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

Programming Assignment 3/4

Scenario:

Sew What? Gifts and Knickknacks loved your work to calculate employee paychecks so much they want to re-hire you again. Monica, the Director of Human Resources (HR), needs a way to track all of the positions being worked on by HR. Because the HR staff is small, Monica does not permit more than 17 positions to be tracked at any given time. You have been tasked with creating an efficient, object-oriented application with a polished user interface to help Monica track these positions, all of which are in different statuses. After meeting with Monica, you learn the status of each position must be one of the following:

- Position posted
- Receiving resumes
- Interviewing applicants
- Position filled

For each position, Monica needs to be able to know the position title, department name of the department assigned the position, and the number of direct reports the position will supervise. For positions in a status of Interviewing applicants or filled, Monica wants to be able to track the starting salary that will be offered for the position. She also mentions the application must have a starting salary that will be offered for the position tracked if applicants are being interviewed or the position is filled, so she can make sure the salary is part of the organizational budget.

When the application is launched, Monica expects to see a menu that contains the following options:

1) **Add a Position**
   This option should allow Monica to enter all information about one position. If she has reached the limit of positions, she should see an error message.

2) **Delete a Position**
   This option should allow Monica to see a list of all entered positions and select a position to delete. If no positions have been entered or if an invalid position has been selected for deletion, Monica should see an error message and be required to select a new menu option.

3) **Display All Positions**
   This option should display all information, well-formatted, about every position entered. Only when the position status is Interviewing applicants or filled should the starting salary also be displayed.

4) **Find Highest Starting Salary**
   This option should find the highest starting salary for a position that is interviewing applicants or filled. It will display, well-formatted, only the position title, department name of the department assigned the position, and starting salary. If two or more positions are tied for the highest starting salary, all positions with that starting salary should be displayed.

5) **Exit Application**
   After Monica completes an option from the menu other than "Exit Application", she should be re-prompted to enter another option. The only way to exit the application is for Monica to select this option. It should not exit for any other reason.

Other Requirements:

- Your solution must use object-oriented techniques (No points earned for a procedural solution).
- Your solution must demonstrate use of arrays. You may not use ArrayLists.
- Your solution must be designed as a modular solution using methods other than main, with each method performing one task.
- Your solution must include appropriate constants, constructors, accessors (including a toString() method), validating mutators, and special purpose methods
- 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!

Hints:

- Think about what type of validations might be appropriate for this application. Make sure these are all handled. Don't forget about what you learned about data validation in IT 106 (e.g. if statements, try/catch, etc.). Unless otherwise specified, whenever an invalid value is entered, the user must be informed that an error has occurred and then be re-prompted to enter in a new value.

- Remember the methods summarizing data on multiple objects should be created in the implementation class

To Do (Check Blackboard for Due Dates):

Programming Assignment 3: 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 4: 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