**MENU RECOMMENDER**



An Object-Oriented Programming through Java Course Project Report

in partial fulfillment of the degree

**Bachelor of Technology**

in

**Computer Science & Engineering**

**By**

RAYIKANTI SRIJA 2003A51028

MOHAMMAD TAJUDDIN 2003A51026

**Submitted to**





**DEPARTMENT OFCOMPUTERSCIENCE& ENGINEERING**

**CERTIFICATE**

This is to certify that the **Object Oriented Programming through Java - Course Project** Report entitled **“MENU RECOMMENDER** “is a record of bonafide work carried out by the student 2003A51026 MOHAMMAD TAJUDDIN, 2003A51028 RAYIKANTI SRIJA during the academic year 2022-2023 in partial fulfillment of the award of the degree of ***Bachelor of Technology*** in **Computer Science & Engineering** by the SR University, Hasanparthy.

**Lab In-charge Head of the Department**

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **TITLE** | **Page no** |
| Abstract | 1 |
| Objective | 1 |
| Definition of elements used in project | 2 |
| Design | 9 |
| -Screens | 9 |
| Implementation | 12 |
| -Code | 12 |
| Result Screens | 22 |
| Conclusions | 27 |

**ABSTRACT :**

* People who love to choose their favourite dish need to go through the menu. We have created a GUI to open an interface which has multiple cuisines so that we will be able to choose our favourite dish from different country different types of dishes.
* Many factors influence an individual’s health, such as physical exercise, sleep, nutrition, heredity and pollution.
* Being nutrition one of the biggest modifiable factors in our lives, small changes can have a big impact.
* With the exponential increase in the number of available food options, it is not possible to take them into account anymore.
* The only way to consider user preferences, maximize the number of healthy compounds and minimize the unhealthy ones in food, is using recommendation systems

**OBJECTIVE:**

**-**To develop a system that will surely satisfy the customer needs.

-To design a system able to accommodate huge amount of orders at a time.  
-To evaluate its performance and acceptability in terms of security, user-friendliness, accuracy and reliability.  
-To improve the communication between the client and the server and minimize the time of ordering.

-This will take less time to order the food.

**DEFINITIONS OF THE ELEMENTS USED:**

* Event Queue:

Event Queue is a platform-independent class that queues events, both from the underlying

peer classes and from trusted application classes. There is only one Event Queue for the

system.

It encapsulates asynchronous event dispatch machinery which extracts events from the

queue and dispatches them to the appropriate client program. The particular behavior of

this machinery is implementation-dependent. The only requirements are that events which

were actually en-queued to this queue (note that events being posted to the Event Queue

can be collapsed are dispatched:

Sequentially,

That is, it is not permitted that several events from this queue are dispatched

simultaneously. In the same order as they are en-queued.

That is, if AWT Event A is en-queued to the Event Queue before AWT Event B then

event B will not be dispatched before event A.

* Action Event:

This class inherits methods from the following classes:

java.awt.AWTEvent

java.util.EventObject

java.lang.Object

An action event occurs, whenever an action is performed by the user. Examples: When

the user clicks a button, chooses a menu item, presses Enter in a text field. The result is

that an action Performed message is sent to all action listeners that are registered on the

relevant component.

* Toolkit

Toolkit class is the abstract superclass of every implementation in the Abstract Window Toolkit. Subclasses of Toolkit are used to bind various components. It inherits Object class.

* Action Listener

The Java Action Listener is notified whenever you click on the button or menu item. It

is notified against Action Event. The Action Listener interface is found in

java.awt.event [package](https://www.javatpoint.com/package). It has only one method: action Performed().

The class which processes the Action Event should implement this interface. The object

of that class must be registered with a component. The object can be registered using

the **addActionListener()** method. When the action event occurs, that object's

action Performed method is invoked.

Interface Declaration

Following is the declaration for **java.awt.event.ActionListener** interface −

public interface ActionListener extends EventListener.

**.** **Java files handling**:

* From Java 1.5 Scanner class was introduced. This class accepts a File, InputStream, Path and, String objects, reads all the primitive data types and Strings (from the given source) token by token using regular expressions. By default, whitespace is considered as the delimiter (to break the data into tokens).
* File Handling is an integral part of any programming language as file handling enables us to store the output of any particular program in a file and allows us to perform certain operations on it.
* In simple words, file handling means reading and writing data to a file.

