**INTEGRATED DEVELOPMENT ENVIRONMENT**

**( IDE )**

IDEs typically present a single program in which all development is done. This program typically provides many features for authoring, modifying, compiling, deploying and debugging software. The aim is to abstract the configuration necessary to piece together command line utilities in a cohesive unit, which theoretically reduces the time to learn a language, and increases developer productivity. It is also thought that the tight integration of development tasks can further increase productivity. For example, code can be parsed while being written, providing instant feedback on syntax errors.

* 1. **About the IDE**

Integrated development environment (IDE) (also known as integrated design environment, integrated debugging environment or interactive development environment) is a [software application](http://en.wikipedia.org/wiki/Software_application) that provides comprehensive facilities to [computer programmers](http://en.wikipedia.org/wiki/Computer_programmer) for [software development](http://en.wikipedia.org/wiki/Software_development). An IDE normally consists of:

* a [source code editor](http://en.wikipedia.org/wiki/Source_code_editor)
* a [compiler](http://en.wikipedia.org/wiki/Compiler) and/or an [interpreter](http://en.wikipedia.org/wiki/Interpreter_%28computing%29)
* [build automation](http://en.wikipedia.org/wiki/Build_automation) tools
* a [debugger](http://en.wikipedia.org/wiki/Debugger)

It is very much realistic, having full provision of Graphical User Interfaces ( GUI ) that helps the user to make a instant reaction to different events by just clicking.

The provision for shortcuts have also been defined in case the user is comfortable with the shortcuts.

* 1. **Software Description**

The application provides the provision to user for creating applications with the java technology that has in build complier and Interpreter support to compile the code that will generate the .class file and correspondingly will show the output in the textbox or the console.

* 1. **Modules of IDE**

Different modules in this application are :-

* Sign In & Sign Up

Users that are already registered for using this application can use it directly by providing their access Id and Password while others can Sign Up in a simple fashion in order to have their access.

* Writing Code

A text area is provided for users to write their logical code using java only.

* Compiler

Complier has been embedded for the written code to be compiled and a byte code(.java file) is created in the same directory.

* Interpreter

Interpreter has been embedded to Interpret the byte code created by the Compiler.

* Intellisense

Intellisense has been provided that lower down the task of writing the entire syntax. ie it incorporates reusability of the correct syntax.

* Report Issue

The users can report their queries to the administrator of our application.

* KeyBoard Shortcuts

Users can have the knowledge of all the shortcuts used in the project and can employ the same to save time and effort.

**DESIGNING OF SOFTWARE**

**Design** is a process of problem solving and planning for its solution. After the

purpose and specifications of network are determined, developers need to design

or employ designers to develop a plan for a solution.

Designing before implementation stage facilitates understanding of the problem ,

the implementation becomes easier and organized.

Designing in our case will include a network diagram containing all the required

components to meet our purpose. Along with the diagram, the hardware and

software requirements, the description of the components used and how the

network will work will also be mentioned.

**2.1 Hardware Requirements**

* Processor: Pentium IV or higher
* Ram: 512MB or higher
* Disk Space : 4 GB or higher

**2.2 Software Requirements**

* Net Beans (any version )
* JDK ( Java Development toolkit )
* JVM ( Java Virtual Machine )
* MS Access and MySQL

**2.3 System Design**

Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development.Systems design is therefore the process of defining and developing systems to satisfy specified requirements of the user. Until the 1990s systems design had a crucial and respected role in the data processing industry. In the 1990s standardization of hardware and software resulted in the ability to build modular systems. The increasing importance of software running on generic platforms has enhanced the discipline of software engineering.

**2.3.1 Compiler**

Complier has been embedded for the written code to be compiled and a byte code(.java file) is created in the same directory.

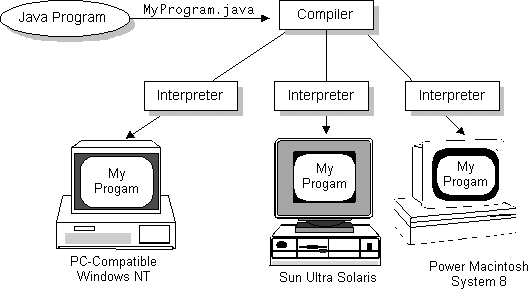
****

Fig :- 2.1 Compiler

**2.3.2 Interpreter**

We can run Java on most platforms provided a platform must has a Java interpreter. That is why Java applications are platform independent. Java interpreter translates the **Java bytecode into the code that can be understood by the Operating System**. Basically, A Java interpreter is a software that implements the Java virtual machine and runs Java applications. As the Java compiler compiles the source code into the Java bytecode, the same way the Java interpreter translates the Java bytecode into the code that can be understood by the Operating System.

When a Java interpreter is installed on any platform that means it is **JVM** (Java virtual machine) enabled platform. It (Java Interpreter) performs all of the activities of the Java run-time system. It loads Java class files and interprets the compiled byte-code.The interpreter also serves as a specialized compiler in an implementation that supports dynamic or "just in time," compilation which turns Java byte-code into native machine instructions.

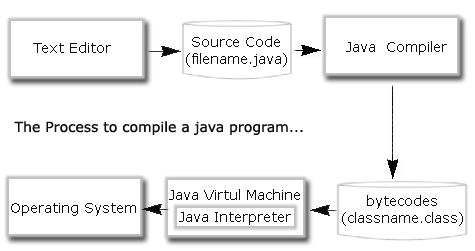


Fig :- 2.2 Interpreter

**IMPLEMENTAION**

An implementation is a realization of a technical specification or algorithm as a program, software component, or other computer system through programming and deployment. Many implementations may exist for a given specification or standard. For example, web browsers contain implementations of World Wide Web Consortium-recommended specifications, and software development tools contain implementations of programming languages.

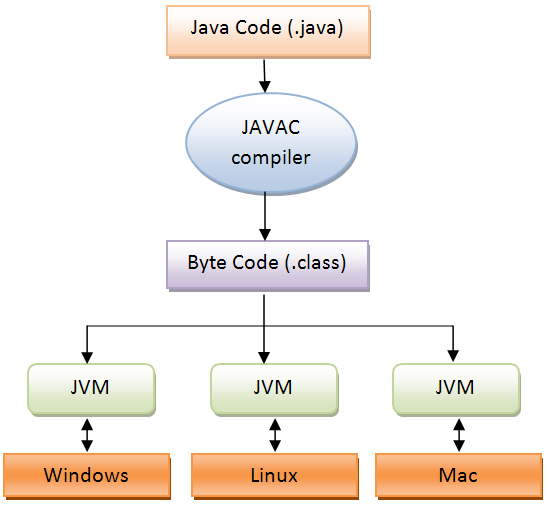
****

Fig :- 2.3 Implementation

**Complier**

A textbox has been provided where user can write his code with the help of GUI ( Graphical User Interfaces ).

After the code has been written, user can compile the program. An another textbox is provided where the errors are going to be displayed after compilation. If the compilation has been successful then a corresponding .class file will be generated in the default directory of the system.

**Interpreter**

After Successful compilation of the code, Interpreter has been called to execute the .class file generated by the compiler previously and correspondingly the output is displayed in the text area provided.

**Intelliscence( Static )**

User need not to write the entire syntax of the code. You can use the shortcut keys that will automatically print the rest of the syntax.

**Saving Option**

You can save your program at the backend also rather than storing in the system directories. That is mysql has been used for the purpose that keeps record of all the files that are generated with our application.

**SUMMARY AND CONCLUSIONS**

IDE is designed to maximize programmer productivity by providing tightly-knit components with similar [user interfaces](http://en.wikipedia.org/wiki/User_interface). However, because an IDE is a complicated piece of software by its very nature, this higher productivity only occurs after a lengthy learning process.

It has piece together command line utilities in a cohesive unit, which theoretically reduces the time to learn a language, and increases developer productivity. It is also thought that the tight integration of development tasks can further increase productivity. For example, code can be parsed while being written, providing instant feedback on syntax errors.

**4.1 Features of the Project**

* Embedded compiler and Interpreter
* Security
* Portability
* Graphical user interface ( GUI )
* Database Connectivity
* features like undo, redo etc.

**4.2 Future Aspects of the Project**

* Transfer byte code via web
* INTELLISENSE ( Dynamic ) For Reusability of the code.

**REFERENCES**

1. http://apl.jhu.edu/~hall/java/IDEs.html

2. http://faq.programmerworld.net/

3. http://netbeans.org

4. http://www.cs.unc.edu/~jbs/images/java/java-dataflow.GIF

5. http://www.javaworld.com/javaworld/jw-05-1997/jw-05-indepth.html

6. http://www.jcreator.com/

7. http://www.roseindia.net/java/java-introduction/javatools/java-inetrpreter.shtml

