Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Academic Year: 2022-2023

Name – Aksh Dilip Nishar **SAP ID – 60004220123**

Experiment No - 11

AIM: To implement exceptions in Java

1. WAP to to input the data through command Line and Find out total valid and in-valid integers. (Hint: use exception handling).

THEORY:

In Java, an exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime.The core advantage of exception handling is to maintain the normal flow of the application. An exception normally disrupts the normal flow of the application; that is why we need to handle exceptions. The java.lang.Throwable class is the root class of Java Exception hierarchy inherited by two subclasses: Exception and Error. Java Exception Handling in which we are using a try-catch statement to handle the exception.

CODE:

```
import java.util.Scanner;
public class Exp12_1 {
public static void main(String[] args) {
System.out.println("Aksh Nishar 60004220123");
Scanner input = new Scanner(System.in);
System.out.println("Enter the number");
try {
int number = input.nextInt();
System.out.println("Number is valid " + number);
} catch (Exception e) {
System.out.println("EXCEPTION :" +e);
}
System.out.println("Program hasn't crashed");
input.close();
}
}
```

OUTPUT:

```
D:\DJSCE\Sem 3\Java\Java Practice>javac JavaPractice.java

D:\DJSCE\Sem 3\Java\Java Practice>java JavaPractice
Aksh Nishar 60004220123
Enter the number
10
Number is valid 10
Program hasn't crashed

D:\DJSCE\Sem 3\Java\Java Practice>
```

2. Write a Java Program to calculate the Result. Result should consist of name, seatno, date, center number and marks of semester three exam. Create a User Defined Exception class MarksOutOfBoundsException, If Entered marks of any subject is greater than 100 or less than 0, and then program should create a user defined Exception of type MarksOutOfBoundsException and must have a provision to handle it.

THEORY:

In Java, we can create our own exceptions that are derived classes of the Exception class. Steps to create custom Exception Handling : Create a new class whose name should end with an Exception like MarksOutOfBoundsException .This is a convention to differentiate an exception class from regular ones. Make the class extends one of the exceptions which are subtypes of the java.lang.Exception class. Generally, a custom exception class always extends directly from the Exception class. Create a constructor with a String parameter which is the detailed message of the exception. In this constructor, simply call the super constructor and pass the message

Syntax:

```
returntype methodname(){
//code to be executed
methodname();//calling same method
}
```

CODE:

```
import java.util.*;
import java.io.*;
class MarksOutOfBoundException extends Exception
{
MarksOutOfBoundException(String err)
{
System.out.println(err);
}
}
public class Exp12_2{
public static void main(String args[])
{
System.out.println("Aksh Nishar 60004220123");Scanner input = new Scanner(System.in);
int m,m2,m3,seatNo,centerNum,choice=1;
String name,date;
while(choice == 1)
{
try{
System.out.println("Enter the Seat Number : ");
seatNo =input.nextInt();
String str1 = input.nextLine();
System.out.println("Enter Name of Student : ");
name = input.nextLine();
System.out.println("Enter the Center Number : ");
centerNum =input.nextInt();
String str =input.nextLine();
System.out.println("Enter Date : ");
date = input.nextLine();
System.out.println("Enter the Marks in Maths : ");
m = input.nextInt();
System.out.println("Enter the Marks in Chemistry : ");
m2 = input.nextInt();
System.out.println("Enter the Marks in Physics : ");
m3 = input.nextInt();
}
catch(Exception e)
{
System.out.println(e);
}
System.out.println("\nEnter your choice : \n1.Enter more Student data \n2.Exit ");
choice = input.nextInt();
}
}
```

```
public static void main(int seatNo , int centerNo ,String date,String name , int marks ,int marks2,int marks3) throws MarksOutOfBoundException
{
if(marks >= 100 || marks <= 0)
{
throw new MarksOutOfBoundException("Input marks of all subjects should be greater than 0 and less than 100");
}
else if(marks2 >= 100 || marks2 <= 0)
{
throw new MarksOutOfBoundException("Input marks of all subjects should be greater than 0 and less than 100");
}
else if(marks3 >= 100 || marks3 <= 0)
{
throw new MarksOutOfBoundException("Input marks of all subjects should be greater than 0 and less than 100");
}
else{
System.out.println("\nStudent Details :\nName : " + name + "\nSeat Number: " + seatNo + "\nCenter Number : " + centerNo + "\nDate : " + date);
System.out.println("Marks inMaths : " + marks + "\nMarks in physics : " + marks2 + "\nMarks in chemistry: " + marks3 );
}
}
}
```

OUTPUT:

```
D:\DJSCE\Sem 3\Java\Java Practice>java JavaPractice
Aksh Nishar 60004220123
Enter the Seat Number :
123
Enter Name of Student :
Aksh Nishar
Enter the Center Number :
125
Enter Date :
10/10/2022
Enter the Marks in Maths :
99
Enter the Marks in Chemistry:
98
Enter the Marks in Physics :
97

Enter your choice :
1.Enter more Student data
2.Exit
2
```

CONCLUSION:

Hereby, implemented exceptions in Java.