**JFrame**

JFrame is a Swing’s top-level container that renders a window on screen. A frame is a

base window on which other components rely, such as menu bar, panels, labels, text

fields, buttons, etc. Almost every Swing application starts with JFrame window. This

article provides a summary of common practices when using JFrame in Swing

development.

When we create a JFrame object, by default, it is not visible. We have to call

its setVisible(boolean visible) method to make it visible. true parameter makes the frame

visible, to hide the frame pass false to the set Visible method.

* **JLabel**

The object of JLabel class is a component for placing text in a container. It is used to

display a single line of read only text. The text can be changed by an application but a

user cannot edit it directly. It inherits JComponent class.

JLabel class declaration

**public** **class** JLabel **extends** JComponent **implements** SwingConstants, Accessible

JLabel is **a class of java Swing** . JLabel is used to display a short string or an image

icon. JLabel can display text, image or both. JLabel is only a display of text or image

and it cannot get focus . JLabel is inactive to input events such a mouse focus or

keyboard focus.

* JButton

The JButton class is used to create a labeled button that has platform independent

implementation. The application result in some action when the button is pushed. It

inherits AbstractButton class.

The class **JButton** is an implementation of a push button. This component has a label

and generates an event when pressed. It can also have an Image.

The JButton is the class that is used to create a button in a JavaSE application.Use

to send actions to your application whenever a user interacts with the button by either

clicking, hovering, etc

* JTextField

The object of a JTextField class is a text component that allows the editing of a single line

text. It inherits JTextComponent class.

JTextField are a very useful part of your Swing Toolkit. Simply put, the TextField lets the

user enter text into your program without using a command line prompt.

JTextField is also a feature of Java Swing.which allows the single line editing of the text. The

main intention is to keep its stability and compatibility intact with java.awt.TextField and

forms an integral part of the javax.swing package. It inherits the JTextComponent class and

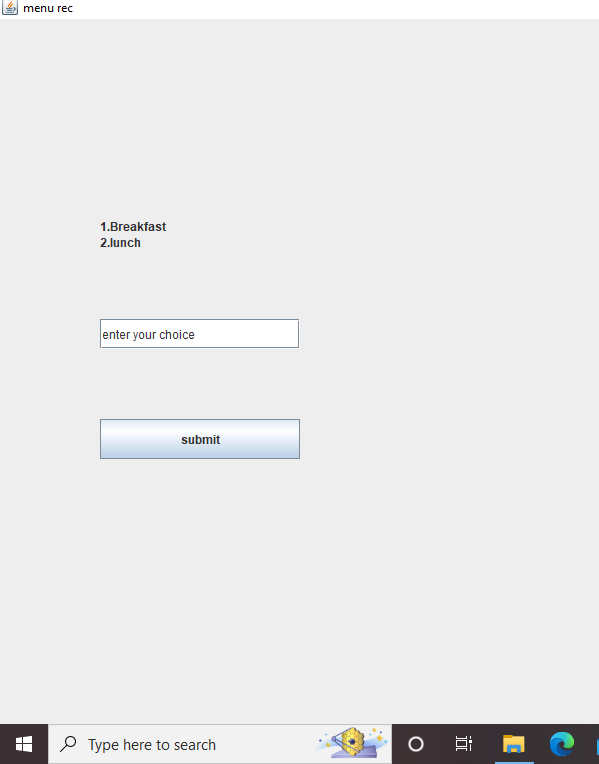
makes use of the SwingConstant interface.