**APPENDIX**

**6.1 Appendix A (Source code)**

**6.1.1 User Authentication**

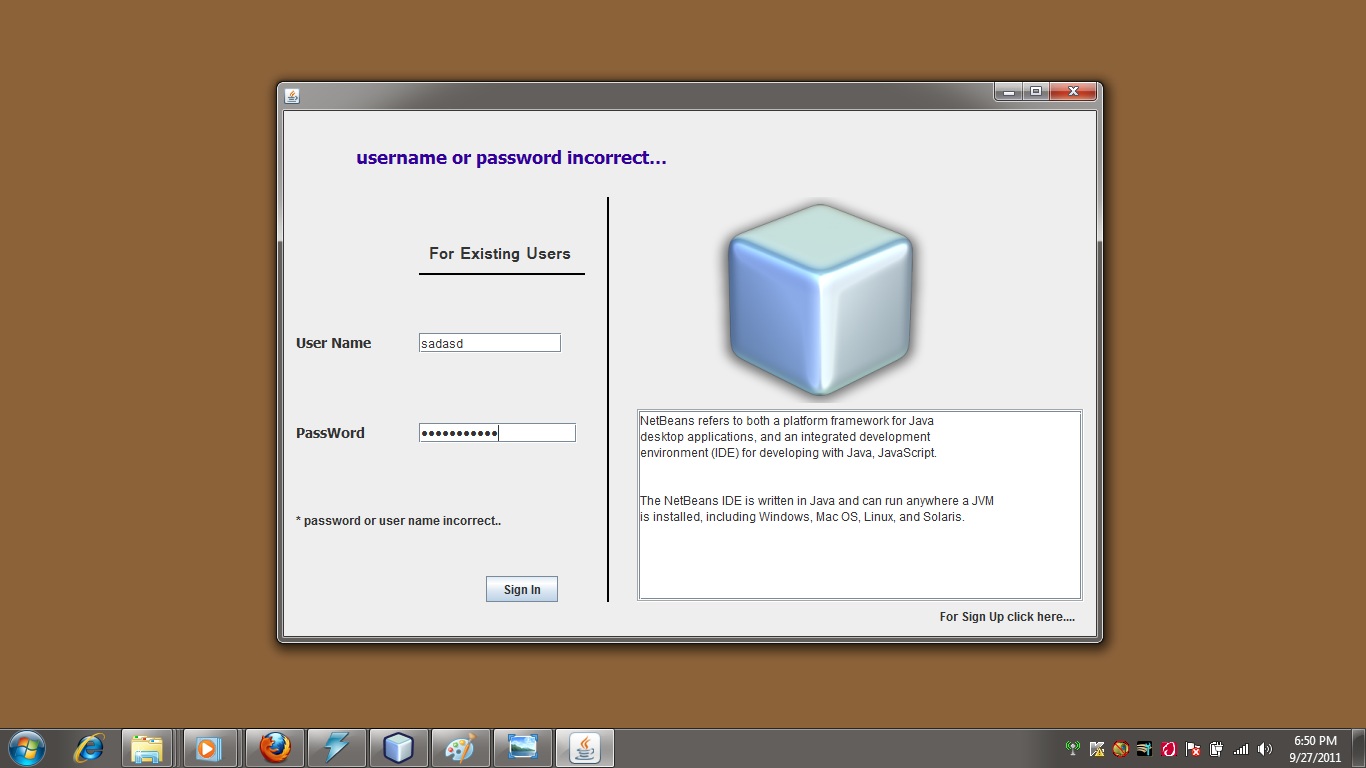
****

Fig :- 2.4 Authentication

**6.2.2 Main Interface**

This is the main interface where user can do writing compiling etc. More over recent projects provision has been enabled to give the view of recent projects to the users.

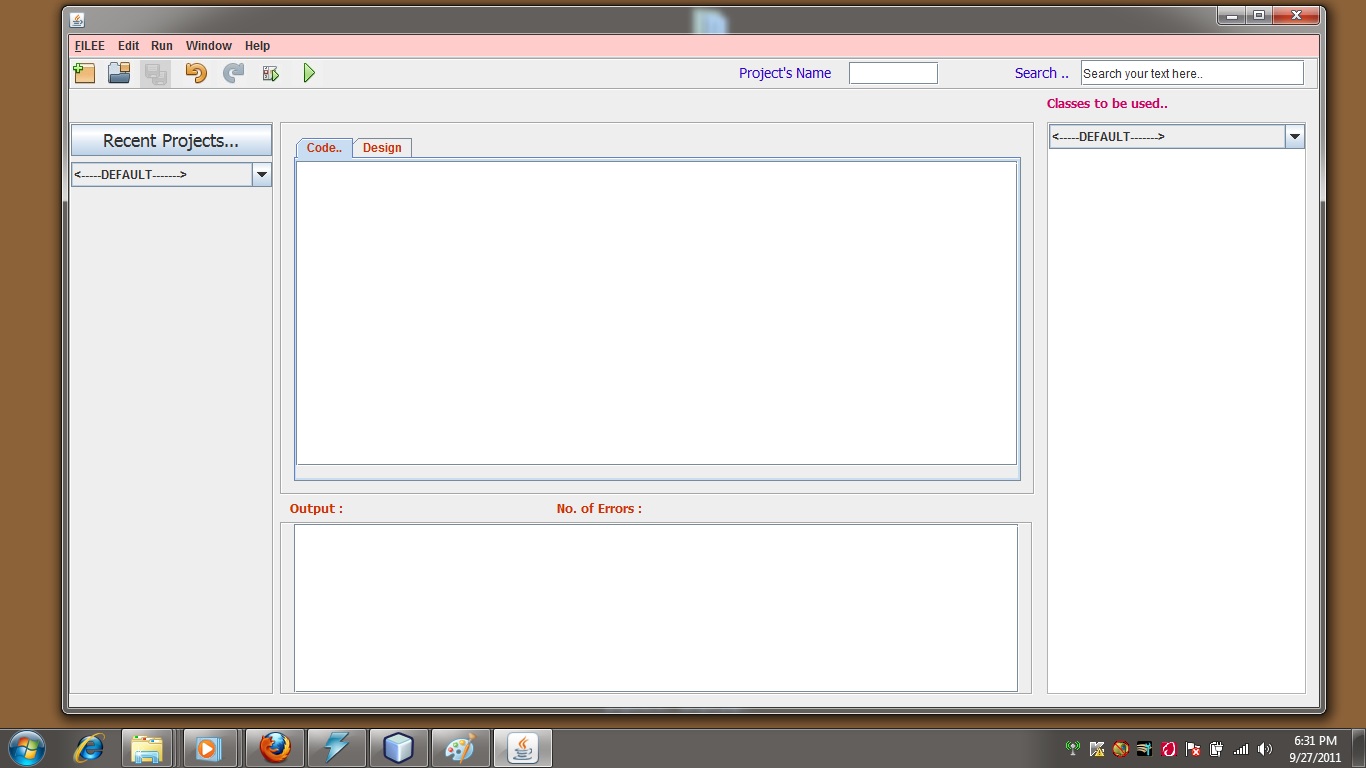
****

Fig :- 2.5 Main Interface

**6.2.3 Creating a new project**

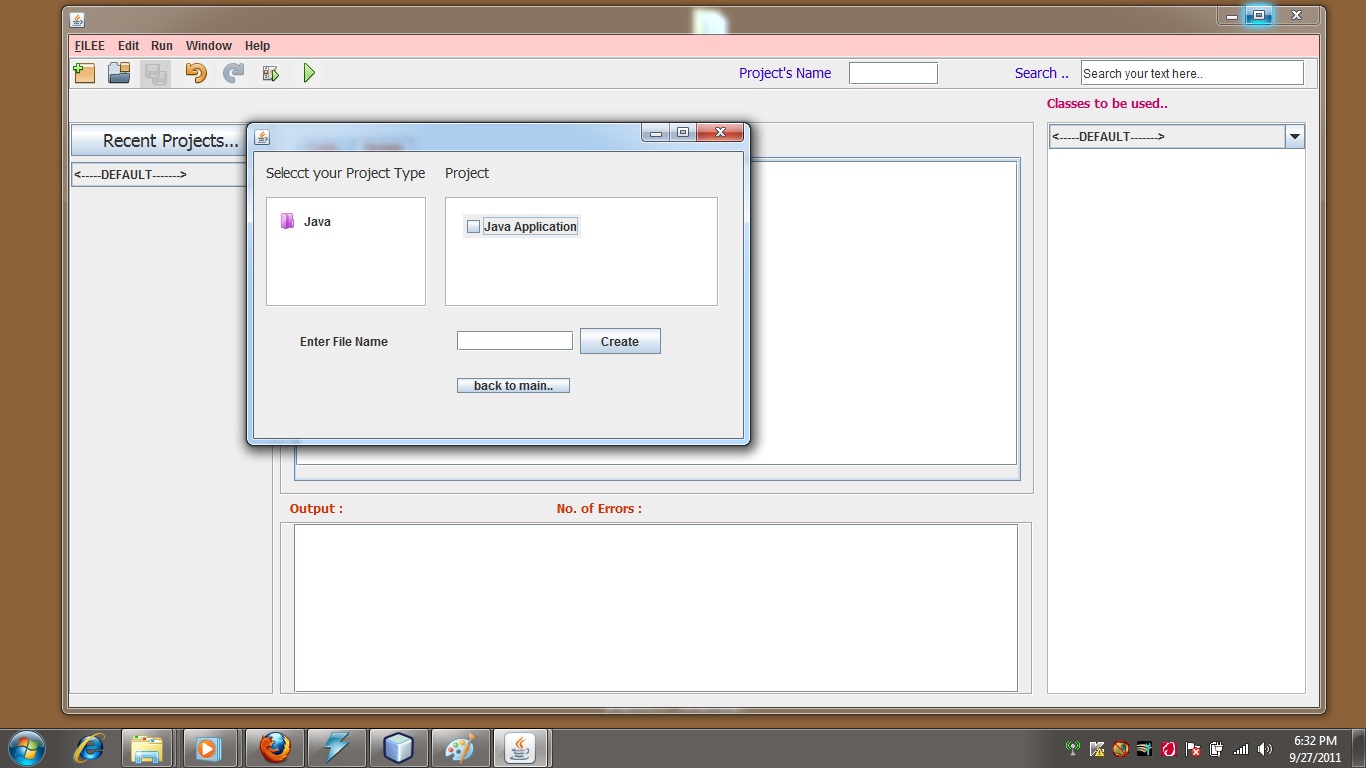
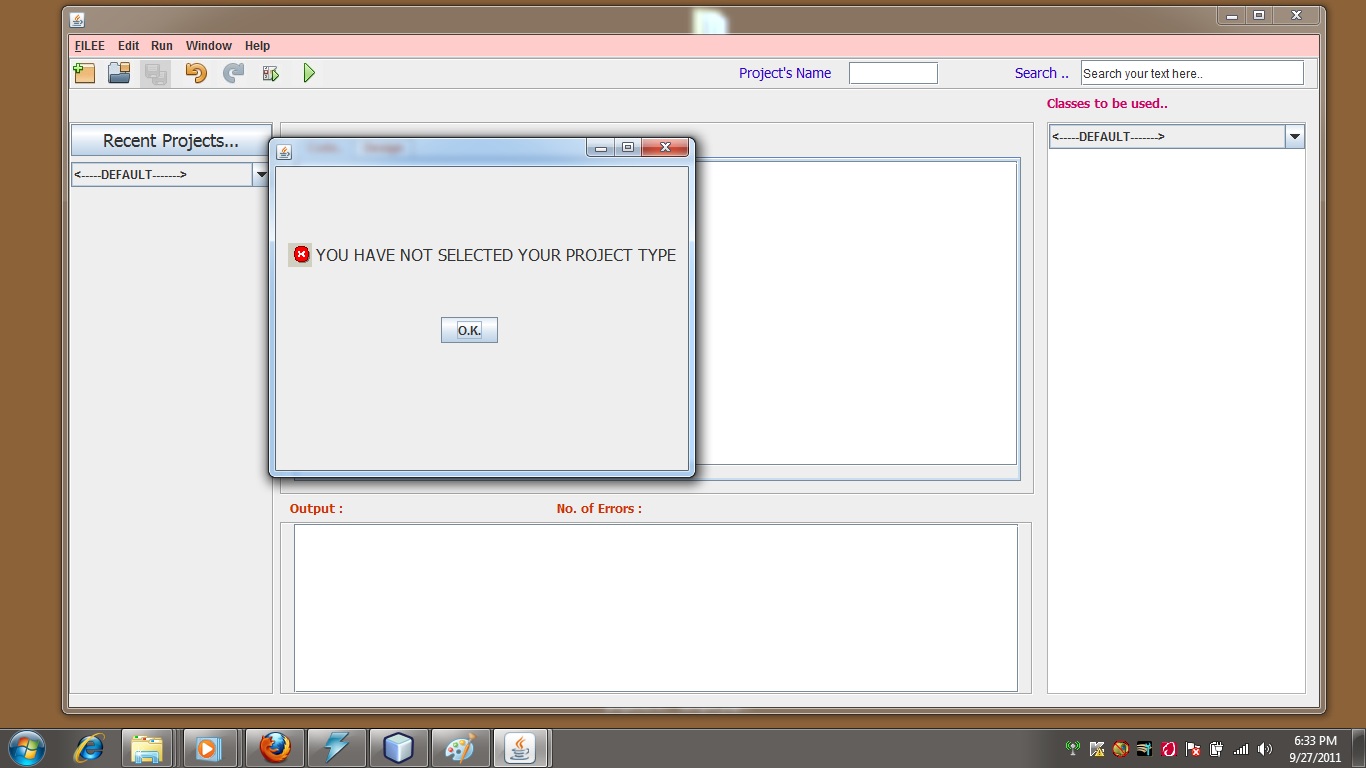