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Academic Year: 2022-2023

Name – Aksh Dilip Nishar **SAP ID - 60004220123**

Experiment No - 12

AIM: To implement Multithreading

1. WAP to print Table of Five, Seven and Thirteen using Multithreading (Use Thread class for the implementation). Also print the total time taken by each thread for the execution.

THEORY:

Multi-threading enables you to write in a way where multiple activities can proceed concurrently in the same program. By definition, multitasking is when multiple processes share common processing resources such as a CPU. Multi-threading extends the idea of multitasking into applications where you can subdivide specific operations within a single application into individual threads. Each of the threads can run in parallel. The OS divides processing time not only among different applications, but also among each thread within an application.

CODE:

```
class Five extends Thread {
public void run() {
long start = System.currentTimeMillis();
for (int i = 1; i < 10; i++) {
System.out.println("5*" + i + "=" + i * 5);
try {
// miliseconds time
Thread.sleep(2000);
} catch (InterruptedException e) {
}
}
long end = System.currentTimeMillis();
System.out.println("Total time taken by 5 table:" + (end
- start));
}
}
class Seven extends Thread {
public void run() {
long start = System.currentTimeMillis();
for (int i = 1; i < 10; i++) {
System.out.println("7*" + i + "=" + i * 7);
try {
// miliseconds time
Thread.sleep(2000);
} catch (InterruptedException e) {
}
}
long end = System.currentTimeMillis();
System.out.println("Total time taken by 7 table:" + (end
-
start));
}
}
class Thirteen extends Thread {
public void run() {
long start = System.currentTimeMillis();
for (int i = 1; i < 10; i++) {
System.out.println("13*" + i + "=" + i * 13);
try {
// miliseconds time
Thread.sleep(2000);
} catch (InterruptedException e) {
}
}
long end = System.currentTimeMillis();
System.out.println("Total time taken by 13 table:" +
(end -
start));
}
}
public class Exp13_1 {
public static void main(String[] args) {
System.out.println("Aksh Nishar 60004220123");
Five f = new Five();
Seven s = new Seven();
Thirteen t = new Thirteen();
```

```
f.start();                                  }
s.start();                                  }
t.start();
```

OUTPUT:

```
D:\DJSCE\Sem 3\Java\Java Practice>java JavaPractice
Aksh Nishar 60004220123
5*1=5
5*2=10
5*3=15
5*4=20
5*5=25
5*6=30
5*7=35
5*8=40
5*9=45
Total time taken by 5 table:94
7*1=7
13*1=13
7*2=14
13*2=26
13*3=39
7*3=21
13*4=52
7*4=28
7*5=35
13*5=65
7*6=42
13*6=78
7*7=49
7*8=56
13*7=91
7*9=63
13*8=104
Total time taken by 7 table:118
13*9=117
Total time taken by 13 table:121
```

2. Write java program to implement the concept of Thread Synchronization

THEORY:

When we start two or more threads within a program, there may be a situation when multiple threads try to access the same resource and finally they can produce unforeseen result due to concurrency issues. The function Thread.sleep() is used so that it sleeps a thread for the specified amount of time.Till the time another thread is running. The function isAlive() is used so that it tests if the thread is alive.(It returns a boolean value).

CODE:

```
class Movie extends Thread {
int vacant = 1, required;
Movie(int x) {
required = x;
}
public synchronized void run() {
if (required <= vacant) {
System.out.println(required + " for " +
Thread.currentThread().getName());
try {
Thread.sleep(100);
} catch (Exception e) {
}
vacant = vacant - required;
} else {
System.out.println("none for " +
Thread.currentThread().getName());
}
}
}
class Exp13_2 {
public static void main(String z[]) {
System.out.println("Aksh      Nishar
60004220123");
Movie m = new Movie(1);
Thread t1 = new Thread(m);
Thread t2 = new Thread(m);
t1.setName("Prayag");
t2.setName("Aksh");
t1.start();
t2.start();
}
}
```

OUTPUT:

```
D:\DJSCE\Sem 3\Java\Java Practice>javac JavaPractice.java

D:\DJSCE\Sem 3\Java\Java Practice>java JavaPractice
Aksh Nishar 60004220123
1 for Prayag
none for Aksh
```

CONCLUSION: Thus, we implemented programs on Multithreading.

