Object Orientated Programming

Lab Manual 2

Instructor:
Mr. Samyan Qayyum Wahla

Learning Objectives:

- Student should be translate C++ syntax to Java
- Learn to write well documented and formatted code.

CLO:

- CLO1

Registration Number:

_____

Name:

_____

Guidelines/Instructions:

- Use of Djava is must in this lab.
- Write well commented code.
- Name of variables should be meaningful.
- Code should be well formatted.
- Create meaningful variable names. Add comments for readability. Indent each line of your code.
- Plagiarism/Cheating is highly discouraged by penalizing to both who tried and one who shared his/her code.

Reading and Practice Content

Example 1: Writing Data to File

Create a File

```
import java.io.File;  // Import the File class
import java.io.IOException;  // Import the IOException class to handle errors

public class CreateFile {
  public static void main(String[] args) {
    try {
      File myObj = new File("filename.txt");
      if (myObj.createNewFile()) {
        System.out.println("File created: " + myObj.getName());
      } else {
        System.out.println("File already exists.");
      }
    } catch (IOException e) {
      System.out.println("An error occurred.");
      e.printStackTrace();
    }
  }
}
```

Write to Text File

```
import java.io.FileWriter;   // Import the FileWriter class
import java.io.IOException;  // Import the IOException class to handle errors

public class LabManual2 {
  public static void main(String[] args) {
    try {
      FileWriter myWriter = new FileWriter("filename.txt");
      myWriter.write("Welcome to Object Orientated Programming Course");
      myWriter.close();
      System.out.println("Successfully wrote to the file.");
    } catch (IOException e) {
      System.out.println("An error occurred.");
      e.printStackTrace();
    }
  }
}
```

Example 2: Reading Data from File

Read Data from Created Text File

```
import java.io.File;  // Import the File class
import java.io.FileNotFoundException;  // Import this class to handle errors
import java.util.Scanner; // Import the Scanner class to read text files

public class LabManual2 {
  public static void main(String[] args) {
    try {
      File myObj = new File("filename.txt");
      Scanner myReader = new Scanner(myObj);
      while (myReader.hasNextLine()) {
        String data = myReader.nextLine();
        System.out.println(data);
      }
      myReader.close();
    } catch (FileNotFoundException e) {
      System.out.println("An error occurred.");
      e.printStackTrace();
    }
  }
}
```

Example 3: Structs equivalent in Java

Struct Equivalent

```
  class Student{
   public int Id;
   public String name;

}
```

Example 4: String tokenization

String Tokenization

```
 public static void main(String []args){
  String strMain = "I am Student of Computer Science";
   String[] splitArr = strMain.split(" ");
   for (int i=0; i < splitArr.length; i++)
   {
     System.out.println(splitArr[i]);
   }
 }
```

Example 5: Type Casting

Double to Integer conversion

```
    public static void main(String[] args) {
      // create double type variable
      double number = 20.65;
      System.out.println("User Input Double Value: " + number);
      try{
      // convert into int type
      int data = (int)number;
      System.out.println("The integer value after conversion: " + data);
      }
      catch (Exception ex){
            System.out.println("Invalid conversion");
      }
}
```

String to Integer conversion

```
  public static void main(String[] args) {
      // create string type variable
      String data = "10";
      System.out.println("The string value is: " + data);

      // convert string variable to int
      int num = Integer.parseInt(data);
      System.out.println("The integer value is: " + num);
  }
```

Example 6: Input and output using GUI

Display Output

```
import javax.swing.*;

public class Java_GUI {

  public static void main(String[] args) {

    JFrame frame;

      frame = new JFrame();
      JOptionPane.showMessageDialog(frame,"Hello World");
  }
}
```

Take Input from User and Display

```
  public static void main(String[] args) {
    String first_name;
    first_name = JOptionPane.showInputDialog("Enter First Name");
    String last_name;
    last_name = JOptionPane.showInputDialog("Enter Last Name");
    String full_name;
    full_name = "You name is " + first_name + " " + last_name;
    JOptionPane.showMessageDialog( null, full_name );
  }
```