Fig :- 2.6 New Project

****

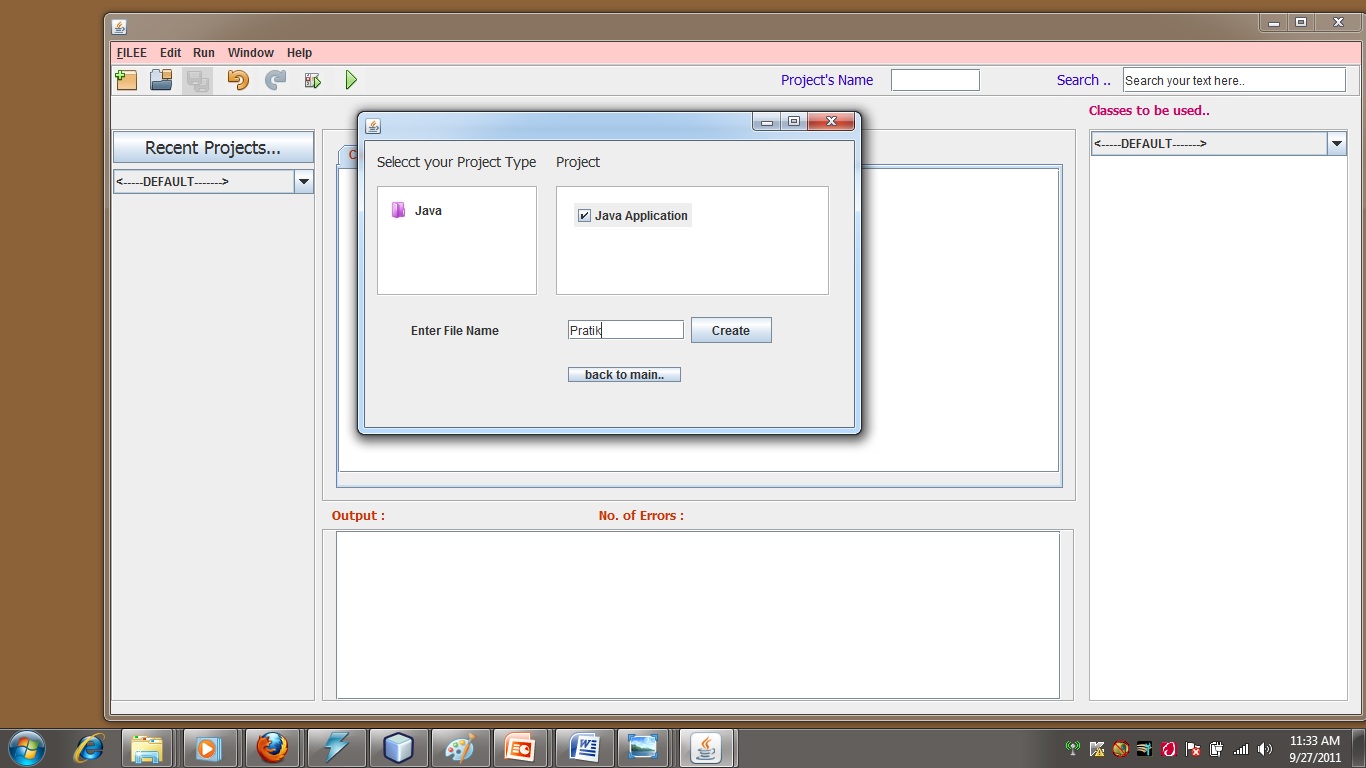
****

Fig :- 2.7 Error

**6.2.4 Write Code here**

A textbox has been provided where user can write his code with the help of GUI ( Graphical User Interfaces ).

After the code has been written, user can compile the program. An another textbox is provided where the errors are going to be displayed after compilation. If the compilation has been successful then a corresponding .class file will be generated in the default directory of the system.

After Successful compilation of the code, Interpreter has been called to execute the .class file generated by the compiler previously and correspondingly the output is displayed in the text area provided.

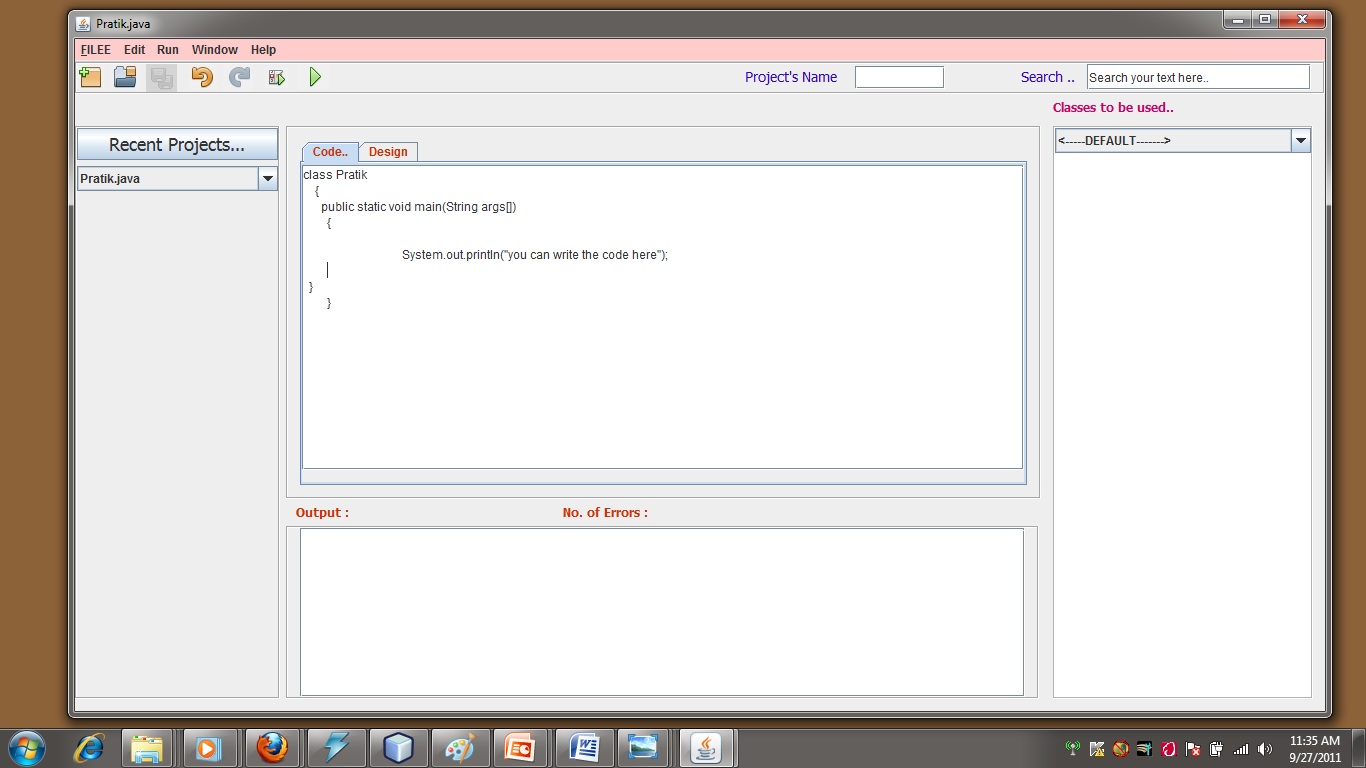
****

Fig :- 2.8 Writing code

**6.2.5 Save the Project**

You can save your program at the backend also rather than storing in the system directories. That is mysql has been used for the purpose that keeps record of all the files that are generated with our application.

