffner	905-401-7824	$58.00
2	Jack	Bradbury	819-456-6014	$93.00
3	Eleanore	Sanders	780-525-6148	$72.00
4	Jerry	Craig	705-939-9680	$59.00
5	Danny	Mason	780-321-1544	$24.00
6	Caroline	Jamar	514-260-3407	$ 0.00
7	Fred	McEntire	902-764-2062	$ 0.00
8	Peggy	Moshier	416-484-0127	$95.00

Rows in table: 1000

Customer Purchases

T-Shirt with Logo-$17.00
Belt-$55.00

Tota

Total savi gs: $11.00

Commit and push your code to GitHub with the image: "Question 12 complete".

Submitting your work

As noted above in the GitHub section, your work needs to be captured in a private GitHub repository with **ProfBlanc** as a collaborator.

Once you are confident that you have completed the test with quality, submit the link to your private GitHub repository into Blackboard.

You MUST submit your work by the due date, or your test will not be marked. Manage your time carefully and be ready to submit before the deadline.

All work on this test must be your own.