**CS 2050 - Programming Project # 2**

Assign Date: Session 4.

Due Date: Session 7. Can be resubmitted ONE TIME until Session 11 if your first version is submitted by the first due date.

In this project, you construct several tests to check the code that you wrote in Project 1. Write pseudocode before coding and submit the former with your project.

**Problem Scenario**

Metropolitan State University of Denver needs a Java programmer to assist it with creating a grade book application. You have graciously volunteered to help. The university wants to track their students and their grades in various courses. For each student, the University wants to track this information:

1. Student id (String – should be a unique value)
2. Student first name (String)
3. Student last name (String)
4. Student email address (String – should be a unique value)

The University wants to ensure that they have a correct email address for each student. They want you to verify there is a “@” in the email address.

For each grade item, the University wants to track:

1. Student id (String – should match a student in the student list)
2. Grade item id (int – should be a unique value)
3. Course id (String)
4. Item type (String – must be one of the following: HW, Quiz, Class Work, Test, Final)
5. Date (String – format yyyymmdd)
6. Maximum score (String – the integer value must be greater than 0)
7. Actual score (String – the integer value must be in the range of 0 to maximum score)

**Program Requirements**

In Programming Project # 1, you created the Student class and GradeItem class. In this programming project, you test the code in those classes for each of these components:

1. Constructor
2. Get methods
3. equals method
4. toString method

**Note:** All your test code is in the main class. Do not make any changes to the Student and GradeItem classes created for Programming Project # 1 (unless you still have corrections to make). Your main class has this name:

FirstnameLastname\_02 (Use YOUR first and last name!)

The UML diagram for the main class is as follows:

|  |
| --- |
| FirstnameLastname\_02 |
| - student1: Student  - student2: Student  - gradeItem1: GradeItem  - gradeItem2: GradeItem  - INPUT\_FILENAME: String |
| + main (args: String): void  + processStudentData (String[] data): void  + processGradeItemData (String[] data): void |

**Note:** All methods and fields in the main class are declared static.

**Input Data File**

We use comma separated text files as input to this program. The first input file contains two lines of data about the Student and GradeItem classes in the following format:

STUDENT,ADD,studentId,firstName,lastName,emailAddress

GRADE ITEM,ADD,id,studentId,courseId,type,date,maxScore,actualScore

**Sample Input File 01**

STUDENT,ADD,900123456,Joe,Doe,joedoe@msudenver.edu

GRADE ITEM,ADD,1,900123456,23456,HW,20190112,100,95

Place this input file in the same folder as the main class and the two classes from Project 1. You can open input files by using the following String variable:

final String INPUT\_FILENAME = "Project\_02\_InputXX.txt";

where XX is the input file number, XX = 01, 02, 03, …. Specifically, the files for this project are named Project\_02\_Input01.txt, Project\_02\_Input02.txt, Project\_02\_Input03.txt, etc. You create multiple input files to test all the possible cases. See the addendum at the end of this document for a tip on easily changing the input file number (the “XX” in the sample file name above).

**Testing**

The main class runs three sets of tests. Before each test, display a message stating which test is being run. Also display a blank line, a row of \*s and another blank line between each of the tests.

Test Set 1: The first set of tests checks the constructors with parameters for each field and the toString() method for each class.

Test 1a:

Instantiate a Student object by hardcoding valid argument values of your choice in the call to the constructor. Store the reference to the Student object in the student1 field. Then use the System.out.println(student1.toString()) method to display the values to the console.

Test 1b:

Instantiate a GradeItem object by hardcoding valid argument values of your choice in the call to the constructor. Store the reference to the GradeItem object in the gradeItem1 field. Then use the System.out.println(gradeItem1.toString()) method to display the field values to the console.

Test Set 2: The second set of tests checks the constructors, getters and the reading of data from a second input data file. Steps required to complete the task are as follows:

1. The main method tries to open the input data file. Use the File and Scanner objects to open and read data from the input file. The main method should:
   1. Throw an exception if the data file is not found. The exception handler should display an error message including the name of the file that could not be found. Display the message using System.err.println method.
   2. If the file opens successfully then run the rest of the tests in Test Set 2.

Test 2a:

1. The main method reads the first line of data from the input data file.
2. Use the split() method to parse the line into an array of type String.
3. Check the first item in the array to make sure it is “STUDENT”.
4. If it is, call static method processStudentData passing it the array created in step 2.
5. processStudentData checks the second item in the array to make sure it is “ADD”.
6. processStudentData method creates the Student object using the data given in the array and stores the reference in student2 field.
7. processStudentData method uses the try/catch block to catch and display an error message if an IllegalArgumentException is thrown by the constructor in the Student class. Display the message using System.err.println method.
8. The main method displays Student data using that class’s get methods.
9. If the first field is not “STUDENT” or the second field is not “ADD”, display an appropriate message and move on to the next test.