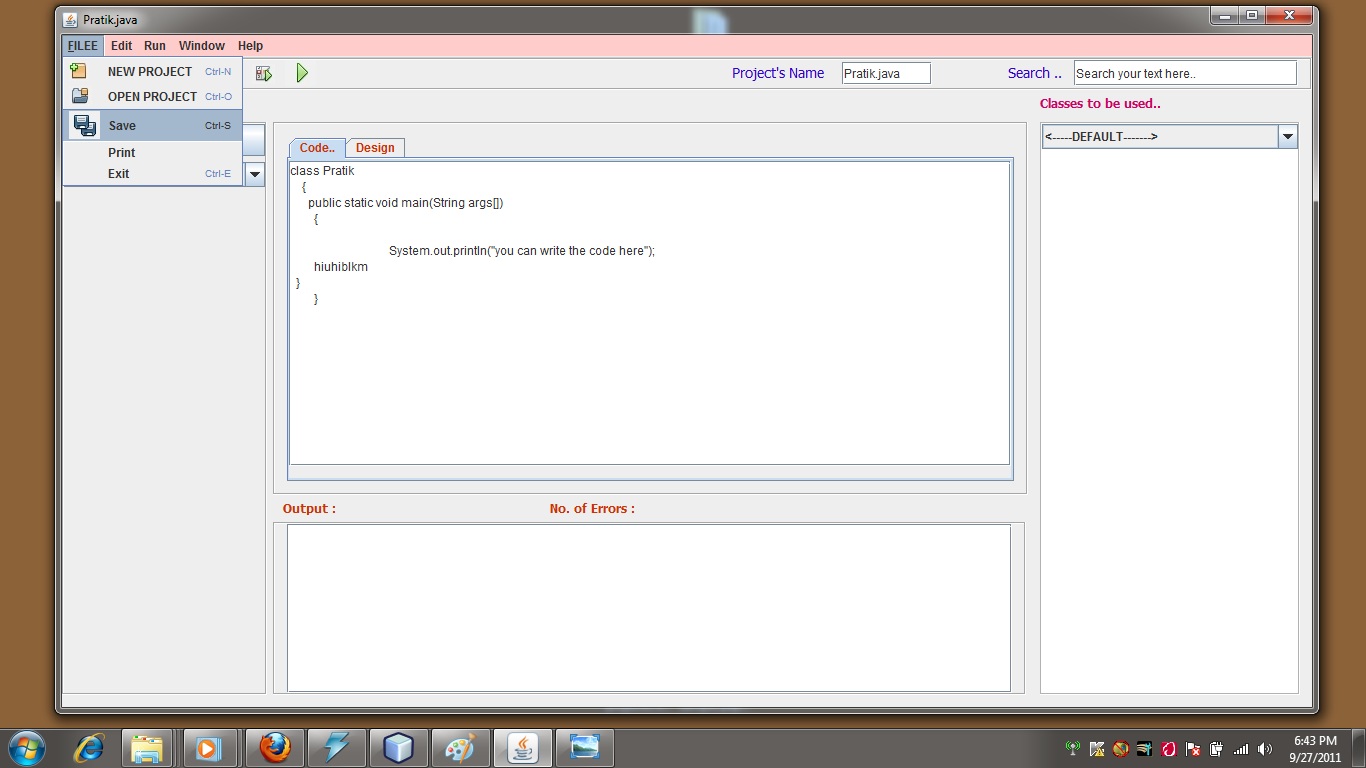
****

Fig :- 2.9 Saving project

**6.2.6 Text Manipulation**

**6.2.6.1 Undo**

It erases the last change done to the document reverting it to an older state. In some more advanced programs such as graphic processing, undo will negate the last command done to the file being edited.

The opposite of undo is **redo**. The redo command reverses the undo or advances the buffer to a more current state.

In most Windows applications, the Undo command is activated by pressing Ctrl+Z or Alt+Backspace.

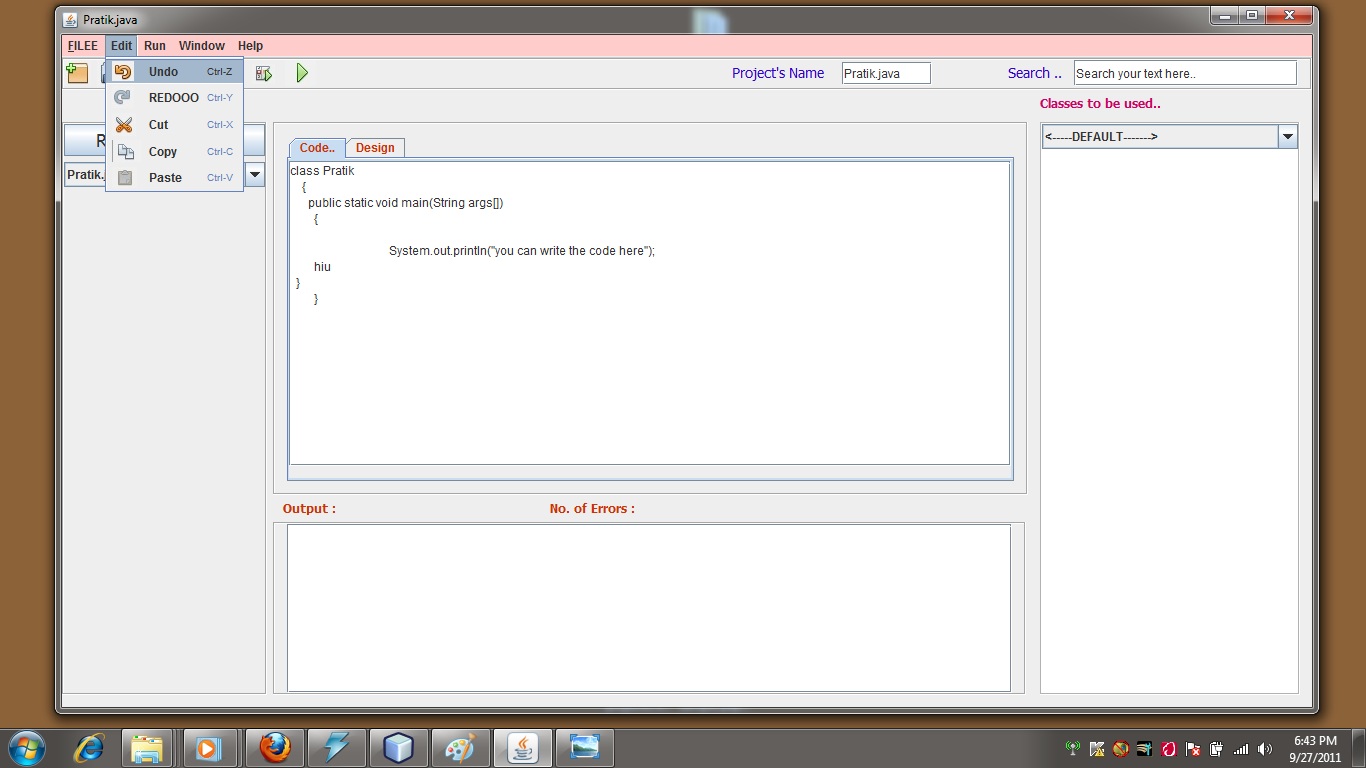


Fig :- 2.10 Undo

**6.2.6.2 Redo**

Redo reverts the effects of the undo action. The simplest form of redo is flip-undo, in which using undo after undoing redoes the undone action. In this case, the program flips back and forth between two states when the undo button is pressed.

In a more typical redo model, there are separate undo and redo buttons. The redo can be used for each undo action performed. Making a new edit usually clears the redo list. If a branching redo model is used, the new edit branches the action history.

