**ITSS / OPRE 3312 – Object-Oriented Programming**

**Instructor – Professor Khan**

**Project: Project 2 – Student Management System – Version 2**

Total Points Possible: 100 pts

**Exercise Overview**

Implement a Java program that creates a ‘Student Management System’

The user is presented with a menu and should be able to add students, delete students, display full list of students in order and lastly find a student by ID.

Version 2 additional requirement: You will add another module called ‘Course Management System’. When the program starts users will see 3 main options 1. Student Management System 2. Course Management System 3. Exit. When users click 1, options from project 2 Version 1 will display. Exit SMS option should take the users back to the new top-level menu. When users click 2, the following new options for Course Management System will appear. Exit CMS from this menu should take the users back to the top-level menu.

**Functional Requirements (how the code will work from the user perspective)**

* Program displays a description of the system
* Possible options for the user to select:
  + Student Management System(SMS)
    - Add a new student
    - Activate or Inactivate a student
    - Display all students in ascending order by first name and also write to a file called “StudentReport.txt”
    - Search for a student
    - Assign on-campus job(Ask for an ID, assign a job)
      * Possible jobs are – TA(Teaching Assistant or Research Assistant)
    - Display students with on-campus job(you will need a method in your Student\_Employee subclass). The subclass method should be able to print Student ID, Name, Employment type and Job
    - Exit SMS
  + Course Management System(CMS)
    - Add a new course
      * You will not have to create an array, just write the course to a file called “Courses.txt”
      * Create the file if it doesn’t exist
      * Check if the course exists in the file and add it if it doesn’t
    - Assign student a new course
      * Ask for a Student ID and Course ID
      * Write to a file as soon as assignment is made. Filename = “CourseAssignment.txt”. If the file doesn’t exist, then create it
    - Display students with assigned courses. You will have to read from the file
    - Exit CMS
  + Exit

**Technical Requirements (how you must code it)**

The system should include the following Java components:

* You must have two interfaces. One will be implemented by Student class and the other will be implemented by Course class. These two interfaces will declare all of your necessary getters, setter and/or displays
* You must have these classes
  1. Driver class or main class
  2. Student class
     + Student\_Employee subclass
  3. Course class
* You must implement an object array of students
  1. The object will have
     + Student ID(Randomly assigned ID between 0 and 99)
     + Student First Name
     + Student Last Name
     + Student level
       - Freshman
       - Sophomore
       - Junior
       - Senior
     + Active
       - true – this is the default
       - false
* Student\_Employee must inherit from Student class. You will need an object array for Student\_Employee
  1. Fields are
     + Employment type- Full time or Part time
     + Job - TA or RA
* Your course class will have
  + - Course ID(Example – 3311, 3312 etc.)
    - Course name(Java 101, History 101 etc.)
* You need to ask the user for the number of students they will have in the system
  + Create the object array based on the number
* Name of your source code main class as follows: YourName\_Section000\_Project2.java
* Variables (other than local vars) should be declared at the beginning of the main method.

**Exception handling requirement:**

* You need to show at least 3 exception handling implementation within your program

**Hints and suggestions.**

* Use comments to label the different sections (and subsections) of your code.
* While writing code, print variable values to console frequently.
* Use simple input values to test the menu and functions.

**Students to complete the following sections:**

Student Name: Prof. Khan

Class and Section: ITSS 3312, Section 501 or 002

**Example output** (copy and paste from the Eclipse console)

Welcome to Student and Course Management System!

This system will allow you to manage students and courses. Let’s start with the number of students this system will have: 5

\*\*\*Welcome to Student and Course Management System\*\*\*

Press ‘1’ for Student Management System (SMS)

Press ‘2’ for Course Management System (CMS)

