

# Programming Assignment 3/4

**Scenario:**

Every year, the Career Services department of George Mason University holds two career fairs for interested students. This year due to limited staffing, the Career Services department has requested you to build a capacity management program to keep track of the registered vendors. The Career Fair can hold a maximum of 50 vendors. Each vendor has the ability to hold 15 on the spot in-person interviews.

Each vendor is identified by a table identifier (must be 6 characters long with the first two being upper case characters and the last four being only digits), name of the vendor, and vendor type (Sponsor, For Profit, and Nonprofit). Once a user enters the vendor type, the system should determine how much the vendor paid to attend the career fair. The vendor type prices are $1000, $550, and $300 respectively. Additionally, each vendor should track the total number of visitors, the amount of in-person interviews they had performed, and the cost of the table.

Create a program with a menu that allows a user to select one of the following options:

1. *Register Vendor* – This option will allow a user to create a vendor by entering in the table identifier, vendor name, and vendor type. However, if the maximum number of vendors (50) has already been reached, an error message should display instead.
2. *Remove Vendor* – This option will allow a user to a user to remove a vendor from the career fair. To enable this, the user must first see a list of all the vendors available. The user may then select which vendor to delete. However, if the user chooses an invalid vendor, or if there are no vendors to choose from, the user should be provided with an error message.
3. *Visit Vendor* – This option will allow a user to visit a vendor. If the vendor, has in-person slots available, then the user can request an in-person interview. If all the in-person interviews slots (15) are taken, then the system should take note that a person has visited. If the user chooses an invalid vendor, then user should be provided with an error message.
4. *Display Vendors* – This option will allow a user to view all information about all registered vendors, including the number of interviews scheduled, number of visitors, and cost of the table. However, if there are no vendors registered, the user should be provided with an error message.
5. *Exit the Program* – This option will provide a final report showing the current number of vendors entered and total amount of money earned from each vendor based on their vendor type.

Upon completion of the action from the menu item, the user should be prompted to select another menu item. This will continue until the user has selected to exit the program.

**Other Requirements:**

* You may assume a user will NOT enter two vendors with the table id and vendor name.
* Your solution must use object-oriented techniques (No points earned for a procedural solution).
* Your solution must demonstrate use of arrays
* 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, mutators, and special purpose methods (including a toString() method) with exception handling, as necessary in support of the problem.
* Your solution must not import any Java library other than 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:**

* Don’t forget what you learned about working with array insertions and deletions from IT 106.
* Think about what type of validations might be appropriate for this application. Make sure these are all handled.
* 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
   1. 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:
   1. A 1-2 sentence describing the purpose of the method
   2. A list of the names, data types, and brief description (1-2 sentences) for each input variable into the method, if there are any
   3. 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. As you are naming your Java files, make sure they are indicative to the purpose of the file. **Do not include your name in the file name or use generic names, such as Assignment2.java**. 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 |