Test 2b:

1. The main method reads the second line of data from the second input data file.
2. Use split() to parse the line into an array of String.
3. Check the first item in the array to make sure it is “GRADE ITEM”.
4. If it is, call a static method processGradeItemData passing it the array created in step 2.
5. processGradeItemData checks the second item in the array to make sure it is “ADD”.
6. processGradeItemData method creates the GradeItem object using the data given in the array and store the reference in gradeItem2 field.
7. processGradeItemData method uses the try/catch block to catch and display an error messages if an exception is thrown by the constructor in GradeItem class. Display the message using System.err.println method.
8. The main method displays the GradeItem data using that class’s get methods.
9. If the first field is not “GRADE ITEM” or the second field is not “ADD”, display an appropriate message and move on to the next test.

Close the input data file when you have completed the tests in Test Set 2, before running Test Set 3.

Test Set 3

The third set of tests checks the equals() methods for both classes.

Test 3a:

Create two equal instances of the Student object. Use the equals() method to compare the two Student objects (student1 and student2), and display a message stating that they are equal. Repeat with unequal instances.

Test 3b:

Create two equal instances of the GradeItem object. Use the equals() method to compare the two GradeItem (gradeItem1 and gradeItem2) objects, and display a message that are equal. Repeat with unequal instances.

**Notes on Testing**

Test these different scenarios too:

1. No test data input file (the FileNotFoundException).
2. Input file with an invalid email address for the Student.
3. Input file with an invalid GradeItem type.
4. Input file with an invalid value for the maximum score field, a value <= 0.
5. Input file with an invalid value for actual score, a value < 0 or > maximum score value.
6. Objects are not equal.

**Sample Output Displayed** (use your own toString() method to display an object the way you want.)

Running Test 1a:  
Student{studentId=900123456,firstName=Joe,lastName=Doe,  
emailAddres=joedoe@msudenver.edu}

Running Test 1b:  
GradeItem{gradeItemId=1,studentId=900123456,courseId=23456,  
type=HW,date=20190112,maxScore=100,actualScore=95}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Running Test 2a:  
Student Object Data  
900123456  
Joe  
Doe  
joedoe@msudenver.edu

Running Test 2b:

GradeItem Data  
1  
900123456  
23456  
HW

20190112  
100  
95

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Running Test 3a:

Student objects are equal: Student IDs are xxxx

Student objects are unequal: Student IDs are xxxx and yyyy

Running Test 3b:

GradeItem objects are equal: Student IDs are xxxx

GradeItem objects are unequal: Student IDs are xxxx and yyyy

**What to Submit**

Printed copies in this order: your main class (named FirstnameLastname\_02), your pseudocode, the Student class, the GradeItem class, the input files with their corresponding output file test results. Write your name and Project 2 in the upper right-hand corner of the top page. If you change the Student or GradeItem class, submit them, too, and highlight the changes. Resubmitting? Place the earlier submission(s) at the end and indicate on the top page of the new submission that you are resubmitting.

Addendum: Use the Command Line to Parameterize the Input File Name

This tip uses the FileNumber class which can be found online in the Projects folder. Copy it into your project folder’s source code or place it in a folder to where there is a class path. We stated above that the input file name has this format:

final String INPUT\_FILENAME = "Project\_02\_InputXX.txt";

where XX = 01, 02, 03, …

Rather than edit your program each time to change the “XX”, use the command line to set the value for XX. Here’s how.

Insert the following code before the final statement that declares the input file name.

// If the user (you!) didn’t supply a file number in the

// command line, use 01; otherwise use the supplied file number

String xx = ""; // Holds the input file number, 01, 02, …

// Assume we’ll use input file 01

xx = FileNumber.getFileNumber(args, “Command Line”);

final String INPUT\_FILENAME = "Project\_02\_Input"

+ xx + ".txt";

The parameter “args” is the same parameter in your main method declaration. Then, in the command line, enter the parameter 02 or 03 or 04 or … Or enter just a single digit and the code inserts a “0” at the beginning of the digit to form 01, 02, 03, …

Alternatively, you can code

xx = FileNumber.getFileNumber(args);

which has the same effect as the first call to getFileNumber. If you want to input the file number from the keyboard, simply code

xx = FileNumber.getFileNumber();

Note that you can assign *any* string to xx, like “Test” or “EmptyFile”. Your code then creates input file names like:

“Project\_02\_InputTest.txt” or

“Project\_02\_InputEmptyFile.txt”