Press ‘0’ to exit

1

\*\*\*Welcome to SMS\*\*\*

Press ‘1’ to add a student

Press ‘2’ to deactivate a student

Press ‘3’ to display all students

Press ‘4’ to search for a student by ID

Press ‘5’ to assign on-campus job

Press ‘6’ to display all students with on-campus jobs

Press ‘0’ to exit SMS

1

Enter first name: Joe

Enter last name: Smith

Enter level of the student: Freshman

Joe Smith has been added as a Freshman with ID 7

\*\*\*Welcome to SMS\*\*\*

Press ‘1’ to add a student

Press ‘2’ to deactivate a student

Press ‘3’ to display all students

Press ‘4’ to search for a student by ID

Press ‘5’ to assign on-campus job

Press ‘6’ to display all students with on-campus jobs

Press ‘0’ to exit SMS

1

Enter first name: Mary

Enter last name: Cane

Enter level of the student: Junior

Mary Cane has been added as a Junior with ID 6

\*\*\*Welcome to SMS\*\*\*

Press ‘1’ to add a student

Press ‘2’ to deactivate a student

Press ‘3’ to display all students

Press ‘4’ to search for a student by ID

Press ‘5’ to assign on-campus job

Press ‘6’ to display all students with on-campus jobs

Press ‘0’ to exit SMS

\*\*\*Welcome to SMS\*\*\*

Press ‘1’ to add a student

Press ‘2’ to deactivate a student

Press ‘3’ to display all students

Press ‘4’ to search for a student by ID

Press ‘5’ to assign on-campus job

Press ‘6’ to display all students with on-campus jobs

Press ‘0’ to exit SMS

3

Joe Smith

ID: 7

Level: Freshman

Status: Active

Mary Cane

ID: 6

Level: Junior

Status: Active

\*\*\*Welcome to SMS\*\*\*

Press ‘1’ to add a student

Press ‘2’ to deactivate a student

Press ‘3’ to display all students

Press ‘4’ to search for a student by ID

Press ‘5’ to assign on-campus job

Press ‘6’ to display all students with on-campus jobs

Press ‘0’ to exit SMS

2

Enter the ID for the student you want to deactivate: 6

Mary Cane has been deactivated

\*\*\*Welcome to SMS\*\*\*

Press ‘1’ to add a student

Press ‘2’ to deactivate a student

Press ‘3’ to display all students

Press ‘4’ to search for a student by ID

Press ‘5’ to assign on-campus job

Press ‘6’ to display all students with on-campus jobs

Press ‘0’ to exit SMS

4

Enter the student ID: 7

Joe Smith

ID: 7

Level: Freshman

Status: Active

\*\*\*Welcome to SMS\*\*\*

Press ‘1’ to add a student

Press ‘2’ to deactivate a student

Press ‘3’ to display all students

Press ‘4’ to search for a student by ID

Press ‘5’ to assign on-campus job

Press ‘6’ to display all students with on-campus jobs

Press ‘0’ to exit SMS

5

Enter student ID: 7

Enter job: TA

Enter job type: part-time

Joe Smith has been assigned part-time TA job

\*\*\*Welcome to SMS\*\*\*

Press ‘1’ to add a student

Press ‘2’ to deactivate a student

Press ‘3’ to display all students

Press ‘4’ to search for a student by ID

Press ‘5’ to assign on-campus job

Press ‘6’ to display all students with on-campus jobs

Press ‘0’ to exit SMS

6

Joe Smith

ID - 7

Part-time TA

\*\*\*Welcome to SMS\*\*\*

Press ‘1’ to add a student

Press ‘2’ to deactivate a student

Press ‘3’ to display all students

Press ‘4’ to search for a student by ID

Press ‘5’ to assign on-campus job

Press ‘6’ to display all students with on-campus jobs

Press ‘0’ to exit SMS

0

\*\*\*Welcome to Student and Course Management System\*\*\*

Press ‘1’ for Student Management System (SMS)

Press ‘2’ for Course Management System (CMS)

Press ‘0’ to exit

2

\*\*\*Welcome to CMS\*\*\*

Press ‘1’ to add a new course

Press ‘2’ to assign student a new course

Press ‘3’ to display student with assigned courses

Press ‘0’ to exit CMS

1

Enter course ID: 3311

Enter course name: Java 101

Confirmation: New course 3311 has been added

Press ‘1’ to add a new course

Press ‘2’ to assign student a new course

Press ‘3’ to display student with assigned courses

Press ‘0’ to exit CMS

2

Enter student ID: 7

Enter course ID: 3311

Confirmation: Joe Smith has been assigned course 3311

Press ‘1’ to add a new course

Press ‘2’ to assign student a new course

Press ‘3’ to display student with assigned courses

Press ‘0’ to exit CMS

3

Joe Smith

ID – 7

Courses: 3311

Press ‘1’ to add a new course

Press ‘2’ to assign student a new course

Press ‘3’ to display student with assigned courses

Press ‘0’ to exit CMS

0

\*\*\*Welcome to Student and Course Management System\*\*\*

Press ‘1’ for Student Management System (SMS)

Press ‘2’ for Course Management System (CMS)

Press ‘0’ to exit

0

Good Bye!!!

**Research and Analysis**. Describe the system including input, processing, arrays, classes and output in your own words (10 pts). Type response here.

