Laboratory Manual 4

4 Java File and I/O Streams

Java class to perform File operation

Know the purpose of, how to use and when to use the following java classes to perform file operation along with common methods available for each class.

- Path
- Files
- File
- JFileChooser (Swing component)
- FileWriter
- FileReader
- PrintWriter
- BufferedWriter
- BufferedReader
- InputStream
- OutputStream
- FileInputStream
- FileOutputStream
- BufferedInputStream
- BufferedOutputStream
- DataInputStream
- DataOutputStream
- PrintStream

4.1 Reading a Text File

1. Create a NetBeans Project
    1. Create a project called "JavaTextEditor". See how to create a NetBeans project in Lab Manual 1.
    2. Design the GUI (See lab Manual 1. How to design GUL), The GUI should look as below

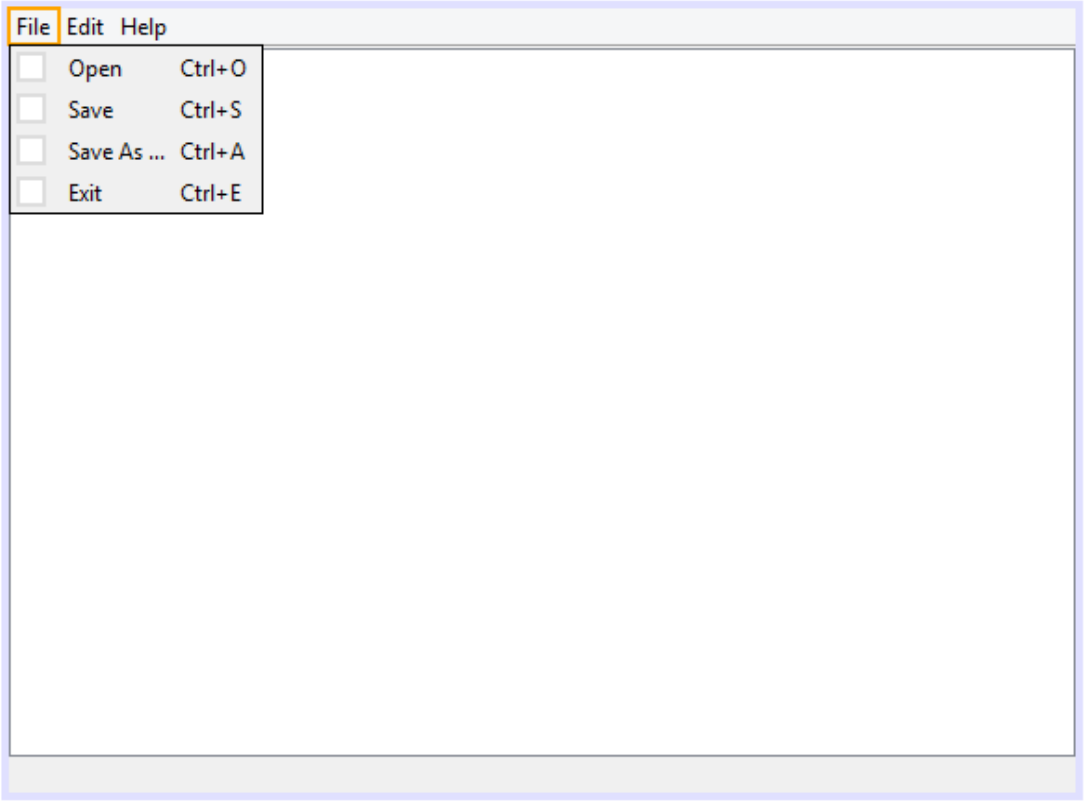

Fig : Java Text Editor Program user interface design

    3. Create "actionperformed" event for Open menu Item

4. Add Event handler/application logic code
   i. Add a JFileChooser class to your code
      ➢Use either the code or the GUI bulder to add JFileChooser

```
JFileChooser fOpen = new JFileChooser();
fOpen.setDialogTitle("Open File");
fOpen.set AcceptAllFileFilterUsed(true);
fOpen.setFileSelectionMode(JFileChooser.FILES_ONLY);
```

or

Drag and Drop File Chooser from the Pallet window to others group of the navigator windows
Change the default variable and set other important properties of the jFileChooser component

- Common properties
  - currentDirectory
  - fileSelectionMode
  - dialogTitle
  - dialogType
  - approveButtonText
  - AcceptAllFileFilterUsed