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Academic Year: 2022-2023

Name – Aksh Dilip Nishar **SAP ID - 60004220123**

Experiment No - 13

AIM: To implement Applets

1. Write a java program to draw the house on an applet

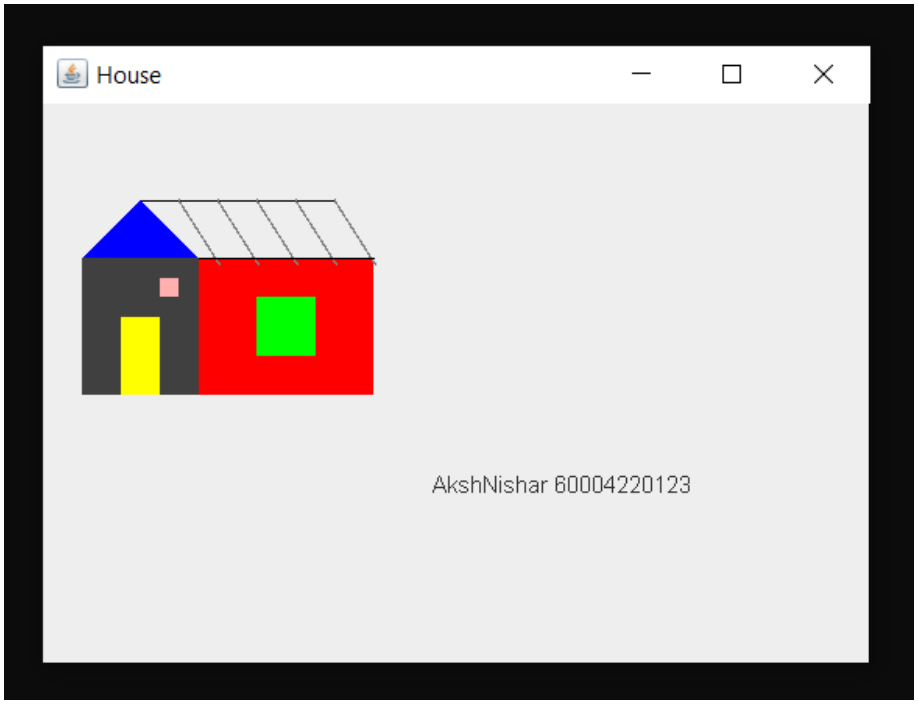
THEORY:

The javax.swing.JFrame class is a type of container which inherits the java.awt.Frame class. JFrame works like the main window where components like labels, buttons, textfields are added to create a GUI. Unlike Frame, JFrame has the option to hide or close the window with the help of setDefaultCloseOperation(int) method. Here, I have created a house using VSCode IDE .

CODE:

```
import java.awt.Color;
import java.awt.Graphics;
import javax.swing.JFrame;
import javax.swing.JPanel;
public class Exp14_1{
public static void main(String[] args) {
JFrame jf = new JFrame();
jf.setVisible(true);
jf.setTitle("House");
jf.setSize(300, 300);
jf.setLocation(300, 100);
jf.setDefaultCloseOperation(JFrame.EXIT_ON_C
LOSE);
JPanel p = new JPanel() {
@Override
public void paint(Graphics g) {
g.drawString("Aksh                    Nishar
60004220123",200,200);
g.setColor(Color.DARK_GRAY);
g.fillRect(20, 80, 60, 70);
g.setColor(Color.RED);
g.fillRect(80, 80, 90, 70);
g.setColor(Color.PINK);
g.fillRect(60, 90, 10, 10);
g.setColor(Color.yellow);
g.fillRect(40, 110, 20, 40);
g.setColor(Color.BLACK);
g.drawLine(50, 50, 150, 50);
g.setColor(Color.GRAY);
g.drawLine(150, 50, 171, 83);
g.drawLine(110, 50, 131, 83);
g.drawLine(130, 50, 151, 83);
g.drawLine(90, 50, 111, 83);
g.drawLine(70, 50, 91, 83);
g.setColor(Color.BLACK);
g.drawLine(80, 80, 170, 80);
g.setColor(Color.GREEN);
g.fillRect(110, 100, 30, 30);
g.setColor(Color.BLUE);
g.fillPolygon(new int[] { 50, 20, 80 }, new int[] {
50, 80,
80 }, 3);
}
};
jf.add(p);
}
}
```

OUTPUT:



2. Write java program to create an advertisement banner on an applet using multithreading

THEORY:

The following program implements the use of abstract class where we declare all the functions and define and Applets are embeddable Java applications that are expected to start and stop themselves on command, possibly many times in their lifetime. A Java-enabled web browser normally starts an applet when the applet is displayed and stops it when the user moves to another page or (in theory) when the user scrolls the applet out of view. To conform to this API, we would like an applet to cease its nonessential activity when it is stopped and resume it when started again. An important compromise was made early in the design of Swing relating to speed, GUI consistency, and thread safety. To provide maximum performance and simplicity in the common case, Swing does not explicitly synchronize access to most Swing component methods. This means that most Swing components are, technically, not threadsafe for multithreaded applications.