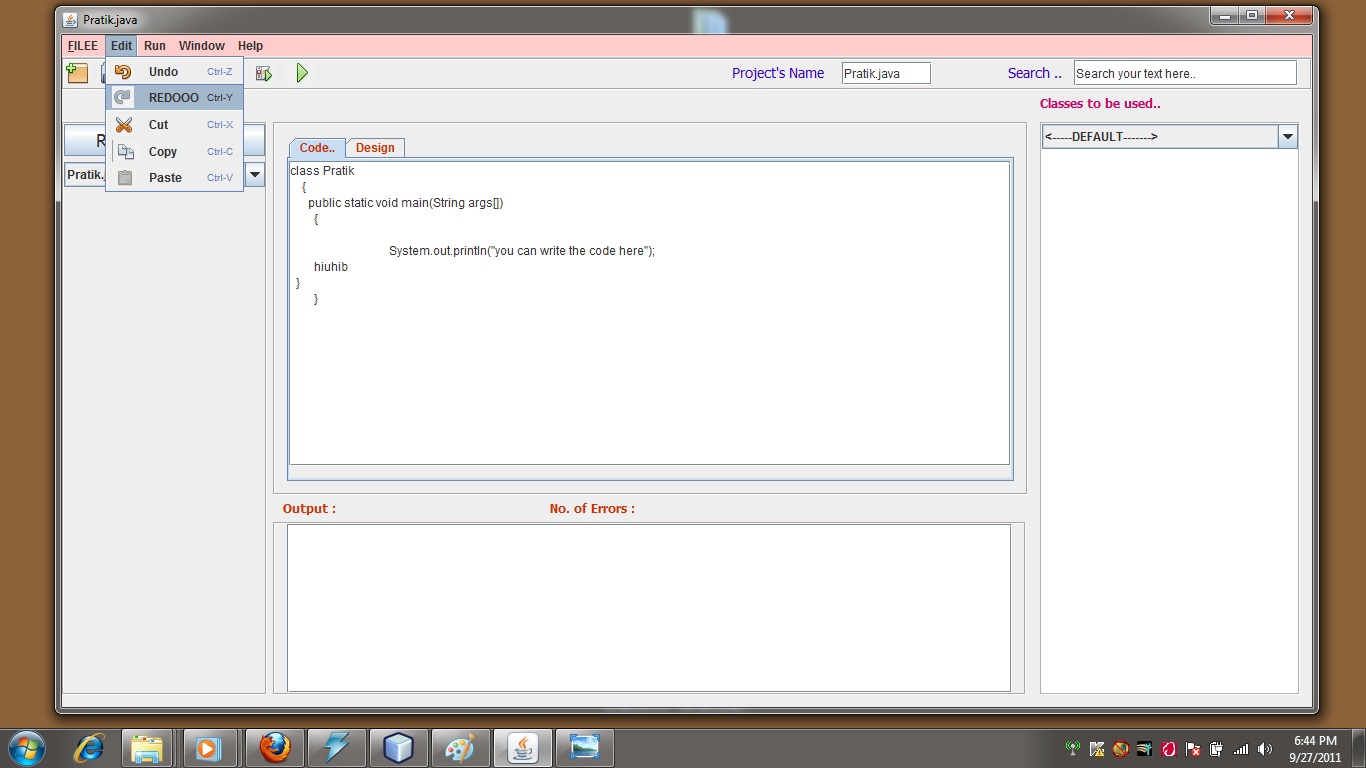
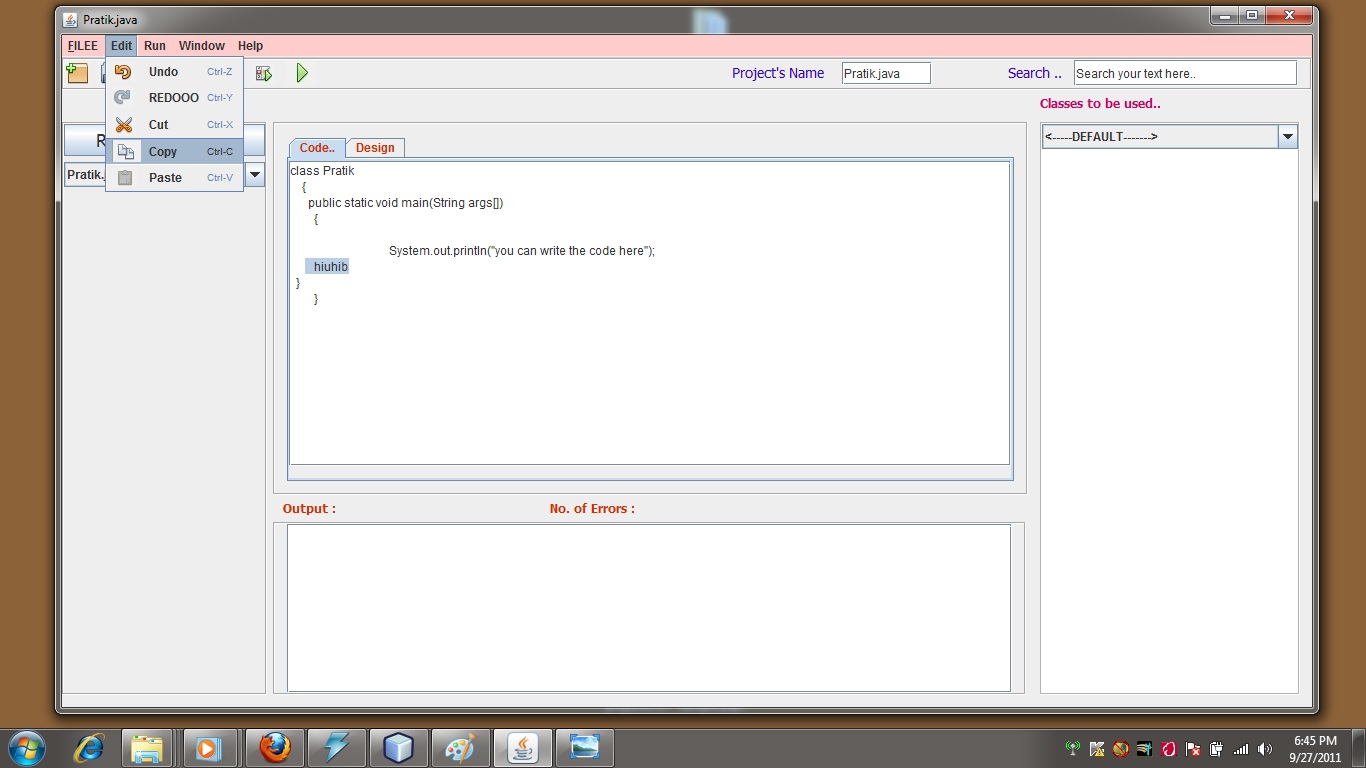


Fig :- 2.11 Redo

**6.2.6.3 Copy**



**6.2.6.4 Paste**

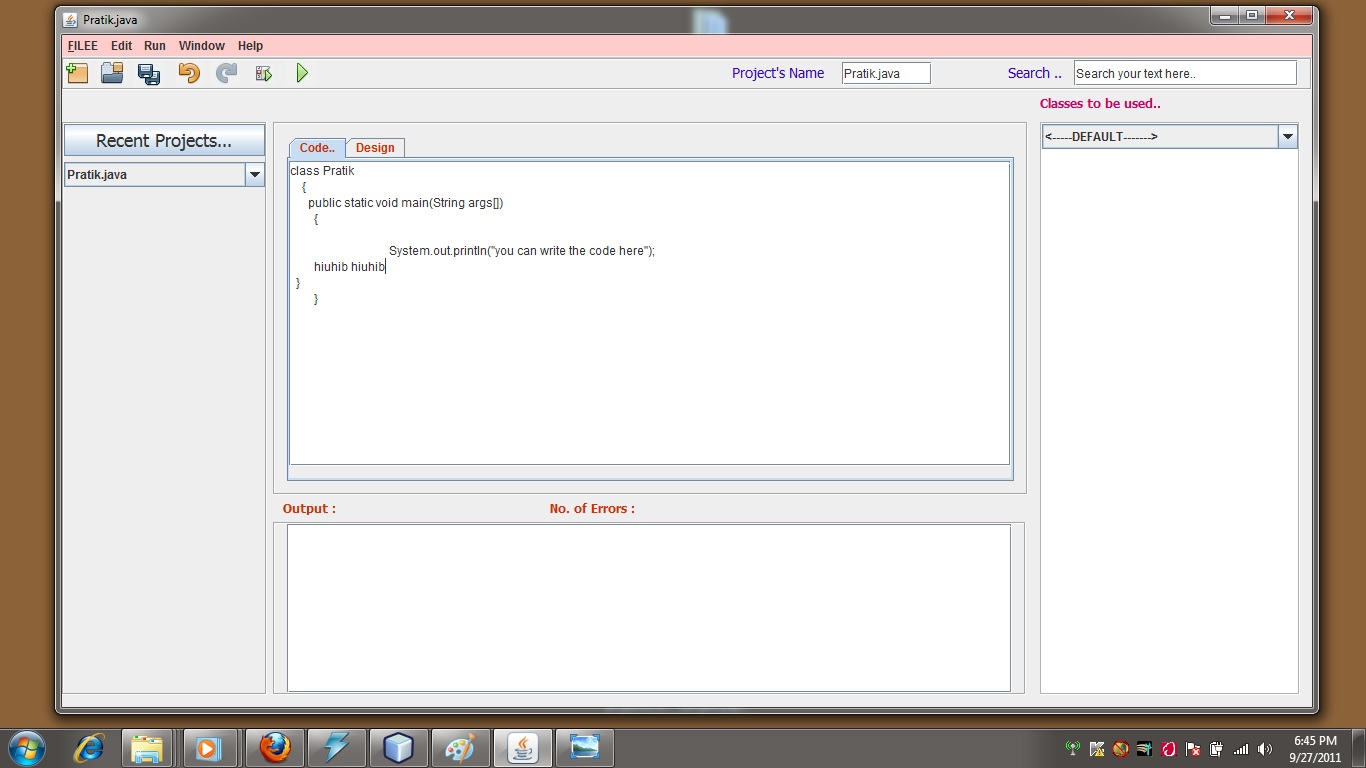
****

Fig :- 2.12 Copy and Paste

**6.2.7 Code Compilation**

In Java, programs are not compiled into executable files. They are compiled into Bytecode which the JVM then executes at runtime. Java source code is compiled into bytecode when we use the javac compiler. The bytecode gets saved on the disk with the file extension .class. When the program is to be run, the bytecode is converted, using the Just-In-Time(JIT) compiler. The result is machine code which is then fed to the memory and is executed.

So Java has four step compilation:

* compiler will check the syntactical error
* create byte-code
* create a blank \*.class file
* merge that byte code to that blank \*.class file