Example 7: Different String Manipulation functions

String Concatenation

```
public static void main(String[] args) {
// Concatenation
  String str1 = "Computer";
  String str2 = "Science";
//  Method-1 (using concat function)
  String str3 = str1.concat(str2);
  System.out.println(str3);
// Method-2 (using “+” symbol)
  String str4 = str1 + " " + str2;
  System.out.println(str4);
}
```

Find the length of string	stringName.length()
Find the positon of character within string	stringName.indexOf(char)
	stringName.charAt(char)
Convert String to Lower Case	stringName.toLowerCase()
Convert String to Upper Case	stringName.toUpperCase()

Find the sequence of specified string within string	stringName.contains(“specified_string”)
Compare the specified string with case sensitive. If resultant value is 0, string is equal to specified string.otherwise no	stringName.compareTo(“specified_string”)
Compare the specified string without case sensitive. If resultant value is 0, string is equal to specified string.otherwise no	stringName.compareToIgnoreCase(“specified_string”)
Compare and return the suffix of string. Return true if the desired prefix exist in string	stringName.endsWith(“specified_string”)
Compare and return the suffix of string. Return true if the desired prefix exist in string	stringName.startsWith(“specified_string”)
Compares this string to the specified object. Return true if the desired string is equal to string	stringName.equals(“specified_string”)
Returns a new string resulting from replacing all occurrences of oldChar in this string with newChar.	replace(char oldChar, char newChar

Example 8: Java Lists

List provides an interface to store the ordered collection of data.

Basic syntax of list is

List<DataType> listName = new ArrayList<DataType>( );

Declaration of List and Assign Values

```
import java.util.*;
public class JavaList {

  public static void main(String[] args) {
 //list declaration
    List <String> strList = new ArrayList <String> ();
    strList.add("Now");
    strList.add("move");
    strList.add("towards");
    strList.add("list");
    System.out.println("List is: "+ strList);

  }
}
```

Update Values at Specific Index of List

```
public static void main(String[] args) {
 //list declaration
    List <String> strList = new ArrayList <String> ();
    strList.add("Now");
    strList.add("move");
    strList.add("towards");
    strList.add("list");
    System.out.println("List is: "+ strList);
   // Update List Values
```

```
    strList.set(3,"Java");
    System.out.println("Updated List is: "+ strList);
  }
```

Remove Object from List (through Object Name)

```
public static void main(String[] args) {
 //list declaration
    List <String> strList = new ArrayList <String> ();
    strList.add("Now");
    strList.add("move");
    strList.add("towards");
    strList.add("list");
    System.out.println("List is: "+ strList);
   // Update List Values
    strList.set(3,"Java");
    System.out.println("Updated List is: "+ strList);
// Remove By Index
    strList.remove(3);
    System.out.println("Updated List after removal of Specified index: "+ strList);
  }
```

Remove Object from List (through Index Position)

```
public static void main(String[] args) {
 //list declaration
    List <String> strList = new ArrayList <String> ();
    strList.add("Now");
    strList.add("move");
    strList.add("towards");
    strList.add("list");
    System.out.println("List is: "+ strList);
   // Update List Values
    strList.set(3,"Java");
    System.out.println("Updated List is: "+ strList);
// Remove By Index
    strList.remove(3);
    System.out.println("Updated List after removal of Specified index: "+ strList);
// Add new Object
    strList.add("OOP");
    System.out.println("List is: "+ strList);
    // Remove By Object
    strList.remove("towards");
    System.out.println("Updated List after removal of Specified String: "+ strList);
  }
```

Write a Java program in which you are required to implement the following software specifications.

1. Create a class CourseResult with the following data members
    a. String CourseId
    b. String CourseTitle
    c. int CreditHours
    d. int marks
    e. int semester
2. Create class Student with following data members
    a. String StudentName
    b. String RegistrationNumber
    c. String Degree
    d. List<CourseResult> list
3. Create a variable of student in main names s1. Your will only access s1 for all your tasks.

Constraints for each attribute are given below.

- **StudentName** //should be alphabetic, special characters and numbers are not allowed
- **RegistrationNumber** //Format should be like this: 2015-CS-888
- **Degree** //it should be MS, BS or BE
- **CourseId** // Format should be valid according to your course codes given in your LMS. For instance, software engineering lab has course ID of CS381L. Length of course code should be from 2 to 8 characters.
- **CourseTitle** // should be alphabetic. Length of course code should be from 10 to 35 characters.
- **CreditHours** // values from 1 to 3 are allowed
- **Marks //** values from 0 to 100 are allowed
- **Semester //** valid range is from 1 to 8

Create the following functions for the validation of above constraints.