CODE:

```
import javax.swing.*;
import java.awt.*;
class MyFrame extends JFrame implements Runnable {
Container c;
JLabel title, ad,ad1,name;
public MyFrame() {
// add(new Board());
setTitle("Advertisment");
setBounds(300, 90, 900, 600);
setDefaultCloseOperation(EXIT_ON_CLOSE);
setResizable(false);
setVisible(true);
c = getContentPane();
c.setBackground(Color.black);
c.setLayout(null);
title = new JLabel("SALE SALE SALE !!!");
title.setSize(300, 50);
title.setLocation(350, 30);
title.setForeground(Color.white);
title.setFont(new Font("Verdana", Font.BOLD, 38));
c.add(title);
name = new JLabel("Mihir Vora 60004220115");
name.setSize(500, 50);
name.setForeground(Color.red);
name.setFont(new Font("Verdana", Font.BOLD, 25));
name.setLocation(250, 250);
c.add(name);
ad = new JLabel("FLAT 50% Discount hurray");
ad.setSize(400, 30);
ad.setForeground(Color.red);
```

```
ad.setFont(new Font("Verdana", Font.BOLD, 25));
ad.setLocation(350, 90);
c.add(ad);
ad1 = new JLabel("For first few Customers");
ad1.setSize(400, 30);
ad1.setForeground(Color.pink);
ad1.setFont(new Font("Verdana", Font.BOLD, 19));
ad1.setLocation(370, 120);
c.add(ad1);
ad1 = new JLabel("So , what are you waiting for ?");
ad1.setSize(400, 30);
ad1.setForeground(Color.yellow);
ad1.setFont(new Font("Verdana", Font.BOLD, 9));
ad1.setLocation(390, 150);
c.add(ad1);
new Thread(this).start();
}
public void run() {
try {
while (true) {
if (title.getText() == null) {
title.setText("SALE SALE SALE !!!");
Thread.sleep(500);
} else {
title.setText(null);
Thread.sleep(500);
}
}
} catch (InterruptedException ex) {
}
}
}
public class Exp14_2{
public static void main(String[] args) {
new MyFrame();
}
}
```

OUTPUT:

CONCLUSION: Thus, we implemented programs on applets.

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Academic Year: 2022-2023

Name – Aksh Dilip Nishar **SAP ID - 60004220123**

Experiment No - 14

AIM: Designing Graphical User Interfaces in Java using AWT and Event handling

1. Writa a java program to create a registration form using AWT.

THEORY:

Java AWT (Abstract Window Toolkit) is an API to develop Graphical User Interface (GUI) or windows-based applications in Java. Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavy weight i.e. its components are using the resources of underlying operating system (OS). The java.awt package provides classes for AWT API such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc. The AWT tutorial will help the user to understand Java GUI programming in simple and easy steps.

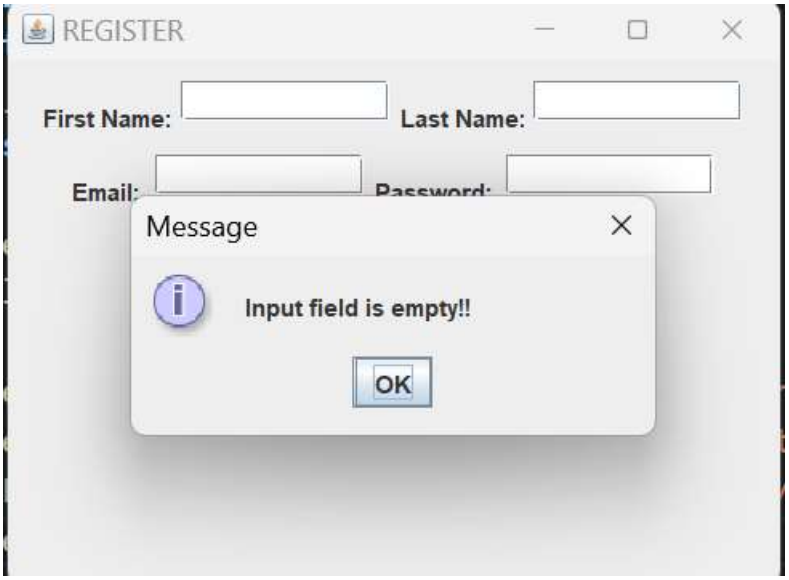
CODE:

