GEORGE MASON UNIVERSITY | IT 206
Object-Oriented Techniques for IT Problem Solving

Programming Assignment 9/10

Scenario:

Monica from Sew What? Gifts and Knickknacks needs your assistance for one final application. She has received employee complaints the current ordering system does not make it efficient for employees to enter and process orders received from various sources. She wants you to create a new ordering system to track orders for their holiday gift basket. Each order allows a customer to purchase one holiday gift basket. Because there are a limited number of staff members, the ordering system can only handle up to 216 orders. Each order has an order ID number (system generated rather than user provided, with no two orders having the same order ID number), customer information, special handling instructions (e.g. use back door for delivery), and order source which must be Internet, mail, or phone. Customer information includes a mandatory customer name and phone number, and an optional company name. If a company name is provided, it cannot be blank. Internet orders must store a password to access the order details online, the email address used to place the order, a list of up to 17 status updates about the order, and an optional numeric code between 0 and 47, inclusive, which denotes a discount to be applied to Internet orders. The provided password must contain at least 6 characters with one upper case letter and 1 lower case letter. Each status update contains a name of the customer service representative providing the update and text of the status update which can be anything relevant to the order (e.g. order waiting to be processed). Orders by mail do not store anything other than what has been identified for all orders. Orders by phone must store a separate callback phone number, which may not necessarily be the same phone number as the customer's phone number, and a best time to call back which must be a value of: morning, afternoon, or evening.

Create an efficient, object-oriented solution, using good design principles and a polished user interface which will allow a user to enter orders, one at a time, until they have indicated they are finished entering orders. Once all orders are entered, print a well-formatted report listing all order information based on order source for each order, a count of each order by source, and a count of all orders.

Other Requirements:

- The only way to exit the application is for the user to indicate they are finished entering orders. The application should not exit for any other reason.
- Your solution must use object-oriented techniques, including appropriate constants, constructors, accessors (including a toString() method), validating mutators, and special purpose methods. No points earned for a procedural solution.

- Your solution must be designed as a modular solution using methods other than main, with each method performing one task.
- Your solution must contain reasonably **appropriate** validation to validate all data entered. Whenever a user enters invalid data, you must re-prompt them. Try to think about what might be considered appropriate as you are designing your solution beyond what has already been explicitly provided. Justify your decisions, if necessary. Some specific validation requirements have been purposefully omitted.
- Your solution must not import any Java library other than JOptionPane.
- Your solution must demonstrate use of arrays.
- Your solution must incorporate at least two different types of Java unchecked exceptions. You may use built-in and/or custom exceptions.
- When thinking about your application design, remember the concepts taught in the course, including, but not limited to: inheritance, polymorphism, association, aggregation, and composition to help decide the most efficient design.
- When thinking about your user interface, try to think about what makes the most sense and what you might want to see as a user using the application.
- You may not use Scanner or System classes for input/output. You must use JOptionPane.
- You may not use System.exit, or any variant that exits the program in the middle of the program. The program should only exit once the algorithm has finished completing.
- Your solution may not use any functions or language constructs not covered during IT 106 or this semester's IT 206 without prior authorization from your instructor, even if you know other functions or language constructs. We want everyone to be on the same "playing field", regardless of previous programming exposure, and get practice with algorithmic design to solve problems (the intent of the course). Using something existing not discussed in class does not give you as much practice as solving the problem yourself. **Doing this may lead to a substantial grade penalty, a grade of zero, or an Honor Code inquiry.** When in doubt, ask!

To Do (Check Blackboard for Due Dates):

Programming Assignment 9: Solution Design

1) List and describe the purpose of each class that will be needed to solve this problem
    a. You must separately identify (list) and describe the purpose of each class. One or two sentences per class should be sufficient.
2) *Data Definition Class(es)* - Create a detailed UML class diagram, listing and explaining all class variables, accessors, mutators, special purpose methods, and constructors associated to each data definition class.
3) *Implementation Class* - Create a table that lists all methods that will be used to create the implementation class. For each method identified, provide the following:
    a. A 1-2 sentence describing the purpose of the method
    b. A list of the names, data types, and brief description (1-2 sentences) for each input variable into the method, if there are any

c. The name and data type of the variable to be returned from the method, or void if nothing will be returned

Note: You do not need to provide any pseudocode. However, you should create a plan for yourself as to how you will perform the logic for each method. If you do not do this, you will have great difficulty in completing the solution implementation.

An example format to use for this table is as follows. You can format this table in any way you'd like, so long as the information is clearly presented. Completing this table correctly will help you build your documentation and code for your solution implementation.

Method:	getMagazine
Purpose:	The method will allow for the creation and population of a Magazine object based on user input of a title, cost, and number in stock
Inputs:	none
Return:	magazine : Magazine – The Magazine object created and populated from user input

Method:	checkout
Purpose:	The method will allow for the purchase of a magazine. It takes into account the sales tax that will be charged as part of the cost calculation
Inputs:	magazine : Magazine – The magazine to be purchased salesTax : double – The sales tax percentage to be charged as part of the checkout
Return:	void

Upload a Word document containing only items above to Blackboard.

Grading Criteria	
Requirement	**Points**
List and describe the class(es) needed to solve the problem	10
Data Definition Class Design – Detailed UML Diagram	40
Implementation Class Design – Table Listing All Methods	50

Programming Assignment 10: Solution Implementation

Write a well-documented, efficient Java program that implements the solution design you identified. Include appropriate documentation as identified in the documentation expectations document.

To Blackboard, **submit ONLY ONE .zip file** containing all of the .java files part of your submission for your solution implementation. Your .zip file should contain only the .java files in your solution. Be careful that you do not submit .class files instead of .java files.

Warning! You must submit **ONLY ONE** .zip file containing **ONLY** your .java files. Failure to follow this instruction precisely will result in a 10 point deduction of the assignment score. **No exceptions!**

Why is this important? The goal is to teach you how to properly package your IT solutions into a "customer-friendly" format while paying attention to "customer" requirements provided to you.

Your program must compile using jGrasp. Any final program that does not compile, for any reason, will receive an automatic zero. Other IDEs often place in additional code that you are unaware of, doing too much of the work for you. **You are strongly discouraged from using IDEs other than jGrasp.**

Grading Criteria	
Requirement	**Points**
Implementation of object-oriented Java program, using efficient practices, such as the use of constants, good variable names, information hiding, no redundant code, etc.	70
Appropriate objective-style documentation	10
Appropriate intermediate comments	20