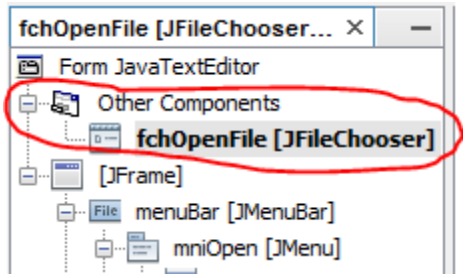

Fig : Add File Chooser to your design

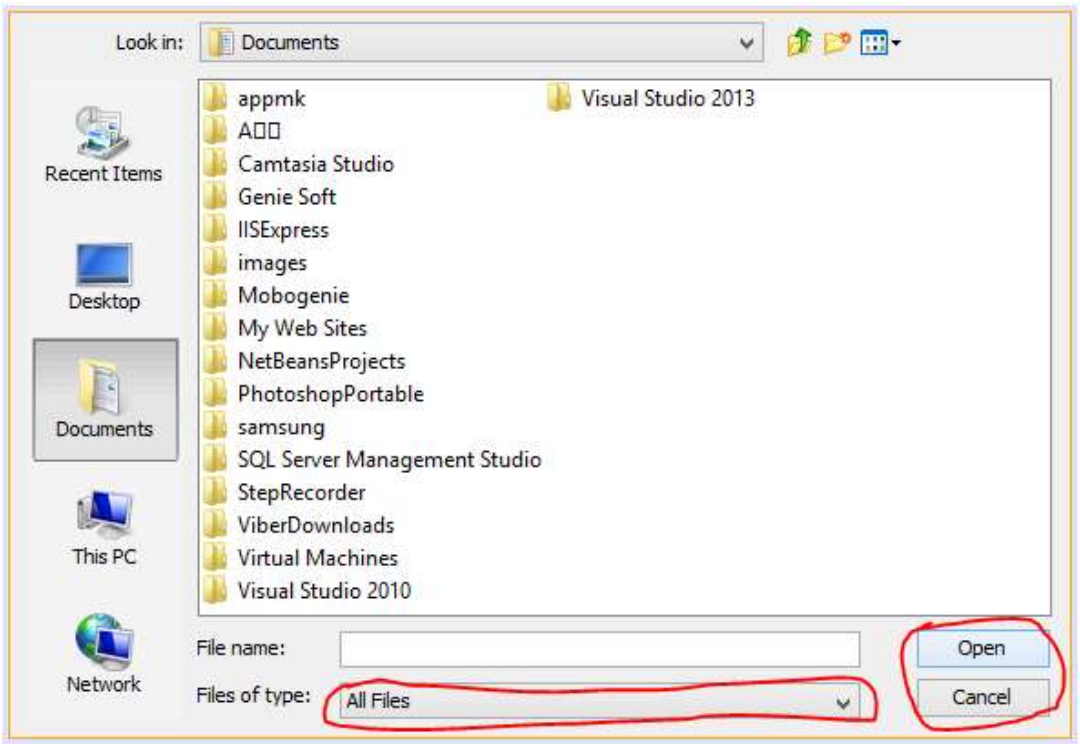

Fig : JFileChooser design

ii. Declare integer variable to accept the action result from the user and open the dialog

- int result = fOpen.showOpenDialog(this); //the dialog created by code
- //or
- // int res = fchOpenFile.showOpenDialog(this);//the dialog created by GUI builder
- **Run and test your program, at this time, it only displays the open file browser, nothing else**

