New **Y**ork **U**niversity
Computer Science Department
Data Structures
Dr. Anasse Bari

Homework One: University Course Registration System using the Object Oriented Programming Paradigm

Classes, Objects, Inheritance, Polymorphism, Interfaces, Abstract Classes, Method Overriding, File I/O, ArrayLists,Serialization and more on Object- Oriented Paradigm

Deadline: *See NYUclasses for the deadline,* ***15% off per day after the deadline (4 days maximum).***

Learning Objectives:

- Learning how to design and engineer a software solution using the Object-Oriented Programming Paradigm (OOP)
- Practicing Abstraction, Encapsulation, Inheritance, Method Overriding, Method overriding and Polymorphism.
- Practicing Sorting of Objects
- Practicing Abstraction (Abstract Data Types, the ArrayList class, Lists..)
- Practicing File/IO in Java
- Practicing Serialization and Binary Files in Java

Read the guidelines bellow carefully to avoid receiving a zero grade on the HW:

- Attach the Java source files and include them into HW's zip file. The file name should be **YourLastName_HW1.zip**
- Make an archive (zip file or compressed file) with all the **java files and the word document as mentioned in the requirement (the .java files NOT the .class files)** and post it on NYU Classes.
- **It is your responsibility to make sure that the files you upload in NYU classes are working and they are all .java and not .class (double check before you submit). We will grade what we received in NYU classes. If the files do not work (or if you uploaded the wrong files you will get a zero.**
- You must comment you code (basic comments explaining the role of a class, a method or variables used in your submission)
- **Compile and run the program before you submit.**
- It is your responsibility to make sure if the Zip files has your actual latest files. You may send the files to yourself by email to double check that is the actual file before you upload on NYU classes.
- **If the graders cannot open the file, you will receive a grade of zero.**
- **If you send the .class files instead of the .java files (source files) you will receive a zero.**
- An act of cheating will be severely addressed with an immediate zero on the homework and a report to the academic advisor and the administration.

- **You will automatically lose 50% of the points for an exercise if the program does not compile and run correctly.**
- **Plagiarized assignments will get a ZERO grade**. You cannot change the variable names of other student's solution and submit it as yours. The program structure of other students must not match yours. Every student must come up with his/her own solution. Any cheating (e.g. copying from internet without citing sources) is a serious violation of the University student code.
- **Homeworks sent by *email* to the instructor or to the graders will NOT be reviewed and will not be accepted.**

Programming Assignment

Consider the situation where you are hired as a software developer by a new university in your hometown. The university administration wants you to design and implement a **C**ourse **R**egistration **S**ystem (CRS).

As we discussed in class, the very first step of software development is **Requirements Gathering and Analysis**.

After several meetings with your client who represent the school's administration that deals with student registration, consider to the minutes mentioned below that define the requirement you need to implement:

- **Req 01:** The school shall store the following information about each course:
  Course name, course id, maximum number of students that can register in the course, current number of registered students, a list of names of the students currently being registered in the given course, course instructor name, course section number, and course location.

- See attached MyUniversityCourses.csv file for your university data.

- **Req 02:** The system shall allow two types of users: **Admin** and **Student**

- **Req 03:** The system shall allow **the Admin** to perform the following tasks: (these are the options that will be in their menu that will be displayed your program when the administrator logs in)

  Courses Management For the Admin:

  1. **Create a new course**
  2. **Delete a course**
  3. **Edit a course** (this will allow the admin to the course name, Course ID, and instructor name)
  4. **Display information for a given course** (by course ID)
  5. **Register a student** (this option will allow the admin to add a student to some students list (arraylist) without assigning him or her to a course check Req 11 for student's information – *Hint: You might need to have an ArrayList of Students where you store students objects*)

  6.Exit

Reports for the Admin: (still under the Admin menu)

1.**View all courses** (for every course the admin should be able to see the list of courses name, course id, and number of students registered and the maximum number of students allowed to be registered)
2.**View all courses that are FULL** (reached the maximum number of students)
3.**Write to a file the list of courses that are Full** ***(you should name the file as CoursesFull.txt) (writing to a normal text file not serialization)***
4.**View the names of the students being registered in a specific course**
5.**View the list of courses that a given student is being registered on** (given a student first name and last name the system shall display all the courses that students is being registered in)
6. **Sort** courses based on the current number of students registered
7. Exit

- **Req 04:** The system shall allow **the student** to perform the following tasks:

Course Management for the students:

1. **View all courses that are available**
2. **View all courses that are not FULL**
3. **Register on a course** (in this case the student must enter the course name, section, and student first name and last name, the name will be added to the appropriate course's list)
4. **Withdraw from a course** (in this case the student will be asked to enter her/his student name and the course name and section, then the name of the student will be taken off from the given course' list)
5. **View all courses that the current student is being registered in**
6. Exit