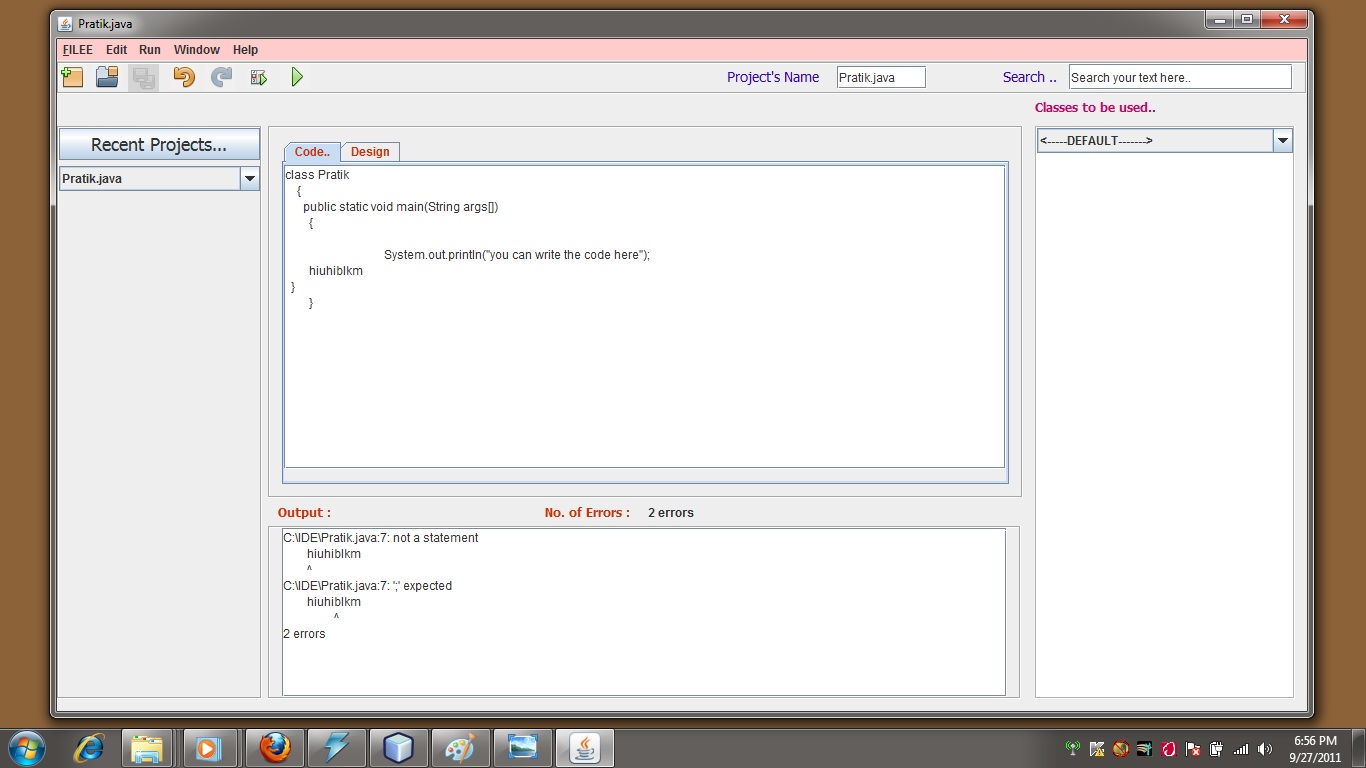
****

Fig :- 2.13 Compilation

**6.2.8 Error Removing**

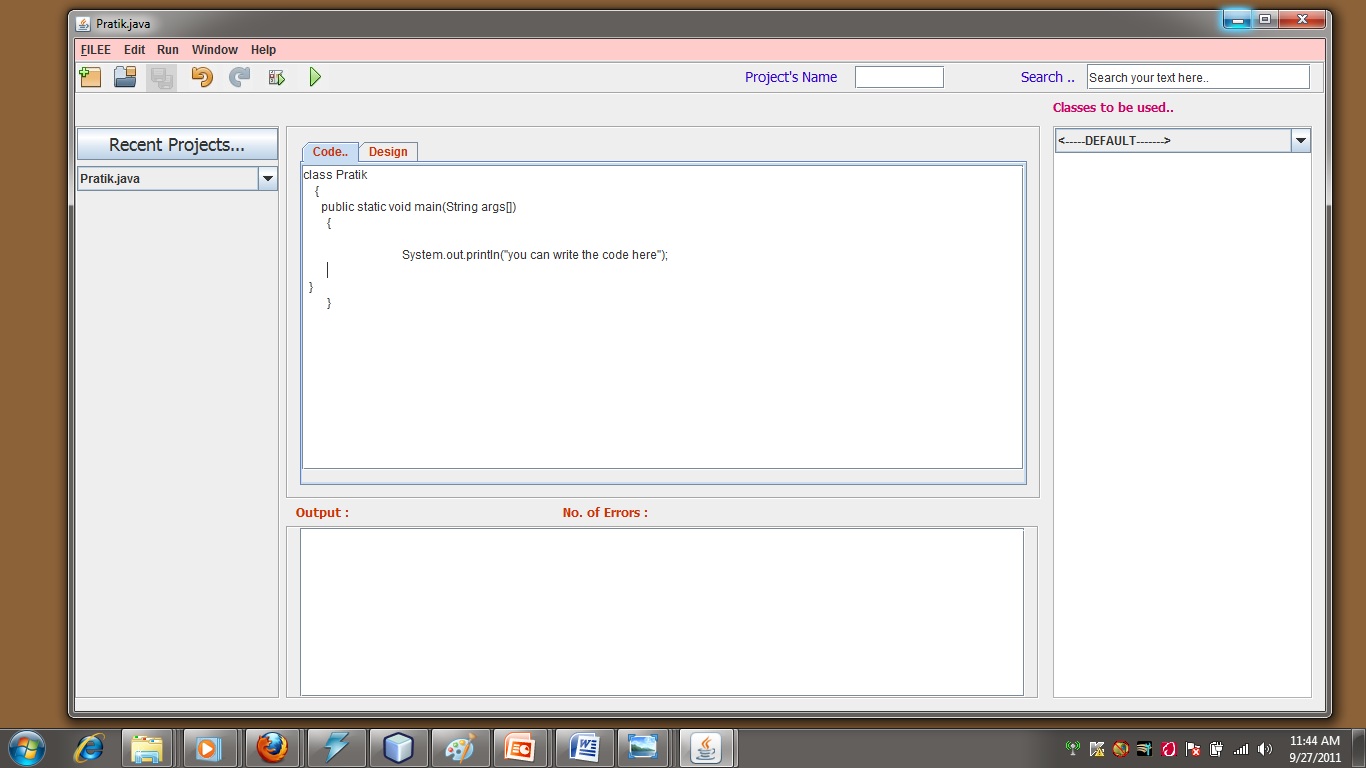
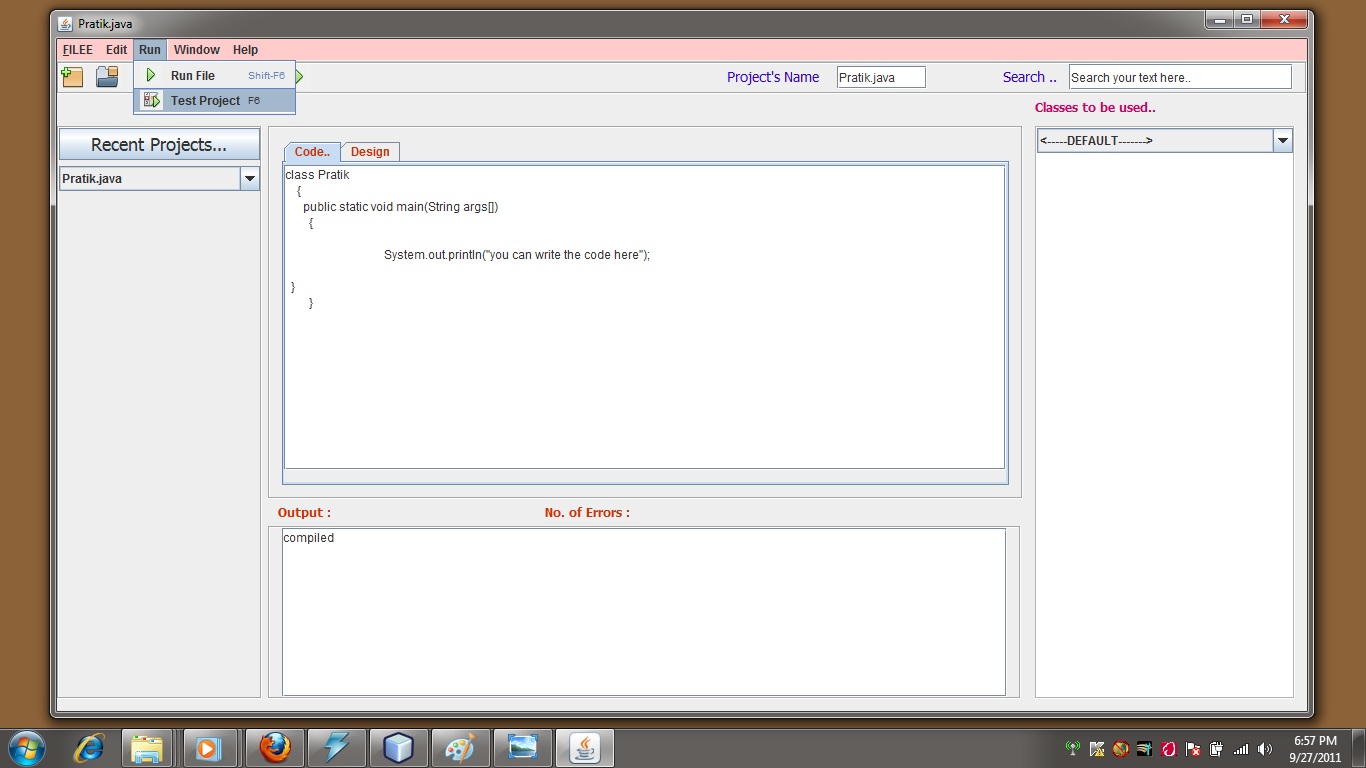
****

Fig :- 2.14 Error Removing

**6.2.9 Compiling Again**

****

**6.2.10 Generation of byte code( .class file )**

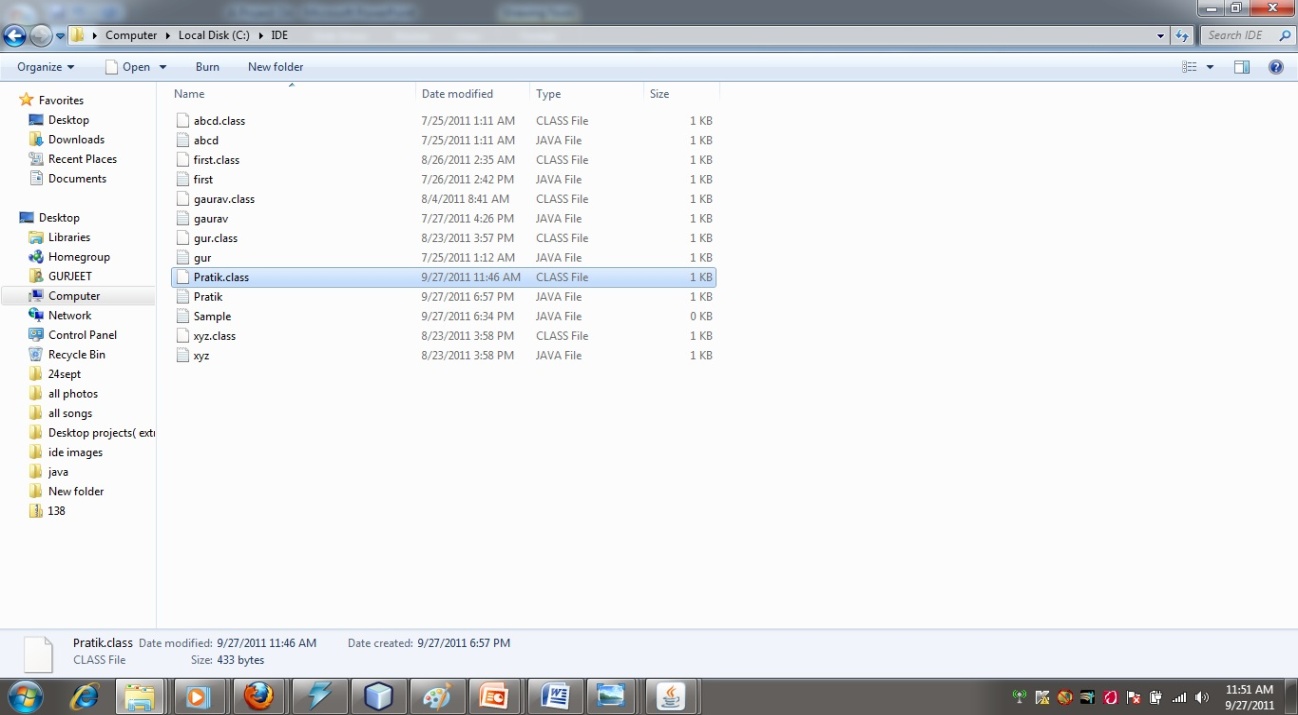
****

Fig :- 2.15 byte code generation

**6.2 Appendix A (Source code)**

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*

\* NewJFrame1.java

\*

\* Created on Jul 17, 2011, 3:35:36 AM

\*/

package ide;