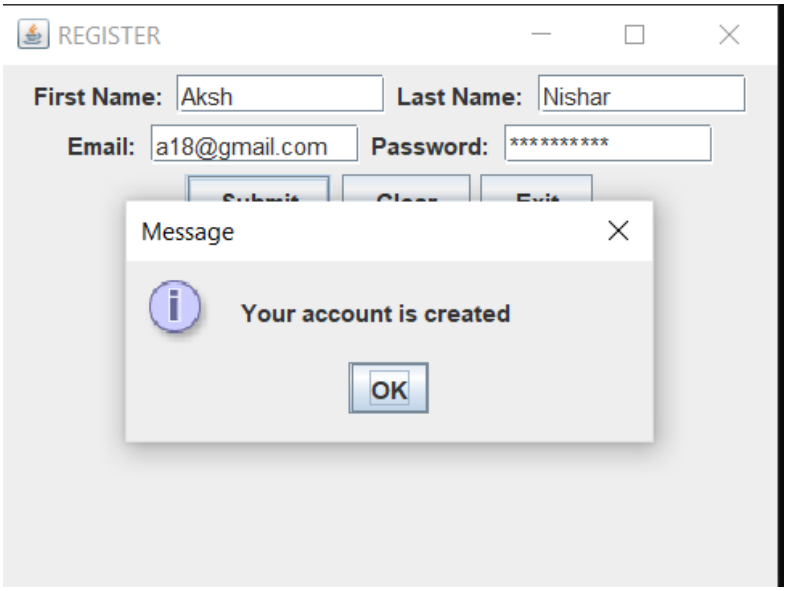
```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
class Exp15_1 extends JFrame implements ActionListener {
Container c;
JLabel lFName, lLName, lPassword, lEmail;
JTextField txtFName, txtLName;
JTextField txtEmail;
JPasswordField txtPassword;
JButton btnSubmit, btnClear, btnExit;
String strFName, strLName, strPassword, strEmail;
Exp15_1() {
c = getContentPane();
c.setLayout(new FlowLayout());
// labels
lFName = new JLabel("First Name: ");
lLName = new JLabel("Last Name: ");
lPassword = new JLabel("Password: ");
lEmail = new JLabel("Email: ");
// text fields
txtFName = new JTextField(10);
txtLName = new JTextField(10);
txtEmail = new JTextField(10);
txtPassword = new JPasswordField(10);
txtPassword.setEchoChar('*');
// buttons
btnSubmit = new JButton("Submit");
btnClear = new JButton("Clear");
btnExit = new JButton("Exit");
c.add(lFName);
c.add(txtFName);
c.add(lLName);
c.add(txtLName);
c.add(lEmail);
c.add(txtEmail);
c.add(lPassword);
c.add(txtPassword);
c.add(btnSubmit);
c.add(btnClear);
c.add(btnExit);
btnSubmit.addActionListener(this);
btnClear.addActionListener(this);
btnExit.addActionListener(this);
}
public void actionPerformed(ActionEvent ae) {
if (ae.getSource() == btnSubmit) {
strLName = txtLName.getText();
strFName = txtFName.getText();
strPassword = txtPassword.getText();
strEmail = txtEmail.getText();
if (strFName.equals("") || strLName.equals("") ||
strPassword.equals("") ||
strEmail.equals("")) {
JOptionPane.showMessageDialog(c, "Input field is
empty!!");
txtFName.requestFocus();
} else {
JOptionPane.showMessageDialog(c, "Your account is
created ");
txtFName.setText("");
txtPassword.setText("");
System.exit(0);
}
```

```
} else if (ae.getSource() == btnClear) {
txtFName.setText("");
txtPassword.setText("");
txtLName.setText("");
txtEmail.setText("");
txtFName.requestFocus();
} else {
System.exit(0);
}
}
```

```
public static void main(String z[]) {
Exp15_1 frm = new Exp15_1();
frm.setDefaultCloseOperation(JFrame.EXIT_ON_CL
OSE);
frm.setBounds(200, 200, 400, 300);
frm.setVisible(true);
frm.setTitle("REGISTER");
}
}
```

OUTPUT:





2. On Applet: Take a Login and Password from the user and display it on the third Text Field which appears only on clicking OK button and clear both the Text Fields on clicking RESET button

THEORY:

In this program we have created a login form where used gets logged in if he enters the correct username and password. After OK button is pressed the username and password is checked, if it matches the actual credentials then used successfully logs in or else it's an unsuccessful login.
Components Used:

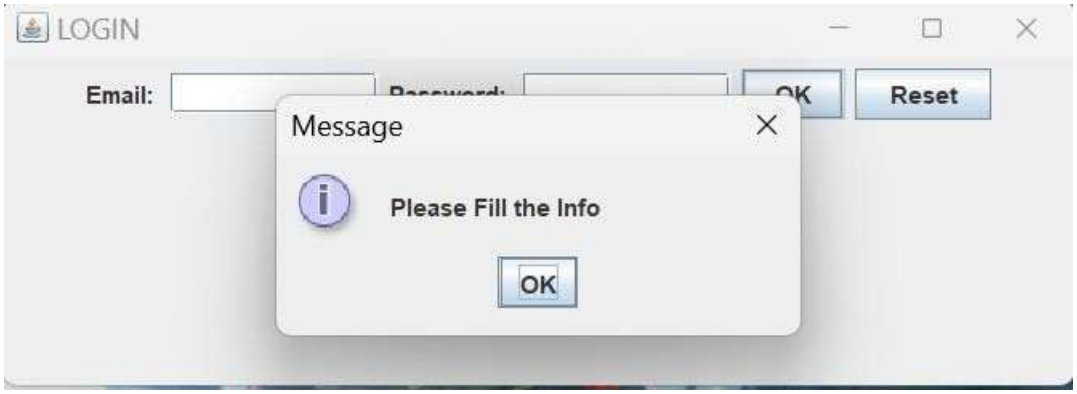
- JFrame
- JTextField
- JPasswordField
- JButton