**DESIGN**

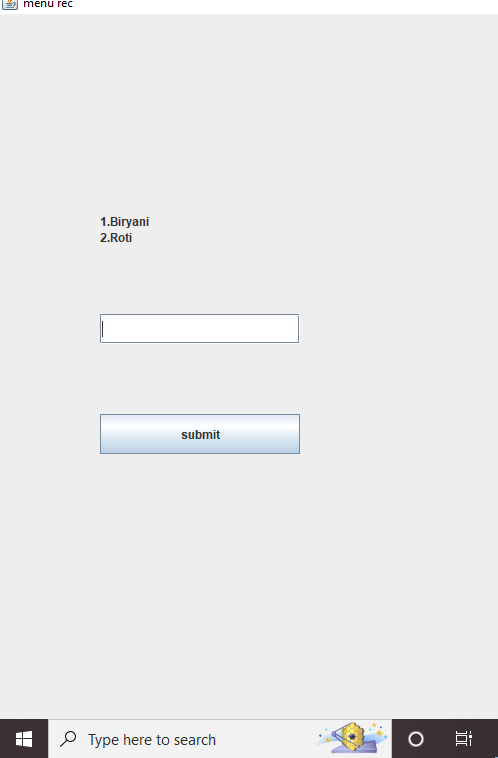
**Screen-1 :**

****

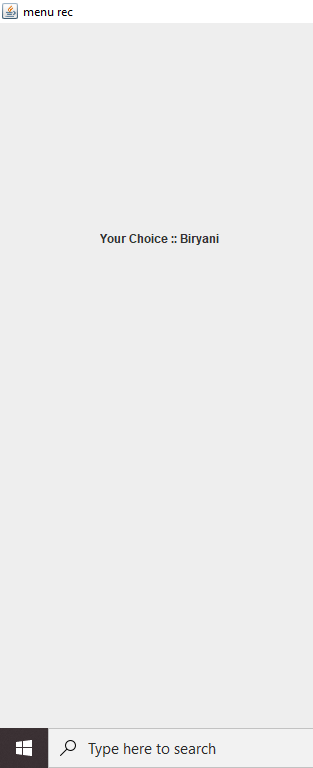
**Screen-2 :**

****

**Screen-3 :**

****

**Screen-4 :**

****

**IMPLEMENTATION :**

CODE :

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

import java.io.\*;

import java.nio.file.Files;

import java.nio.file.Path;

import java.util.\*;

public class menu extends JFrame{

//Integer c;

public static String Readdata(String filepath)throws IOException{

String str="";

Scanner sc=new Scanner(new File(filepath)); //first txt document menu (1.indian,2,chinese,etc)

while(sc.hasNextLine())

str=str+sc.nextLine()+"\n";

// System.out.println(str);

return str;

}

public static void secondframe(JFrame f,String s,int c1){

f.getContentPane().removeAll();

f.repaint();

JLabel l=new JLabel(s);

// l.setSize(100,200);

l.setBounds(100,200,200,30);

f.add(l);

if(c1!=-1){

JTextField t=new JTextField("enter your choice");

t.setBounds(100,300,200,30);

f.add(t);

//t.setSize(200,400);

JButton b=new JButton("submit");

b.setBounds(100,400,200,40);

b.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent e){

String s1=t.getText();

int c2=Integer.parseInt(s1);

System.out.println(c2);

if(c1==1 && c2==1 ){

thirdframe(f,"<html> 1.Idli <br/> 2.Dosa </html>",c1,c2);

}

else if(c1==1 && c2==2 ){

thirdframe(f,"<html> 1.Biryani <br/> 2.Roti </html>",c1,c2);

}

else if(c1==2 && c2==1){

thirdframe(f,"<html> 1.Eggs in salsa <br/> 2.Eggs with chorizo</html>",c1,c2);

}

else if(c1==2 && c2==2){

thirdframe(f,"<html> 1.Guacamole <br/> 2.Tacos</html>",c1,c2);

}

else if(c1==3 && c2==1){

thirdframe(f,"<html> 1.Steamed Buns <br/> 2.Rice Porridge</html>",c1,c2);

}

else if(c1==3 && c2==2){

thirdframe(f,"<html> 1.HotPot <br/> 2.Wonton</html>",c1,c2);

}

else if(c1==4 && c2==1){

thirdframe(f,"<html> 1.kimchijigae <br/> 2.manduguk</html>",c1,c2);

}

else if(c1==4 && c2==2){

thirdframe(f,"<html> 1.TTeokbokki <br/> 2.Jajangmyeon</html>",c1,c2);

}

else if(c1==5 && c2==1){

thirdframe(f,"<html> 1.Cereal <br/> 2.Pancakes</html>",c1,c2);

}

else if(c1==5 && c2==2){

thirdframe(f,"<html> 1.Burger <br/> 2.Hotdog</html>",c1,c2);

}

}});

f.add(b);

}

}

public static void thirdframe(JFrame f,String s,int c1,int c2){

f.getContentPane().removeAll();

f.repaint();

JLabel l=new JLabel(s);