1. boolean validateStudentName(String name);
2. boolean validateRegistrationNumber(String regNo);
3. boolean validateDegree(String degree);
4. boolean validateCourseId(String courseId);
5. boolean validateCourseTitle(String courseTitle);
6. boolean validateCreditHours(int creditHours);
7. boolean validateMarks(int marks);
8. boolean validateSemester(int semester);
9. boolean validateCourse (CourseResult course);

Define the following functions as well

1 **String getGrade(int marks)** – it should calculate grade based on marks using the following criteria.

a IF marks are less than 40 – Grade is F
b IF marks are between 40 and 50(exclusive) - Grade is D
c IF marks are between 50 and 55(exclusive) - Grade is C
d IF marks are between 55 and 60(exclusive) - Grade is C+
e IF marks are between 60 and 65(exclusive) - Grade is B-
f IF marks are between 65 and 70(exclusive) - Grade is B+
g IF marks are between 70 and 80(exclusive) - Grade is A-
h IF marks are above 80 - Grade is A

2 **double getGradePoints(String grade)** – function should return grade points using the following criteria

Grade	CoursePoints
A	4.0
A-	3.7
B+	3.3
B-	3.0
C+	2.7
C	2.3
D	1.0
F	0

3 **int getSemesters(List<CourseResult> list)** – it should return number of semesters based on course list

4 **int getTotalCreditHours(List<CourseResult>)** – it should return number of credit hours based on course list

5 **int getSemesterCreditHours(int semester, List<CourseResult>)** – it should return number of credit hours for a given semester based on course list

6 **double getSemesterGPA(int semester, List<CourseResult> list)** – calculate semester GPA according to following formula.

$$SemesterGPA = \frac{\sum \text{SemesterCourseGradePoints}}{\text{SemesterCourseGradePoints}}$$

7 **double getCGPA(int semester, List<CourseReuslt> list):** calculate GPA using the following formula

$$CGPA = \frac{\sum \text{CourseGradePoints}}{TotalCreditHours}$$

8 **String getSession(String regNo)** – extract session from RegistrationNumber

9 **String getDiscipline(String regNo)**: extract discipline from RegistrationNumber

10 **bool saveData(Student s1)** save data to file named studentdata.txt in the following format

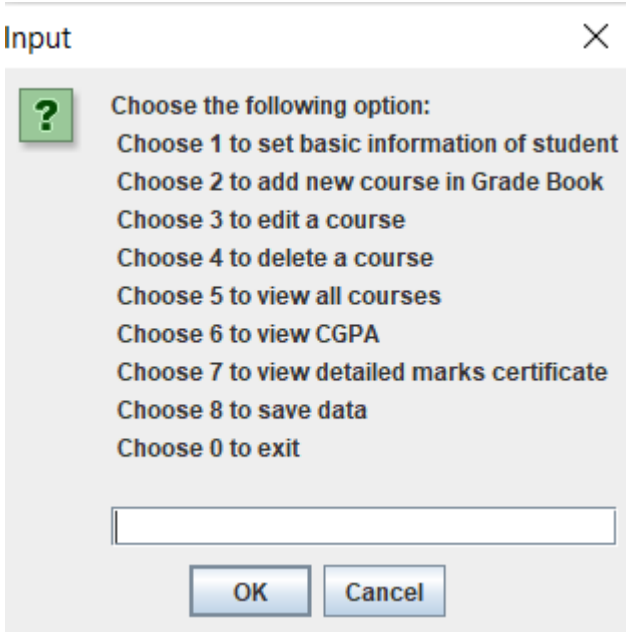
```
Samyan Qayyum,2009-CS-1,BS

CS142,Programming,3,1,86
CS162, Object Oriented,3,2,90
CS162L, Object Oriented Lab,1,2,90
```