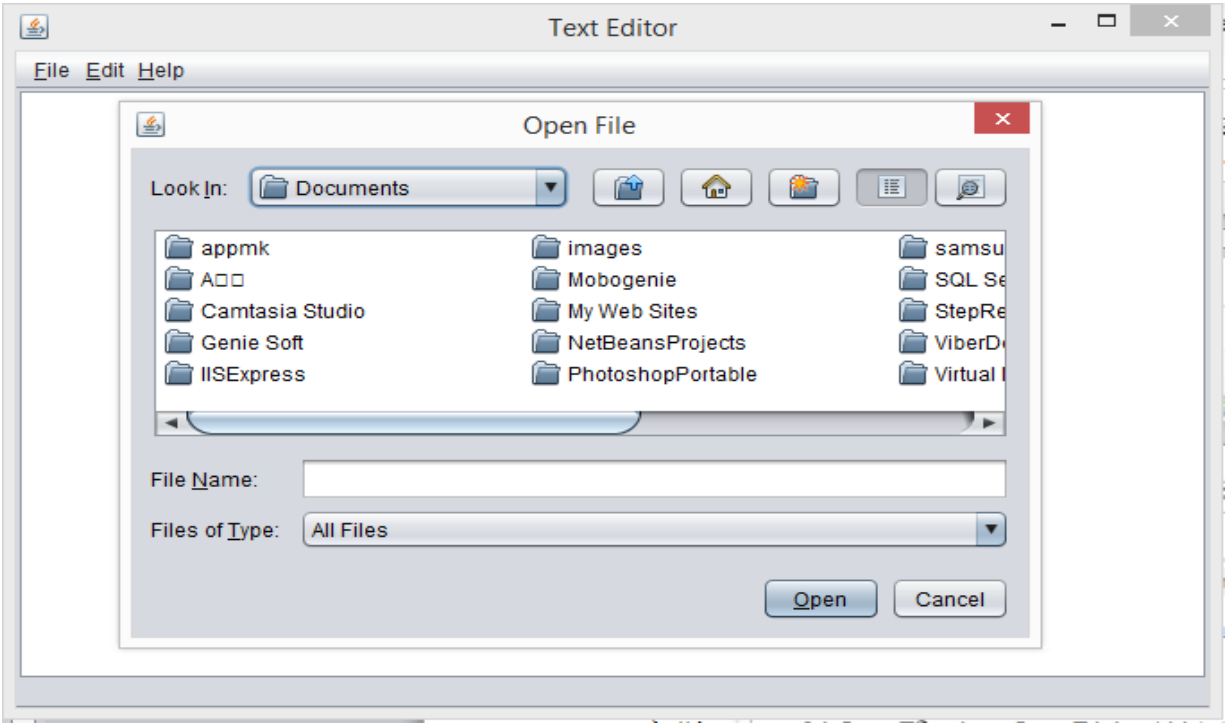

Fig : JFileChooser dialog window

iii. Accept the response of the user for the dialog box

- Check with condition statement
- Get the path and assign to Path class object
- Create File class object and get the selected file
- At this stage the code looks as follow

```
private void openMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser fOpen = new JFileChooser();
    fOpen.setDialogTitle("Open File");
    fOpen.setAcceptAllFileFilterUsed(true);
    fOpen.setFileSelectionMode(JFileChooser.FILES_ONLY);
    int result = fOpen.showOpenDialog(this);
    //int result = fchOpenFile.showOpenDialog(this);
    if(result == JFileChooser.APPROVE_OPTION){
        try {
            File fil = fOpen.getSelectedFile();

        } catch (Exception e) {
        }
    }
```

Fig : Open Menu item event handler

iv. Create a file reader class and try to read the content of the file, display to the text editor.
- ➢Create BufferedReader class object
- ➢Read the content line by line, till the end
- ➢Put the content to the text editor content area

```
if(result == JFileChooser.APPROVE_OPTION){
    try {
        File fil = fOpen.getSelectedFile();
        BufferedReader br = new BufferedReader(new FileReader(fil.getPath()));
        String line;
         while ((line = br.readLine()) != null) {
               txpContent.setText(txpContent.getText() + line + '\n');
            }
    } catch (Exception e) {
    }
```