// l.setSize(100,200);

l.setBounds(100,200,200,30);

f.add(l);

JTextField t=new JTextField("enter your choice");

t.setBounds(100,300,200,30);

f.add(t);

JButton b=new JButton("submit");

b.setBounds(100,400,200,40);

b.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent e){

String s1=t.getText();

int c3=Integer.parseInt(s1);

System.out.println(c3);

if(c1==1&&c2==1&&c3==1){

fourthframe(f,"Your Choice :: Idli");

}

if(c1==1&&c2==1&&c3==2){

fourthframe(f,"Your Choice :: Dosa");

}

if(c1==1&&c2==2&&c3==1){

fourthframe(f,"Your Choice :: Biryani");

}

if(c1==1&&c2==2&&c3==2){

fourthframe(f,"Your Choice :: Roti");

}

if(c1==2&&c2==1&&c3==1){

fourthframe(f,"Your Choice :: Eggs in salsa");

}

if(c1==2&&c2==1&&c3==2){

fourthframe(f,"Your Choice :: Eggs with chorizo");

}

if(c1==2&&c2==2&&c3==1){

fourthframe(f,"Your Choice :: Guacamole");

}

if(c1==2&&c2==2&&c3==2){

fourthframe(f,"Your Choice :: Tacos");

}

if(c1==3&&c2==1&&c3==1){

fourthframe(f,"Your Choice :: Steamed Buns");

}

if(c1==3&&c2==1&&c3==2){

fourthframe(f,"Your Choice :: Rice Porridge");

}

if(c1==3&&c2==2&&c3==1){

fourthframe(f,"Your Choice :: HotPot");

}

if(c1==3&&c2==2&&c3==2){

fourthframe(f,"Your Choice :: Wonton");

}

if(c1==4 &&c2==1&&c3==1){

fourthframe(f,"Your Choice :: kimchijigae");

}

if(c1==4 &&c2==1&&c3==2){

fourthframe(f,"Your Choice :: manduguk");

}

if(c1==4 &&c2==2&&c3==1){

fourthframe(f,"Your Choice :: TTeokbokki");

}

if(c1==4 &&c2==2&&c3==2){

fourthframe(f,"Your Choice :: Jajangmyeon");

}

if(c1==5 &&c2==1&&c3==1){

fourthframe(f,"Your Choice :: Cereal");

}

if(c1==5 &&c2==1&&c3==2){

fourthframe(f,"Your Choice :: Pancakes");

}

if(c1==5 &&c2==2&&c3==1){

fourthframe(f,"Your Choice :: Burger");

}

if(c1==5 &&c2==2&&c3==2){

fourthframe(f,"Your Choice :: HotDog");

}

}});

f.add(b);

}

public static void fourthframe(JFrame g,String s)

{

g.getContentPane().removeAll();

g.repaint();

JLabel l=new JLabel(s);

// l.setSize(100,200);

l.setBounds(100,200,200,30);

g.add(l);

}

public static void main(String args[]) throws IOException{

String Filecontent=Readdata("menu.txt");

JFrame f=new JFrame("menu rec");

f.getContentPane().removeAll();

f.repaint();

JLabel l=new JLabel("<html>"+Filecontent.replaceAll("\n","<br/>")+"</html>");

// l.setSize(100,200);

l.setBounds(100,200,200,30);

JTextField t=new JTextField("enter your choice");

t.setBounds(100,500,200,30);

//t.setSize(200,400);

JButton b=new JButton("submit");

b.setBounds(100,800,200,40);

//int c=-1;

b.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent e){

String s1=t.getText();

int c=Integer.parseInt(s1);

System.out.println(c);

if(c==1){

secondframe(f,"<html> 1.Breakfast <br/> 2.lunch </html>",c);

}

else if(c==2){

secondframe(f,"<html> 1.Breakfast <br/> 2.lunch </html>",c);

}

else if(c==3){

secondframe(f,"<html> 1.Breakfast <br/> 2.lunch </html>",c);

}

else if(c==4){

secondframe(f,"<html> 1.Breakfast <br/> 2.lunch </html>",c);

}

else if(c==5){

secondframe(f,"<html> 1.Breakfast <br/> 2.lunch </html>",c);

}

else if(c==6){

secondframe(f,"<html> 1.latte <br/> 2.Cappucino</html>",-1);

