**Courses Project**

**Assumptions:**

1. Total Course Hours = Number of courses selected \* (course hours)

**Note -** As in the real world total credit hours is the sum of each course hours, but as mentioned in the “CoursesProject.doc” file, I have calculated as (Number of courses selected) \* (course hours).

1. In the search method, the search string and the field should be exact match.
2. Free text fields (e.g. Name & Subject) allows numbers as well.

**Description**

* Installation - You can import the folder in any IDE and can run the program. You need to have Java 7 or higher.
* Input Method - It uses JOptionPane for input and to show Message Dialogue Box for output.
* Folder – This zip folder contains two sub-folders:
* “CoursesProject“: Contains the source code of the project.
* “CoursesProject (Executable JAR)”: Contains an executable jar file. Double click the “Courses (Executable).jar” and it should run the program directly.

**Program Flow**

* Firstly, it will prompt for the user’s name. If the textbox is empty it will show a warning and re-prompt. Once user name has been entered, it will show a course catalog with four options: add, edit, list and search.
* Add() : This method sequentially prompts user for the:
* Name: No empty string is allowed
* Id: Only 5 digits and the ID shouldn’t be in the file
* Length: Only digits
* Subject: No empty string is allowed

If any of the input is invalid, it will re-prompt for the input. Once, all the information is gathered from the user, it calls another function to append the new course in the file.

* Edit() : This method first prompts the:
* ID (of the course to be edited): Only 5 digits allowed

And then, it shows four buttons: Name, Id, Length and Subject. Based on what user selects, it shows an input box to enter the new value for one of the following fields:

* Name: No empty string allowed
* Id: Only 5 digits and the ID shouldn’t be in the file
* Length: Only digits allowed
* Subject: No empty string allowed

If any of the input is invalid, it will re-prompt for the input. And then lastly, it creates a new line of string with the new value in it and it calls the function to replace the old line with the new one.

* List(): This method parses the file and formats it into a string variable and returns that string.
* Search() : This method prompts the user to enter:
* Search Text: Free text field

Once, the user enters the text, then it will iterate through all the fields in the file and see if there's any exact match between the fields and the given text by the user. Once, it finishes searching, it will then print all the courses that has the given string. If no match found, it will display the message indicating that no match was found.

* Next, the program will prompt user, if they would like to perform any additional operation; if “Yes” the program will provide the four options again to the user and if user chooses “No”, it will call the addToFav() function
* addToFav(): This method prompts the user for:
* List of Ids: IDs separated by comma (e.g. “22222,33333”)

It validates the format and checks if all the Ids exists in the file. If there's any format error or at-least one of the Ids doesn't exist, then it will re-prompt the user to enter the lists of Ids. Finally, it will do the sum of the hours (Length) of the fav courses and will display to the user.

* The program ends here.

**Flowchart:**

(double click to open)