import com.sun.xml.internal.ws.api.streaming.XMLStreamReaderFactory.Default;

import java.awt.event.\*;

import java.lang.reflect.Method;

import java.net.\*;

import javax.swing.JLabel;

import java.awt.\*;

import java.io.\*;

import java.sql.\*;

import javax.swing.\*;

import javax.swing.tree.\*;

/////

import java.lang.reflect.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.lang.reflect.Method;

import java.net.\*;

import javax.swing.JLabel;

import java.awt.\*;

import java.io.\*;

import java.sql.\*;

import javax.swing.\*;

import javax.swing.text.Document;

import javax.swing.tree.\*;

import javax.swing.undo.\*;

import javax.swing.undo.UndoManager;

import javax.swing.event.UndoableEditEvent;

import javax.swing.event.UndoableEditListener;

import javax.swing.text.Document;

import javax.swing.text.JTextComponent;

import javax.swing.undo.CannotRedoException;

import javax.swing.undo.CannotUndoException;

import javax.swing.undo.UndoManager;

import java.sql.\*;

import javax.swing.\*;

import java.awt.event.\*;

import java.awt.\*;

import javax.tools.\*;

import java.io.\*;

import java.awt.TextComponent.\*;

import java.util.\*;

import javax.swing.event.DocumentEvent;

import javax.swing.event.DocumentListener;

import java.awt.event.ActionEvent;

import javax.swing.text.BadLocationException;

import javax.swing.GroupLayout.\*;

/\*\*

////

/\*\*

\*

\* @author GURJEET

\*/

public class NewJFrame1 extends javax.swing.JFrame {

private void jMenuItem7MouseClicked(java.awt.event.MouseEvent evt) {

///System.exit(0);

// System.out.println("hhhhh");

}

private void jMenuItem7MousePressed(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

System.exit(0);

System.out.println("hhhhh");

}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

Connection con = null;

Statement st;

ResultSet rs;

try

{

Class.forName("com.mysql.jdbc.Driver");

con = DriverManager.getConnection("jdbc:mysql://localhost:3306/IDE","root","123456");

st = con.createStatement();

rs = st.executeQuery("Select filename from newfile");

while(rs.next())

{

jComboBox2.addItem(rs.getString("filename"));

}

//else

// jOptionPane1.showMessageDialog(this, "Wrong details entered.", "Error", javax.swing.JOptionPane.ERROR\_MESSAGE);

rs.close();

con.close();

}

catch(Exception e)

{

System.out.println(e.getMessage());

}

;

}

String commands[]= {"C:\\Program Files\\Java\\jdk1.6.0\_15\\bin\\java","-classpath",d,classname[1]};

try

{

Process ps=Runtime.getRuntime().exec(commands);

Scanner sc=new Scanner(ps.getErrorStream());

boolean flag=true;

while(sc.hasNextLine())

{

jLabel9.setText(sc.nextLine());

// jTextArea1.setText(sc.nextLine());

//System.out.println(sc.nextLine());

flag=false;

}

if(flag)

{

//jTextArea1.setText("done!!!!");

System.out.println("Done !!");

}

/////

InputStream stderr = ps.getInputStream();

InputStreamReader isr = new InputStreamReader(stderr);

BufferedReader br = new BufferedReader(isr);

String line = null;

while ( (line = br.readLine()) != null)

{

jTextArea2.append(line+"\n");

System.out.println(line);

//System.out.println("");

}

sc.close();

}

catch(Exception e)

{

System.out.println(e.getMessage());

}

}

// jTextArea1.setText("you have not compiled or error in compilation");\*/

}

private void jMenuItem19MousePressed(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

//String commands[]= {"C:\\Program Files\\Java\\jdk1.6.0\_15\\bin\\javac","-classpath","C:\\","C:\\IDE\\hello.java"};

jTextArea2.setText("");

jTextField1.setText(d);

jTextField1.setText(f);

String s=jTextArea1.getText();

String classname[] = new String[1000];

StringTokenizer st = new StringTokenizer(s);

for (int i = 0; st.hasMoreTokens(); i++)

{

classname[i] = st.nextToken();

}

classname[1]= classname[1]+".java";

// String f = getTitle()+".java";

// String commands[]= {"C:\\Program Files\\Java\\jdk1.6.0\_15\\bin\\javac","-classpath","C:\\","C:\\IDE\\"+classname[1]};

String commands[]= {"C:\\Program Files\\Java\\jdk1.6.0\_15\\bin\\javac","-classpath","C:\\",d+f};

// String commands[]= {"C:\\Program Files\\Java\\jdk1.6.0\_15\\bin\\javac","-classpath",d,f};

try

{

Process ps=Runtime.getRuntime().exec(commands);

Scanner sc=new Scanner(ps.getErrorStream());

boolean flag=true;

//String s="";

while(sc.hasNextLine())

{

// s=sc.nextLine();

jTextArea2.append(sc.nextLine());

/\* jLabel9.setText(sc.nextLine());

jTextArea2.setText(sc.nextLine());

jTextArea2.setText(sc.nextLine());

jTextArea2.setText(sc.nextLine());

jTextArea2.setText(sc.nextLine());\*/

//System.out.println(sc.nextLine());

flag=false;

}

if(flag)

{ jLabel9.setVisible(true);

jLabel9.setText("0");

jTextArea2.setText("compiled");

System.out.println("compiled");

}

sc.close();

}

catch(Exception e)

{

System.out.println(e.getMessage());

}

}

private void jMenuItem16MousePressed1(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

{

String s=jTextArea1.getText();

String classname[] = new String[1000];

StringTokenizer st = new StringTokenizer(s);

for (int i = 0; st.hasMoreTokens(); i++)

{

classname[i] = st.nextToken();

}

String commands[]= {"C:\\Program Files\\Java\\jdk1.6.0\_15\\bin\\java","-classpath",d,classname[1]};

try

{

Process ps=Runtime.getRuntime().exec(commands);

Scanner sc=new Scanner(ps.getErrorStream());

boolean flag=true;

while(sc.hasNextLine())

{

jLabel9.setText(sc.nextLine());

// jTextArea1.setText(sc.nextLine());

//System.out.println(sc.nextLine());

flag=false;

}

if(flag)

{

//jTextArea1.setText("done!!!!");

System.out.println("Done !!");

}

/////

InputStream stderr = ps.getInputStream();

InputStreamReader isr = new InputStreamReader(stderr);

BufferedReader br = new BufferedReader(isr);

String line = null;

while ( (line = br.readLine()) != null)

{

jTextArea2.append(line);

}

sc.close();

}

catch(Exception e)

{

System.out.println(e.getMessage());

}

}

}

private void jMenuItem33KeyPressed(java.awt.event.KeyEvent evt) {

// TODO add your handling code here:

// jInternalFrame1.setVisible(true);

}

// System.out.println(M[i].toString());

jComboBox1.addItem(M[i].toString());

System.out.println(M[i].toString());

//jTextArea3.append(M[i].toString()+"\n ");

}

} catch(Exception e) {

System.out.println(e.getMessage());

}

}

}

catch(Exception e)

{

System.out.println(e.getMessage());

}\*/

}

private void jLabel2MousePressed(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

jTextArea1.setEnabled(true);

new NewProject().setVisible(true);

jTabbedPane2.setEnabled(true);

}

private void jLabel3MousePressed(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

jTextArea1.setEnabled(true);

FileDialog fd = new FileDialog(this,"OPEN",FileDialog.LOAD);

fd.setVisible(true);

String dir= fd.getDirectory();

//

d= fd.getDirectory();

//

String file= fd.getFile();

//

f= fd.getFile();

//

String s = dir+file;

setTitle(file);

jTextField2.setText(d);

jTextField1.setText(f);

try{

FileInputStream fin = new FileInputStream(s);

int ch;

String s11="";

FileInputStream fstream = new FileInputStream(s);

while ((ch = fin.read()) != -1)

s11=s11+(char)ch;

fin.close();

String j = s11;

jTextArea1.setText(j);

// jTextField1.setText(j);

}

catch(Exception e)

{

System.out.println(e);

}

}

private void jLabel4MousePressed(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

String a = getTitle();

String result="";

// jTextField3.setText(a);

/\* Connection con = null;

Statement st;

ResultSet rs;

try

{

Class.forName("com.mysql.jdbc.Driver");

con = DriverManager.getConnection("jdbc:mysql://localhost:3306/IDE","root","123456");

st=con.createStatement();

rs=st.executeQuery("select \* from newfile where filename=a");

while(rs.next())

{

String direc = rs.getString("dir");

result = direc+a;

}

con.close();

jTextField3.setText(result);

}

catch(Exception e)

{

}\*/

/////////////

try{

String filname = d+f;

FileWriter fr = new FileWriter(filname);

BufferedWriter br = new BufferedWriter(fr);

br.write(jTextArea1.getText());

br.close();