5. Run, and test your program, try to open a text(.txt) file

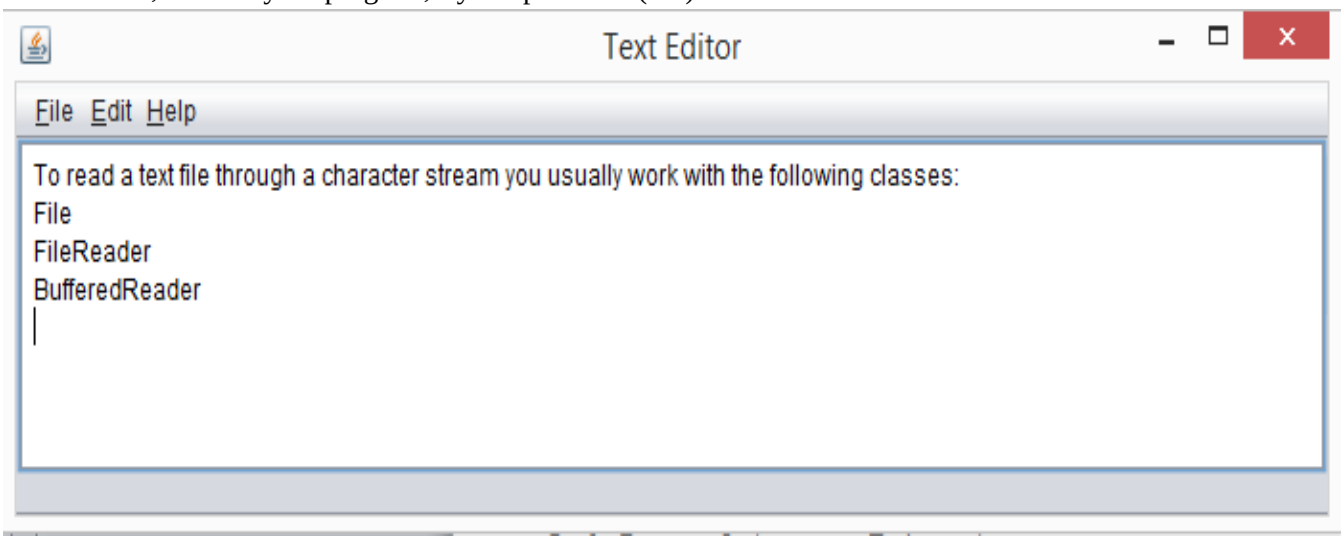

Fig : Java Text Editor/ Text file reader

2. Extend FileFilter abstract class

- ➢ The file chooser dialog box includes a Files of Type drop-down list filter that the user can choose to control what types of files are displayed by the chooser.
- ➢ public boolean abstract accept(File f)
  - ➢ implement this method to return true if you want the file displayed in the chooser, false if you don't want the file displayed
- ➢ public String abstract getDescription()
  - ➢ implement this method to return the description string that is displayed in the Files of Type dropdown list in the chooser dialog box

1. Create a class called "FileTypeFilter" which implements FileFilter abstract class
2. Implement the to override accept and getDescription method of the class FileFilter abstract

```
public class FileTypeFilter extends FileFilter {

    @Override
    public boolean accept(File f) {
        if (f.isDirectory()) {
            return true;
        }
        String name = f.getName();
        if (name.matches(".*\\.txt")) {
            return true;
        } else {
            return false;
        }

    }

    @Override
    public String getDescription() {
        return "Text files (*.txt)";

    }

}
```

Fig : implementing accept and getDescription methods of FileFilter Class

3. Pass this class to the addChoosableFileFilter or setFileFilter method.

```
fOpen.setAcceptAllFileFilterUsed(true);
fOpen.addChoosableFileFilter(new FileTypeFilter());
fOpen.setFileSelectionMode(JFileChooser.FILES_ONLY);
int result = fOpen.showOpenDialog(this);
```

Fig : passing instance of FileFilter class

4. Run and look the difference from the previous one
5. Set the setAcceptAllFileFilterUsed to false and see the difference in the File Type list of the open dialog window.