During your design meeting with your development team you agreed to adopt the following design:

- **Req 05:** Define an *Interface* for admin class that will have the methods' signatures that will be used by the Admin class.
- **Req 06:** Define an *Interface* for a student class that will have the methods' signatures that will be used by the student.
- **Req 07:** Both classes **Admin** and **Student** inherit from a class named **User**.
  A user should have at least the following class member variables: username, password, first name, and last name (You will need to decide on the methods of a User's class that could be inherited or overridden by the student and the admin class)

- **Req 08:** At the beginning of launching the program, you will need to read all the courses information from the comma delimited file ***MyUniversityCourses.csv*** into an **ArrayList of Course Objects**.
  Notice that initially the number of students registered is zero for every course.
  The student list is empty (there are no students registered in the class at the beginning)

- **Req 09:** For simplicity assume that there is one Admin in the program.
  The username and password for the admin are: Admin and Admin001
  The student username and password are: Student and Student001

- **Req 10: You do not need to follow this Req.10, you can come up with your own design, that is just as one possibility.** At the start of the program, the user is asked to check if they are a student or an admin then if the user is admin, he/he will be asked to enter the username and password. Same applies for student. Depending if it is a student or an admin, you will display the appropriate menu.

  At the beginning of the program you will need to read from the the CSV file if that is the first time you will run the program. Then you need to populate (from the CSV) your data structures where you will be storing the course data (you can just use an arraylist). After the program runs once and exists for the first time you will need to save the data permanently using serialization (see Req 12), the second time (and any time after the second time) your program should read the data from a serialized file and store it back to a serialized file.

- **Req 11:** a student class should have at least a username, password, first name and last name.
  You will need to decide on how to keep track on student's courses if needed. You might need to decide on how to store a list of students if needed.

- **Req 12: Serialization**

  Serializing an object allows the programmer to convert the state of that object into a byte stream that can be reverted back into a copy of the object. A Java object is serializable if its class or any of its superclasses implements either the java.io.Serializable interface or its subinterface, java.io.Externalizable. Deserialization is the process of converting the serialized form of an object back into a copy of the object.You will need to use Serialization to store an object permanently (in this assignment's case it could be used to store the Arraylists of students object and the Arraylist of the courses object). Deserialization will be used to read the files where you stored the objects so you can use them again in your program.

  When the program exits, you will need to write the latest copy of the ArrayLists or the object you are using (that contains the arraylists) into the serialized binary file. The moment you launch your program, you will need to read the files to initiate your objects in your program.

  In Chapter One under the class website**, you will find a sample example of serialization and reading from a file using Java.**

  Here more resources on Serialization please go over:

  http://beginnersbook.com/2013/12/how-to-serialize-arraylist-in-java/
  https://docs.oracle.com/javase/tutorial/jndi/objects/serial.html

Non-Functional Requirements:

0. Explain in no more than two pages in a word document, the overall design of your solution. You can use any format you want (UML diagrams, diagrams similar to the UML, or hand written design...). The design will specify the names of the classes and interfaces that are being used in your program, the relationships between these classes and any other important detail in your design. You may want to also include the workflow of your program (e.g. when the user logs in then a specific menu is being displayed then….)

1. In the same word document, you will need to explain **very briefly** how you used these concepts into your program. You will need to provide examples from your code. Please be as brief as possible.

    - Method overloading
    - Method overriding (at least two examples)
    - Abstract Class
    - Inheritance
    - Polymorphism
    - Encapsulation
    - The concept of ADT (Abstract Data Types)

2. In the same word document, explain how to run your program.

It is important to note that you must come up with your own design on how you will fulfill all the requirements mentioned for this homework. Please make you own assumption for the specific details on implementation and the menu option for both the admin and the student, and the way you read from the file and populate your array lists. You can implement whatever makes sense to you as long as it responds the requirement mentioned in this homework.

Here is one possible way to get started (you might come up with a better way, and you do not have to follow the steps bellow):

- Start with the non-functional requirement 0 (see above under non-functional requirement)
- Design the course class
- Design the interfaces for class admin, student, and user and the methods that will operate on the arraylists you may define in these classes (see req 05 and req 06) you will need to think of how you will store and maintain the students list and the courses list. (maybe a separate class or/ both lists in both the student class and the user class)
- Design class admin, class student and class user (see requirements above to see the relationship between these classes) implement the methods that will operate on the lists of courses and students.
- Write the methods to read the CSV and load the data structures (arraylists you define)
- Write the serialization/deserialization routines to make sure the state of the Course Registration System is persistent.