}

catch(Exception e)

{

System.out.println(e);

}

}

private void jLabel13MousePressed(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

// TODO add your handling code here:

//String commands[]= {"C:\\Program Files\\Java\\jdk1.6.0\_15\\bin\\javac","-classpath","C:\\","C:\\IDE\\hello.java"};

jTextField1.setText(d);

jTextField1.setText(f);

String s=jTextArea1.getText();

String classname[] = new String[1000];

StringTokenizer st = new StringTokenizer(s);

for (int i = 0; st.hasMoreTokens(); i++)

{

classname[i] = st.nextToken();

}

classname[1]= classname[1]+".java";

// String f = getTitle()+".java";

// String commands[]= {"C:\\Program Files\\Java\\jdk1.6.0\_15\\bin\\javac","-classpath","C:\\","C:\\IDE\\"+classname[1]};

String commands[]= {"C:\\Program Files\\Java\\jdk1.6.0\_15\\bin\\javac","-classpath","C:\\",d+f};

// String commands[]= {"C:\\Program Files\\Java\\jdk1.6.0\_15\\bin\\javac","-classpath",d,f};

try

{

Process ps=Runtime.getRuntime().exec(commands);

Scanner sc=new Scanner(ps.getErrorStream());

boolean flag=true;

//String s="";

while(sc.hasNextLine())

{

// s=sc.nextLine();

jTextArea2.setText(sc.nextLine());

jLabel9.setText(sc.nextLine());

jTextArea2.setText(sc.nextLine());

jTextArea2.setText(sc.nextLine());

jTextArea2.setText(sc.nextLine());

jTextArea2.setText(sc.nextLine());

//System.out.println(sc.nextLine());

flag=false;

}

if(flag)

{ jLabel9.setVisible(true);

jLabel9.setText("0");

jTextArea2.setText("compiled");

System.out.println("compiled");

}

sc.close();

}

catch(Exception e)

{

System.out.println(e.getMessage());

}

}

private void jLabel7MousePressed(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

// TODO add your handling code here:

//String commands[]= {"C:\\Program Files\\Java\\jdk1.6.0\_15\\bin\\java","-classpath","C:\\","Array2"};

jTextArea2.setText("");

{

String s=jTextArea1.getText();

String classname[] = new String[1000];

StringTokenizer st = new StringTokenizer(s);

for (int i = 0; st.hasMoreTokens(); i++)

{

classname[i] = st.nextToken();

}

String commands[]= {"C:\\Program Files\\Java\\jdk1.6.0\_15\\bin\\java","-classpath",d,classname[1]};

try

{

Process ps=Runtime.getRuntime().exec(commands);

Scanner sc=new Scanner(ps.getErrorStream());

boolean flag=true;

while(sc.hasNextLine())

{

jLabel9.setText(sc.nextLine());

// jTextArea1.setText(sc.nextLine());

//System.out.println(sc.nextLine());

flag=false;

}

if(flag)

{

//jTextArea1.setText("done!!!!");

System.out.println("Done !!");

}

/////

InputStream stderr = ps.getInputStream();

InputStreamReader isr = new InputStreamReader(stderr);

BufferedReader br = new BufferedReader(isr);

String line = null;

while ( (line = br.readLine()) != null)

{

jTextArea2.append(line+"\n");

System.out.println(line);

jLabel9.setText((""));

//System.out.println("");

}

sc.close();

}

catch(Exception e)

{

System.out.println(e.getMessage());

}

}

// jTextArea1.setText("you have not compiled or error in compilation");

}

private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {

jTextArea1.setEnabled(true);

FileDialog fd = new FileDialog(this,"OPEN",FileDialog.LOAD);

fd.setVisible(true);

String dir= fd.getDirectory();

//

d= fd.getDirectory();

//

String file= fd.getFile();

//

f= fd.getFile();

//

String s = dir+file;

setTitle(file);

jTextField2.setText(d);

jTextField1.setText(f);

try{

FileInputStream fin = new FileInputStream(s);

int ch;

String s11="";

FileInputStream fstream = new FileInputStream(s);

while ((ch = fin.read()) != -1)

s11=s11+(char)ch;

fin.close();

String j = s11;

jTextArea1.setText(j);

// jTextField1.setText(j);

}

catch(Exception e)

{

System.out.println(e);

} // TODO add your handling code here:

}

private void jMenu1ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

jTextArea1.setEnabled(true);

new NewProject().setVisible(true);

jTabbedPane2.setEnabled(true);

}

private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

jTextArea1.setEnabled(true);

new NewProject().setVisible(true);

jTabbedPane2.setEnabled(true);

}

private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

String a = getTitle();

String result="";

// jTextField3.setText(a);

/\* Connection con = null;

Statement st;

ResultSet rs;

try

{

Class.forName("com.mysql.jdbc.Driver");

con = DriverManager.getConnection("jdbc:mysql://localhost:3306/IDE","root","123456");

st=con.createStatement();

rs=st.executeQuery("select \* from newfile where filename=a");

while(rs.next())

{

String direc = rs.getString("dir");

result = direc+a;

}

con.close();

jTextField3.setText(result);

}

catch(Exception e)

{

}\*/

/////////////

try{

String filname = d+f;

FileWriter fr = new FileWriter(filname);

BufferedWriter br = new BufferedWriter(fr);

br.write(jTextArea1.getText());

br.close();

}

catch(Exception e)

{

System.out.println(e);

}

////////////////////

}

private void jMenuItem7ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

System.exit(0);

}

private void jMenuItem11ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

/\* cuttext=jTextArea1.getSelectedText();

int cutindex=jTextArea1.getText().indexOf(cuttext);

jTextArea1.replaceRange("",cutindex,cutindex+cuttext.length());\*/

}

private void jMenuItem8ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

/\* getcaretpos=jTextArea1.getCaretPosition();

jTextArea1.insert(cuttext, getcaretpos);\*/

}

private void jComboBox2ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:C:\\IDE\\

}

private void jComboBox2ItemStateChanged(java.awt.event.ItemEvent evt) {

// TODO add your handling code here:

String s = (String)evt.getItem();

try{

FileInputStream fin = new FileInputStream("C://IDE//"+s);

int ch;

String s11="";

while ((ch = fin.read()) != -1)

{

s11 = s11 + (char) ch;}

fin.close();

String j = s11;

jTextArea1.setText(j);

// jTextField1.setText(j);

}

catch(Exception e)

{

System.out.println(e);

}

d = "C://IDE//";

f = s;

setTitle(s);

}

private void jMenuItem12MousePressed1(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

cuttext = jTextArea1.getSelectedText();

}

private void jMenuItem16ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

//String commands[]= {"C:\\Program Files\\Java\\jdk1.6.0\_15\\bin\\java","-classpath","C:\\","Array2"};

jTextArea2.setText("");

{

String s=jTextArea1.getText();

String classname[] = new String[1000];

StringTokenizer st = new StringTokenizer(s);

for (int i = 0; st.hasMoreTokens(); i++)

{

classname[i] = st.nextToken();

}

String commands[]= {"C:\\Program Files\\Java\\jdk1.6.0\_15\\bin\\java","-classpath",d,classname[1]};

try

{

Process ps=Runtime.getRuntime().exec(commands);

Scanner sc=new Scanner(ps.getErrorStream());

boolean flag=true;

while(sc.hasNextLine())

{

jLabel9.setVisible(true);

jLabel9.setText(sc.nextLine());

// jTextArea1.setText(sc.nextLine());

//System.out.println(sc.nextLine());

flag=false;

}

if(flag)

{

jLabel1.setText("done!!!!");

System.out.println("Done !!");

}

/////

InputStream stderr = ps.getInputStream();

InputStreamReader isr = new InputStreamReader(stderr);

BufferedReader br = new BufferedReader(isr);

String line = null;

while ( (line = br.readLine()) != null)

{

jTextArea2.append(line+"\n");

System.out.println(line);

jLabel9.setText((""));

//System.out.println("");

}

sc.close();

}

catch(Exception e)

{

System.out.println(e.getMessage());

}

}

// jTextArea1.setText("you have not compiled or error in compilation");

}

private void jMenuItem19MousePressed1(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

}

private void jMenuItem27ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

Connection con = null;

Statement st;

ResultSet rs;

try

{

Class.forName("com.mysql.jdbc.Driver");

con = DriverManager.getConnection("jdbc:mysql://localhost:3306/IDE","root","123456");

st = con.createStatement();

rs = st.executeQuery("Select filename from newfile");

while(rs.next())

{

jComboBox2.addItem(rs.getString("filename"));

}

//else

// jOptionPane1.showMessageDialog(this, "Wrong details entered.", "Error", javax.swing.JOptionPane.ERROR\_MESSAGE);

rs.close();

con.close();

}

catch(Exception e)

{

System.out.println(e.getMessage());

}

}

private void jMenuItem19ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

jTextArea2.setText("");

String commands[]= {"C:\\Program Files\\Java\\jdk1.6.0\_15\\bin\\javac","-classpath","C:\\",d+f};

//String commands[]= {"C:\\Program Files\\Java\\jdk1.6.0\_15\\bin\\javac","-classpath","C:\\","C:\\IDE\\hello.java"};

try

{

Process ps=Runtime.getRuntime().exec(commands);

Scanner sc=new Scanner(ps.getErrorStream());

boolean flag=true;

long count=0;

while(sc.hasNextLine())

{

jTextArea2.append(sc.nextLine()+"\n");

count = count+1;

//value = value + sc.nextLine();

flag=false;

}

//jLabel9.setText(value);

if(flag)

{

jTextArea2.setText("");

jTextArea2.append("compiled");

jLabel9.setText("");

}

String err ="";

if(jTextArea2.getText()!="compiled")

{

String var = jTextArea2.getText();

err = var.substring(var.length()-9,var.length());

}

jLabel9.setVisible(true);

jLabel9.setText(err);

/\*String co = String.valueOf(count);

jLabel9.setVisible(true);

jLabel9.setText(co);\*/

sc.close();

}

catch(Exception e)

{

System.out.println(e.getMessage());

}

}

private void jMenuItem17ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String args[]) {

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new NewJFrame1().setVisible(true);

}

});

}