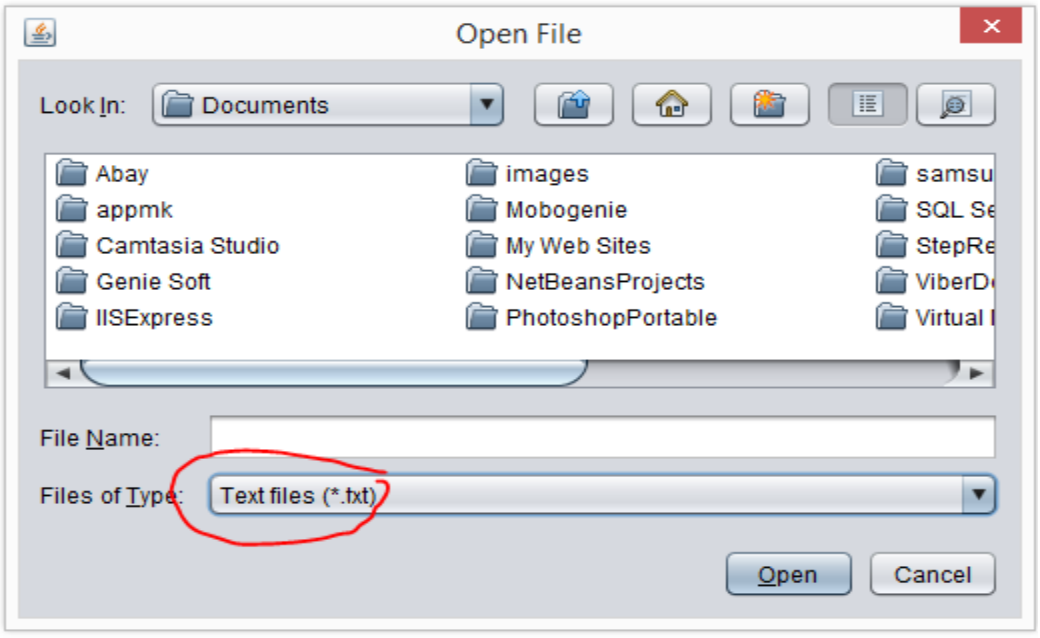

Fig : Using file type filter

4.2 Writing to a Text File

1. Add event for save menu item
2. Write a code handler/application logic code for saving the content of the text editor to a text file
   i. Create a JFileChooser save dialog widow
   ii. Accept the user response to the save dialog window
   iii. Get the path and file name
   iv. Write the content of the text editor to a text file
   v. Free up recourse

```
private void mniSaveActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser fs = new JFileChooser();
    fs.setDialogTitle("Save a File");
    fs.addChoosableFileFilter(new FileTypeFilter());
    fs.setAcceptAllFileFilterUsed(false);
    int result = fs.showSaveDialog(this);
    if (result == JFileChooser.APPROVE_OPTION) {
        String cont = txpContent.getText();
        Path path = Paths.get(fs.getSelectedFile().getPath());
        File f = path.toFile();
        try {
            PrintWriter fw = new PrintWriter(new BufferedWriter(new FileWriter(f, true)));
            fw.write(cont);
            fw.flush();
            fw.close();
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, e.getMessage());
        }
    }
}
```