}

else if(c==7){

secondframe(f,"<html> 1.Chocolate Milkshake <br/> <br/> 2.oreo milkshake </html>",-1);

}

}});

f.setSize(1000,1000);

f.setLayout(new FlowLayout());

f.setVisible(true);

f.add(l);

f.add(t);

f.add(b);

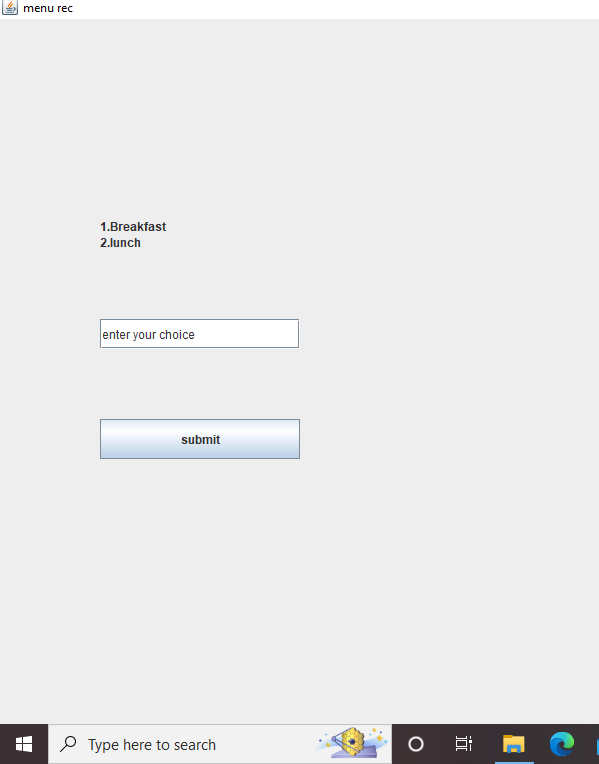
//f.setLayout(null);