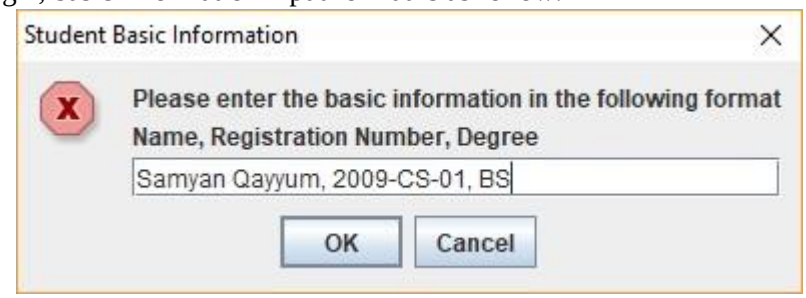
11 **Student readData()** – read the data from the above file and return object of student

12 **boolean saveDMC(Student s1)** – save DMC to file named dmc.txt

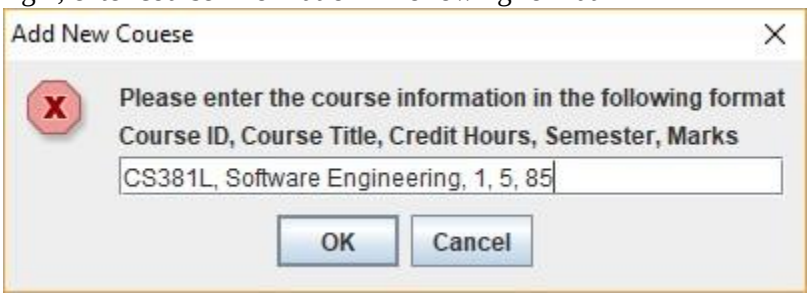
Output



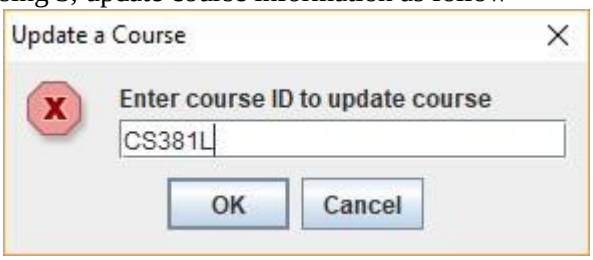
a. On pressing 1, basic information input format is as follow:

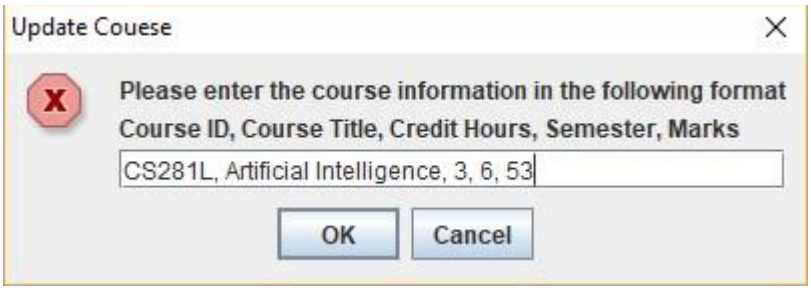


b. On pressing 2, enter course information in following format



c. On pressing 3, update course information as follow





d. On pressing 4, input dialog should ask for course ID to delete a course
e. On pressing 5, course should be shown on output dialog using toString() method in the following format

ID	Name	CH	Marks	Grade
CS381	Software Engineering	3	90	A
CS141	Computing Fundamentals	2	79	A-

f. On pressing 6, CGPA should be shown on output dialog.
g. On Pressing 7, DMC will be shown on output dialog in the following format

Name: Samyan Qayyum Degree: BS CS
Registration Number: 2009-CS-01
Session: 2009

Semester 1:

ID	Name	CH	Marks	Grade
MTH134	Calculus	3	90	A
CS141	Computing Fundamentals	2	79	A-
PHY101	Physics	3	75	A-

SGPA: 3.8125

Semester 2:

ID	Name	CH	Marks	Grade
MTH111	Linear Algebra	1	80	A
CS141	Programming Fundamentals	3	65	B+

SGPA: 3.475

CGPA: 3.7

h. On Pressing 8, courses and DMC will be saved to file
i. On Pressing 0, program will be closed after saving data
j. On program start, read the data from file if file exist.

What to submit

You are simply required to submit a source file **UetGradeBook.java** that includes the implementation of the above mentioned program. No extra file should be submitted.