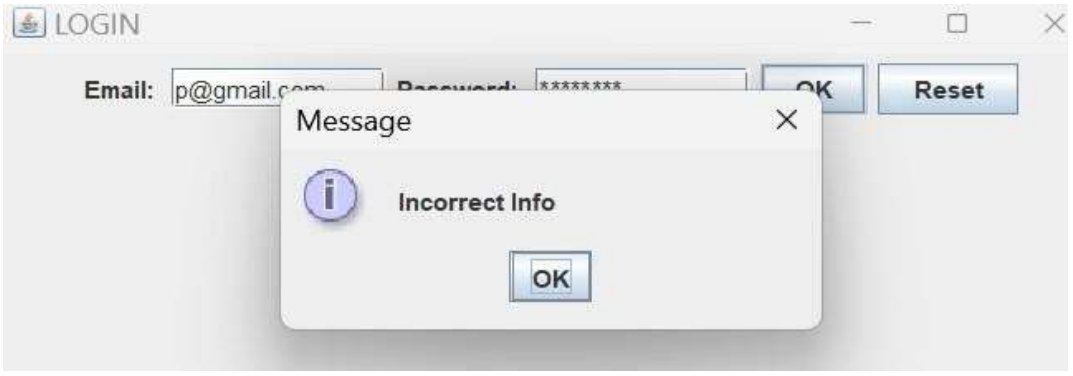
CODE:

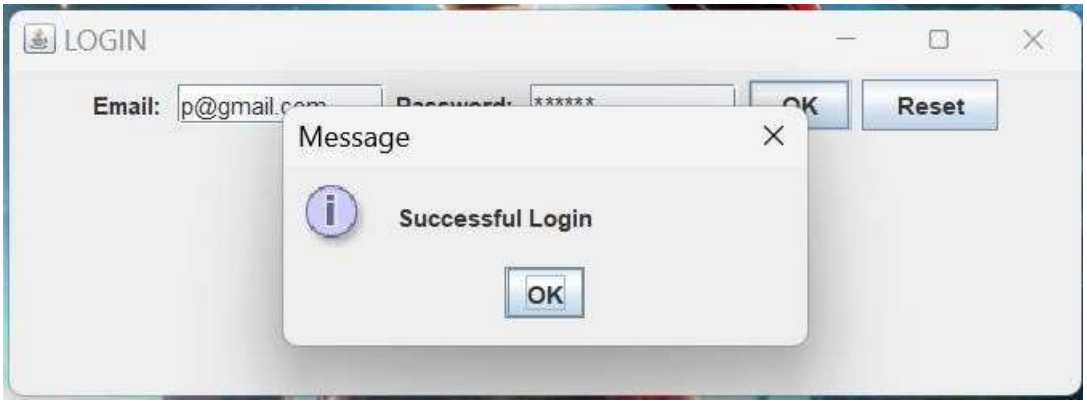
```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
class LoginForm extends JFrame implements ActionListener {
Container c;
JLabel lPassword, lEmail;
JTextField txtEmail;
JPasswordField txtPassword;
JButton btnSubmit, btnClear;
String strPassword, strEmail;
```

```
LoginForm() {
c = getContentPane();
c.setLayout(new FlowLayout());
// labels
lPassword = new JLabel("Password: ");
lEmail = new JLabel("Email: ");
// text fields
txtEmail = new JTextField(10);
txtPassword = new JPasswordField(10);
txtPassword.setEchoChar('*');
// buttons
btnSubmit = new JButton("OK");
btnClear = new JButton("Reset");
c.add(lEmail);
c.add(txtEmail);
c.add(lPassword);
c.add(txtPassword);
c.add(btnSubmit);
c.add(btnClear);
btnSubmit.addActionListener(this);
btnClear.addActionListener(this);
}
public void actionPerformed(ActionEvent ae) {
if (ae.getSource() == btnSubmit) {
strPassword = txtPassword.getText();
strEmail = txtEmail.getText();
if (strPassword.equals("Prayag") &&
strEmail.equals("p@gmail.com")) {
JOptionPane.showMessageDialog(c, "Successful Login");
System.exit(0);
} else {
JOptionPane.showMessageDialog(c, "Incorrect Info");
txtEmail.setText("");
txtPassword.setText("");
}
} else if (ae.getSource() == btnClear) {
txtPassword.setText("");
txtEmail.setText("");
}
}
public static void main(String z[]) {
LoginForm frm = new LoginForm();
frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frm.setBounds(200, 200, 550, 200);
frm.setVisible(true);
frm.setTitle("LOGIN");
}}
```

OUTPUT:







CONCLUSION: Thus, we implemented Graphical User Interfaces in Java using AWT and Event handling.

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Academic Year: 2022-2023

Name – Aksh Dilip Nishar **SAP ID -60004220123**

Experiment No - 15

AIM: : Develop simple swing applications and complex GUI using Java Swing classes

1. WAP Write a program to create a window with four text fields for the name, street, city and pin code with suitable labels. Also windows contains a button MyInfo. When the user types the name, his street, city and pincode and then clicks the button, the types details must appear in Arial Font with Size 32, Italics.