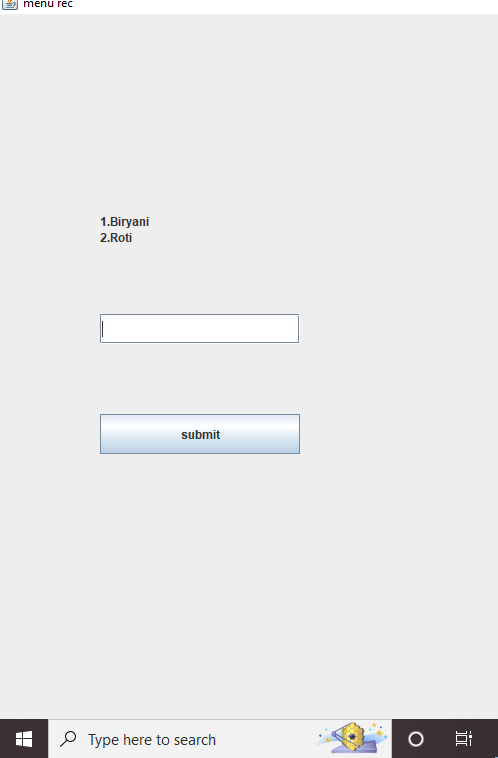
}

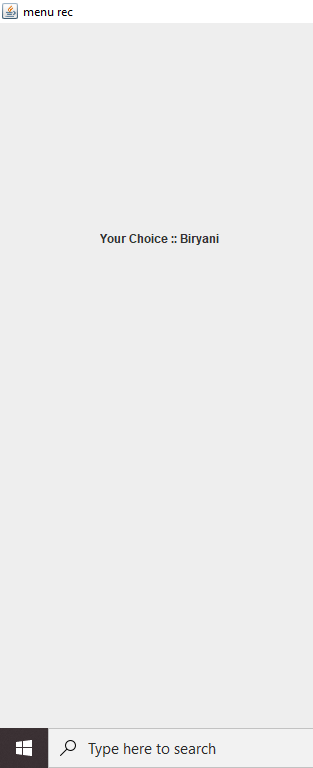
}

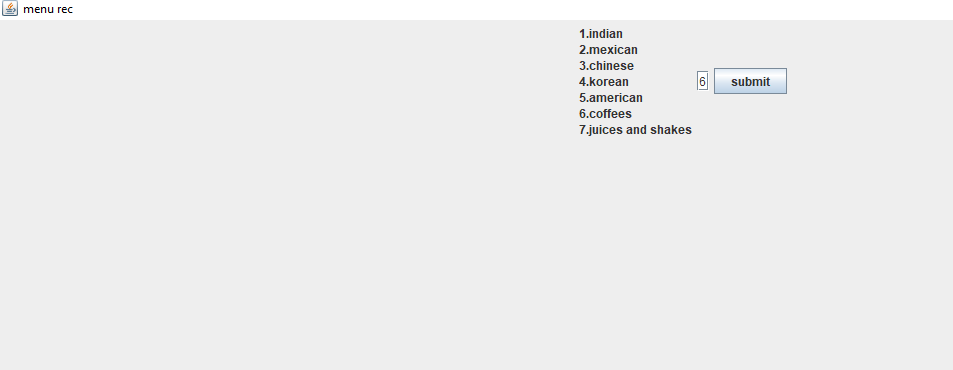
**5. RESULT SCREENS :**

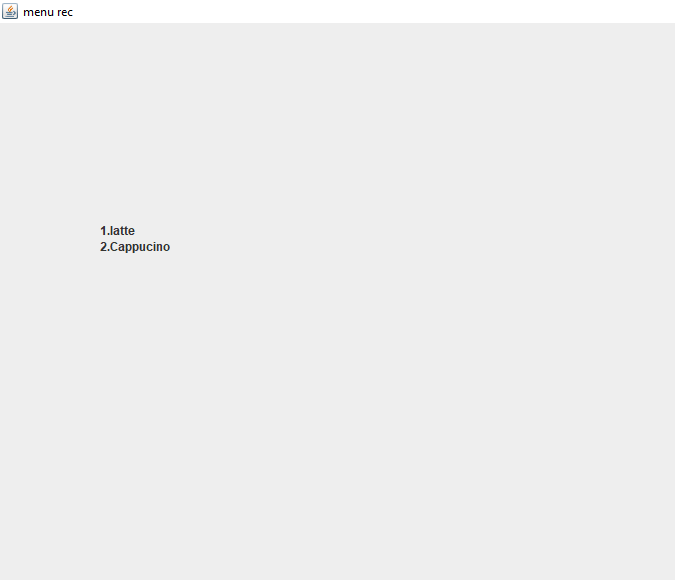


****

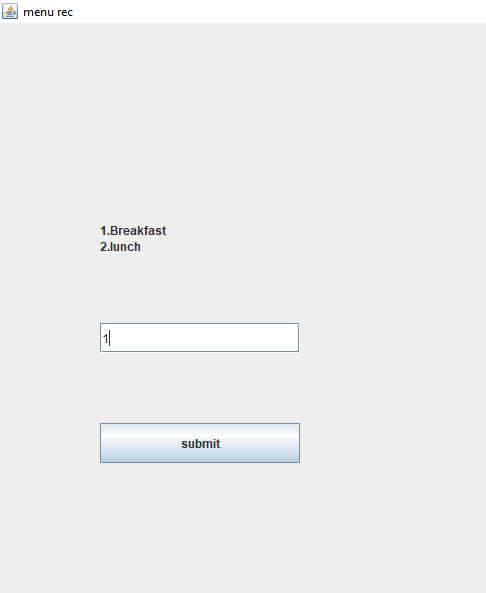


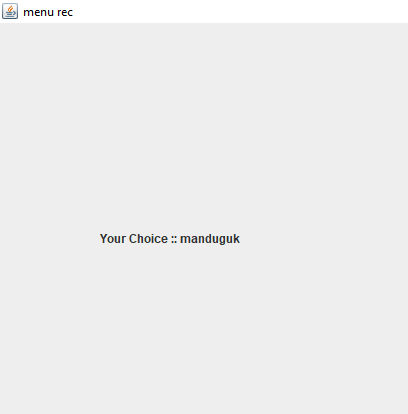






****

****

****

**CONCLUSION :**

* Hence by this interface we can easily choose our favourite cuisine without confusion .And we can easily maintain the data. Creating an algorithm which is having the data of menu .
* This contains all items according to their countries ,cuisines according to the starters desserts etc.
* This will help us to easily classify the items without any confusions.

REFERENCE: