|  |
| --- |
| Project 3 |
| Instructions |   |
| The third programming project involves writing a program to calculate the terms of the following sequence of numbers: 0 1 2 5 12 29 ... where each term of the sequence is twice the previous term plus the second previous term. The 0th term of the sequence is 0 and the 1st term of the sequence is 1. The interface to the program should be a GUI that looks similar to the following:https://learn.umuc.edu/content/enforced/213087-021974-01-2168-OL1-6383/proj3.jpg?_&d2lSessionVal=zVgkiyKvEpzswT1cR9Sw0eaIQThe pair of radio buttons allows the user to choose whether an iterative or recursive method is used to compute the term of the sequence. When the user enters a value for *n* and then clicks the *Compute* button, the *n*th term of the sequence should be displayed in the *Result* field. The *Efficiency* field should contain the number of calls to the recursive method when the recursive option is chosen and the number of iterations of the loop when the iterative option is selected.The *Iterative* radio button should be initially set to selected.When the window is closed, the efficiency values should be computed with values of *n* from 0 to 10 and written to a file. Each line of the file should contain the value of *n*, the efficiency of the iterative method for that value of *n* and the efficiency of the recursive method. The values should be separated by commas so the file can be opened with Excel and used to graph the value of the efficiencies for both the iterative and recursive options along the *y* axis with the value of *n* along the *x*-axis. The graph should be included as your test plan in the Word document that accompanies this project and should also contain a brief explanation of the observed results.The program should consist of two classes. The first class should define the GUI and should be hand-coded and not generated by a GUI generator. In addition to the main method and a constructor to build the GUI, an event handler will be needed to handle the *Compute* button click and another handler will be needed to produce the file described above when the window is closed. The latter handler should be an object of an inner class that extends the WindowAdapter class.The other class should be named Sequence. It should be a utility class meaning that all its methods must be class (static) methods and no objects should be able to be generated for that class. It should contain three public methods:1. The first method computeIterative should accept a value of *n* and return the corresponds element in the sequence using iteration.
2. The second method computeRecursive should accept a value of *n* and return the corresponds element in the sequence using recursion. This method with be a helper method because it will need to initialize the efficiency counter before calling the private recursive method that will actually perform the recursive computation.
3. The third method getEfficiency will return the efficiency counter left behind by the previous call to either of the above two methods.

Be sure that all instance and class variables are declared as *private*. Also any exceptions thrown by nonnumeric inputs should be properly handled.The application should run in the command line. Sometimes it doesn't because of differences in the computer environments, so in addition to the executable file, to ensure that your program receives the full credit, please provide a Word or PDF document with the following information with your submission:* IDE used and its version.
* Steps to run the program using this IDE, including any instructions on how to setup the program in  IDE, for instance, whether it requires a creation of a new project, etc.
* Any additional libraries required to run the application.
* Screenshots demonstrating working application (cut and paste into the Word document or submit separately).
* Test plan, including test cases and testing results, including expected results and the actual results. Table format is preferred over free-form.
* List all the requirements you haven’t been able to implement, lessons learned from this project and what, if anything, you'd do differently.
 |

|  |
| --- |
| Project 4 |
| Instructions |   |
| The fourth programming project involves writing a program to manage a student database. The interface to the program should be a GUI that looks similar to the following:https://learn.umuc.edu/content/enforced/213087-021974-01-2168-OL1-6383/proj4a.jpg?_&d2lSessionVal=zVgkiyKvEpzswT1cR9Sw0eaIQA combo box should allow the user to select one of the four database actions shown. The database should be implemented as a HashMap, with the ID field as the key and a student record consisting of a name and major as the value. The operation should be performed when the user clicks the *Process Request* button. If the user attempts to insert a key that is already in the database an error message should be displayed using a JOptionPane message dialog box. If the user attempts to delete, find or update a record that is not in the database, a message should also be displayed. After each successful operation is completed a JOptionPane window should be displayed confirming the success. In the case of a successful *Find* request, a window should pop up containing the student's ID, name, major and current GPA. When the user selects the *Update* request, the following JOptionPane windows should be displayed to gather information about a course that has just been completed:https://learn.umuc.edu/content/enforced/213087-021974-01-2168-OL1-6383/proj4b.jpg?_&d2lSessionVal=zVgkiyKvEpzswT1cR9Sw0eaIQ    https://learn.umuc.edu/content/enforced/213087-021974-01-2168-OL1-6383/proj4c.jpg?_&d2lSessionVal=zVgkiyKvEpzswT1cR9Sw0eaIQThis program must consist of two classes. The first class should define the GUI and handle the database interactions. It should be hand-coded and not generated by a GUI generator. The second class named Student, should define the student record. It must have instance variables for the student name, major and two variables that are used to compute the GPA. A variable that contains the total number of credits completed and a second variable that contains the total quality points, which are the numeric value of the grade received in a course times the number of credit hours. It should not contain the student ID. The class should have the following three methods:1. A constructor that is used when new student records are created. It should accept the name and major as parameters and initialize the fields that are used to compute the GPA to zero.
2. The second method courseCompleted should accept the course grade and credit hours and update the variables used to compute the GPA. It will be called when an *Update* request is made.
3. The third method should override toString and return a labeled string containing the student name, major and GPA.

Be sure that all instance and class variables are declared as *private*. Also any exceptions thrown by nonnumeric inputs should be properly handled. Finally when a student has not yet completed any course, the GPA should be displayed as 4.0.The application should run in the command line. Sometimes it doesn't because of differences in the computer environments, so in addition to the executable file, to ensure that your program receives the full credit, please provide a Word or PDF document with the following information with your submission:* IDE used and its version.
* Steps to run the program using this IDE, including any instructions on how to setup the program in  IDE, for instance, whether it requires a creation of a new project, etc.
* Any additional libraries required to run the application.
* Screenshots demonstrating working application (cut and paste into the Word document or submit separately).
* Test plan, including test cases and testing results, including expected results and the actual results. Table format is preferred over free-form.
* List all the requirements you haven’t been able to implement, lessons learned from this project and what, if anything, you'd do differently.
 |