This project is very similar to project 2, as it also uses a menu loop and object arrays. This project adds the use of files and exceptions handlings as well to make the system more efficient and useful. This system now can assign jobs to students, with user input of course, and put it in its own object array. Using the newly create Student\_Employee class, it inherits the Student class and it able to assign new employment to existing students. Also, the system adds all created students to a file on the user’s system. The system additionally adds the Course Management Menu, where a user can create a course by inputting the course code and name, assign a student to a course by inputting the specific student’s ID, and display all students who are assigned to a course.

**Design**. Describe the major steps for solving the problem (10 pts). Type response here.

Concerning the additional menus, I applied the same method I used in project 2 with the single menu, using a while loops that implements a switche and cases. Essentially, I added more while loops with a switch inside the original one, and it works the same by returning to the prior loop when the loop condition is reversed. After that, I worked on the Student\_Employment class, and in that class I created a new Student constructor that also includes job and job type String variables. Of course, this class inherited the Student class to be able to use the same methods and variables as the original. With this new constructor created, I was then able to make a Student\_Employment object array, and only added to it when the user assigned the student to a job. I initialized the object by using methods in the Student class to return the appropriate variables. After that, I worked on creating the first file for this project, the StudentReport.txt. By using the FileWriter in conjuction with BufferedWriter and PrintWriter, the user is able to add to the file without overriding or creating a completely new file. The final part of the SMS was displaying the students with jobs, and for that I simply used a for-loop and created a new method that printed the information for the respective students. After that, I worked on the Course class, and added in there the file creations for courses.txt and CourseAssignment.txt. Doing so will keep the files from trying to be created every instance of the project. For creating courses, I used the same methods for creating students and prompted the user for information and added it to the file via the Writers. The course assignment was essentially the same, doing so with the inclusion of the assigned class. The display that information, I utilized the FileReader object to make sure to read all the contents of the file.

I divided my program into few sections with subsections and denoted these using comments as follows:

* Classes
* Arrays
* File creation, reading and writing
* Search
* Menu Options
  + What happens upon pressing each options
* Variables declaration
* Gathering of inputs

**Coding**. Source code and output (25 pts). Submit your source code, your .java file, as a second file when you submit your project.

**Testing**. Describe how you tested this program (5 pts). Include descriptions of testing for specific calculations and/or algorithms, methods, and logic.

As I did with my last project, I tested my program by attempting every scenario I could do as a user, and doing so allowed me to discover many opportunities for exception handling. For the files, I used the FileReader object to display the file contents to make sure the information was recording as intended. For the object arrays, I ran the program several times, creating a different number of students and student employee every time, and printed the arrays to make sure everything was correct.

Create 2 members of your arrays to keep it simple as you build your program. Test everything.

**Submission Instructions**

**Submission**. Submit 2 files in eLearning. First, after writing/pasting responses above, save this Word document on your computer, and upload this document to your submission in elearning. When you save this document, save it with the final name as follows: [your name]\_Project 2.doc or .docx. It must be a Word document. Google docs are not accepted in eLearning.

Second, export your .java file to your computer and submit it as a second file in eLearning. The TA will download, review, and execute the code in this .java file as part of the grading.

To export a .java file to your computer:

In Eclipse on the Package Explorer, click on the .java file you want to send. Then click on File > Export > General / File System > Folder and file name for the file you are sending. Browse for the folder you want to copy it to, such as: C:\Users\[your\_name]\Documents\[folder for ITSS 3312]. Then click Finish. Then you can upload and submit it in eLearning from your Documents file.

**Rubric**

**Research and Analysis** (20 pts).

Key computations are listed and well-explained. Special Classes/methods used are identified and described on why they were used.

**Design** (20 pts).

Design is logical, well-organized, and easy to follow. Design should be evident in the code as comments are used to identify sections and sub-sections. Design write-up above describes the thinking and process for developing the code.

**Coding** (50 pts).

System fulfills the functional requirements (20 pts).

System fulfills the technical requirements (20 pts).

Other coding metrics (10 pts).

Code is clear and well organized, follows the design described above, includes comments in the code.

Code uses Java conventions for identifiers (var and method names), adding to the ease of following and understanding.

Code use Java conventions for indention and spacing, making code easy to read.

Code is efficient and uses the appropriate Java tools and statements.

System is user-friendly and logical; in other words, it is intuitive for the user.

**Testing** (10 pts).

Testing is comprehensive, covering important functions and components, and addresses key use cases.