THEORY:

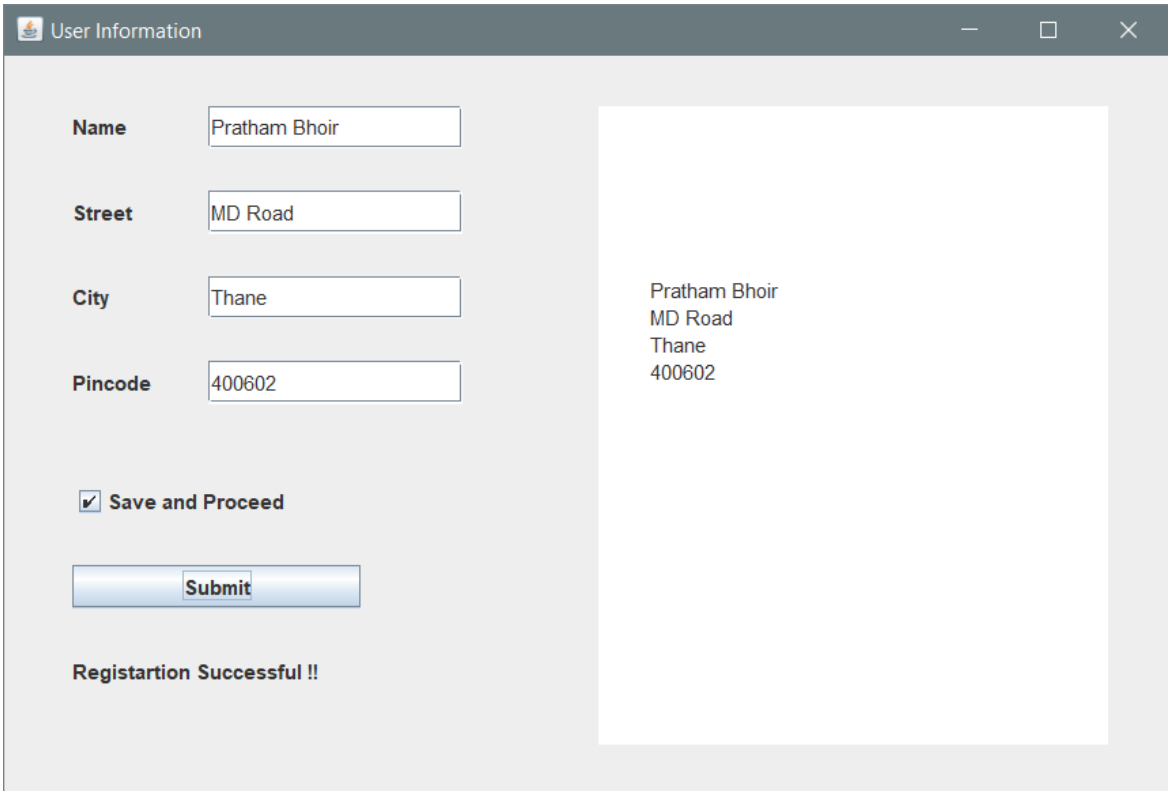
Swing in Java is a lightweight GUI toolkit which has a wide variety of widgets for building optimized window based applications. It is a part of the JFC(Java Foundation Classes). It is build on top of the AWT API and entirely written in java. It is platform independent unlike AWT and has lightweight components. It becomes easier to build applications since we already have GUI components like button, checkbox etc. This is helpful because we do not have to start from the scratch. Different methods of JFrame class are : public void add(Component c) : Inserts a component on this component. public void setSize(int width,int height) : Sets the size (width and height) of the component. public void setLayout(LayoutManager m) : Defines the layout manager for the component. public void setVisible(boolean status) : Changes the visibility of the component, by default false. An object of the java.awt.Font class represents a font in a Java program

CODE:

```
import javax.swing.*; import javax.swing.JFrame;
import javax.swing.border.EmptyBorder; import java.awt.*;
import java.awt.event.*;

class MyFrame extends JFrame implements ActionListener { JLabel label1, label2, label3, label4;

JTextField t1, t2, t3, t4; JTextArea screen; JCheckBox terms; JButton submit;
JLabel msg;


MyFrame()
{
setTitle("User Information"); setSize(700, 500);
setDefaultCloseOperation(EXIT_ON_CLOSE); Container c = getContentPane(); c.setLayout(null);
// name label
label1 = new JLabel("Name"); label1.setBounds(40, 30, 100, 25); c.add(label1);
// name inputbox
t1 = new JTextField(); t1.setBounds(120, 30, 150, 25); c.add(t1);
// Street label
label2 = new JLabel("Street"); label2.setBounds(40, 80, 100, 25);

c.add(label2);
```

```
// street inputbox
t2 = new JTextField();


t2.setBounds(120, 80, 150, 25); c.add(t2);
// City label
label3 = new JLabel("City"); label3.setBounds(40, 130, 100, 25); c.add(label3);
// city inputbox
t3 = new JTextField(); t3.setBounds(120, 130, 150, 25); c.add(t3);
// pincode label
label4 = new JLabel("Pincode"); label4.setBounds(40, 180, 100, 25); c.add(label4);
// pincode textfield
t4 = new JTextField(); t4.setBounds(120, 180, 150, 25); c.add(t4);
// checkbox
terms = new JCheckBox("Save and Proceed"); terms.setBounds(40, 250, 170, 25);

c.add(terms);

// submit button
submit = new JButton("Submit"); submit.setBounds(40, 300, 170, 25);

c.add(submit); submit.addActionListener(this);
// display of form details screen = new JTextArea();
screen.setBounds(350, 30, 300, 375);
screen.setBorder(new EmptyBorder(100, 30, 10, 10)); c.add(screen);
// message
msg = new JLabel(""); msg.setBounds(40, 350, 200, 25); c.add(msg);
setVisible(true);
}


public void actionPerformed(ActionEvent e) { if (terms.isSelected()) {
msg.setText("Registartion Successful !!"); String name = t1.getText();
String street = t2.getText(); String city = t3.getText(); String pincode = t4.getText();
screen.setText(name + "\n" + street + "\n" + city + "\n" + pincode
+ "\n");
} else {

msg.setText("Kindly Save and Procced !!"); screen.setText(" ");
}
}
}


public class Main {
public static void main(String args[]) { MyFrame frame = new MyFrame();
}
}
```