Fig : write a text file

3. Run and test your program, check the saved file by opening with windows text editor (Note pad, Word pad …)

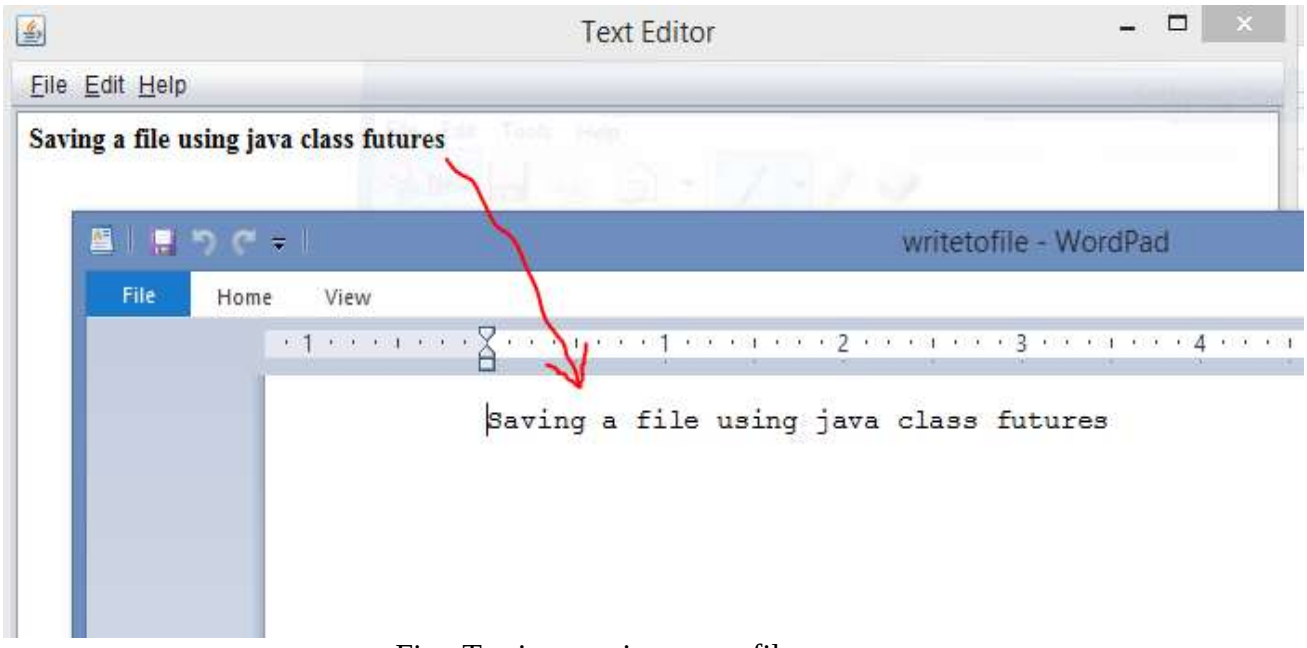

Fig : Testing a write to text file program

4.3 Lab practice on File (5 pts)

1. For the project, CGPA Calculator, we worked on the previous class,
   i. Write an application logic/code that save course information to file "course.txt" in tab delimited format, when a user clicks on the "Save" button.
   ii. Add a button control called "Load", and write code for the "Load" button actionPerformed event that loads course information from "course.txt" file to course list table when a user clicks on it. The program user interface looks like the following figure.

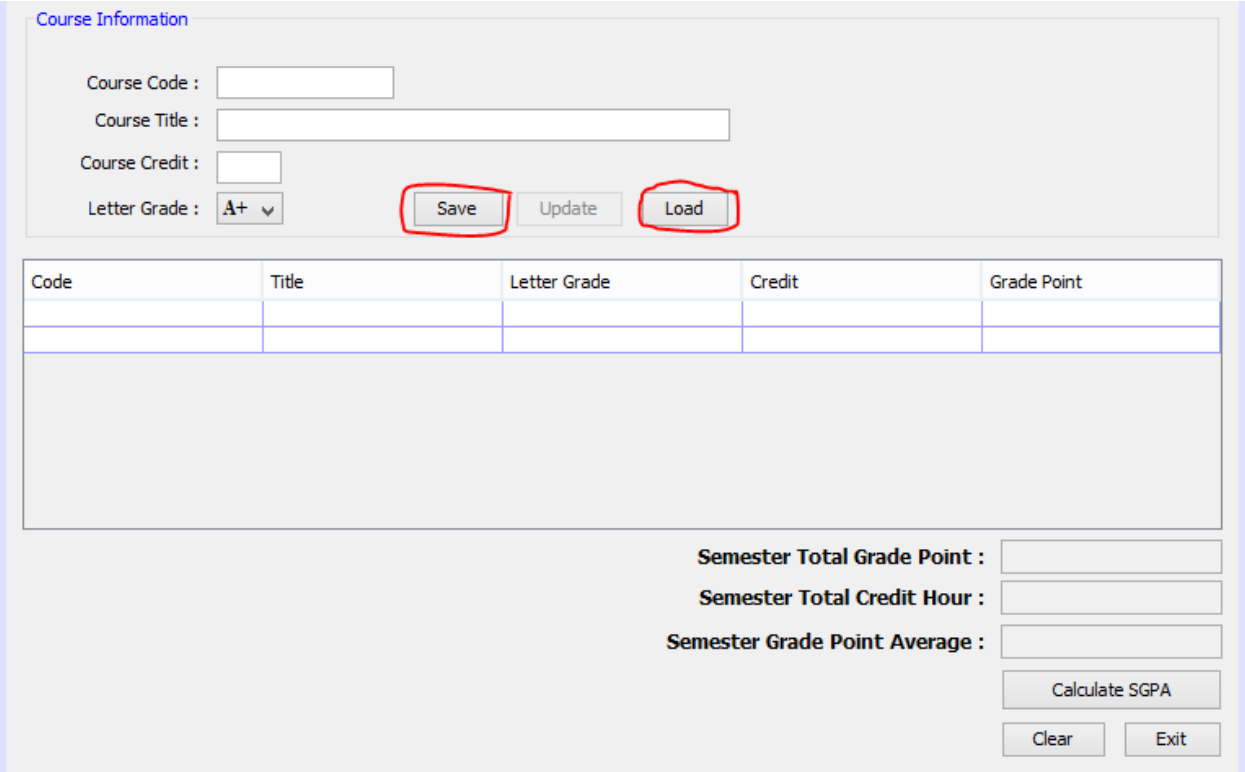

Fig : CGPA Calculator program