Output:


User Information
Name
Pratham Bhoir
Street
MD Road
City
Thane
Pincode
400602
Save and Proceed
Submit
Registartion Successful !!
Pratham Bhoir
MD Road
Thane
400602

2. WA applet with 4 swing buttons with suitable texts on them. When the user presses a button a message should appear in the label as to which button was pressed by the user

THEORY:

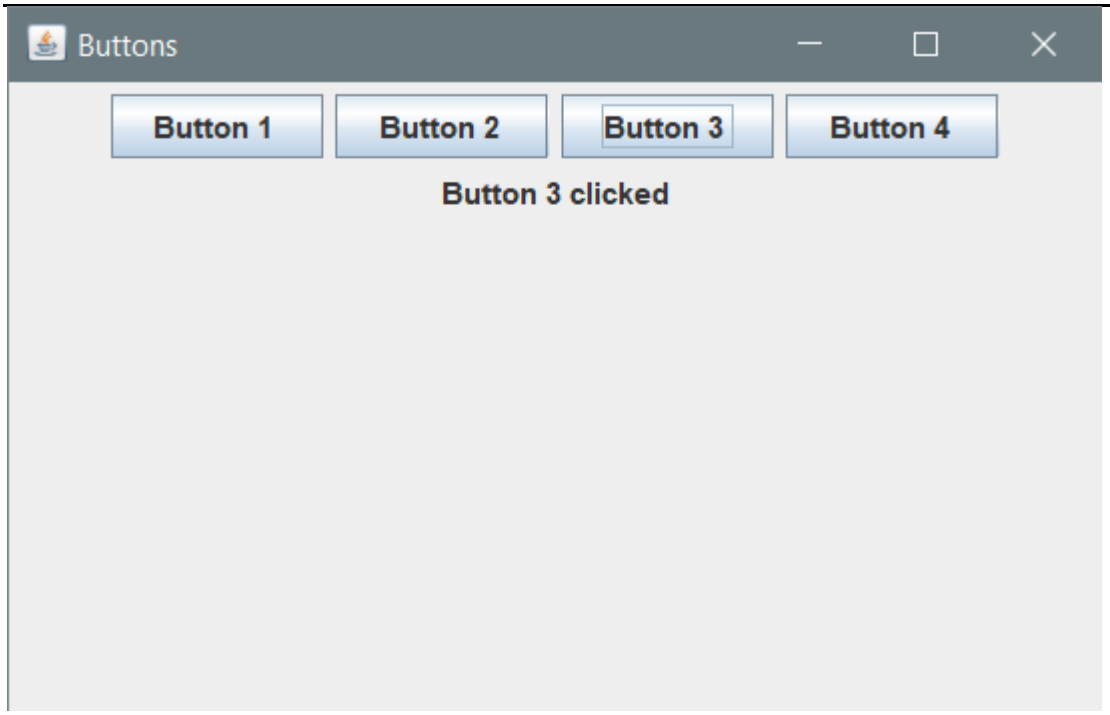
Interface ActionListener : The listener interface for receiving action events. The class that is interested in processing an action event implements this interface, and the object created with that class is registered with a component, using the component's addActionListener method. When the action event occurs, that object's actionPerformed method is invoked. The EventObject contains the getSource( ) method.Suppose you have many buttons in your application. So, you can find which button is clicked by using the getSource() method. The getSource( ) method returns the source of the event. Java CollationElementIterator setText(String source) Set a new string over which to iterate.

CODE:

```
import javax.swing.*; import javax.swing.JFrame; import java.awt.*;
import java.awt.event.*;


class MyFrame extends JFrame implements ActionListener { JCheckBox terms;
JButton btn1, btn2, btn3, btn4; JLabel msg;

MyFrame() { setTitle("Buttons"); setSize(700, 500);
setDefaultCloseOperation(EXIT_ON_CLOSE); Container c = getContentPane();

c.setSize(300, 25);
c.setLayout(new GridLayout(2, 2, 20, 20)); c.setLayout(new FlowLayout());
// button 1
btn1 = new JButton("Button 1 "); btn1.setBounds(40, 30, 250, 25); btn1.addActionListener(this);
c.add(btn1);
// button 2


btn2 = new JButton("Button 2 "); btn2.setBounds(40, 80, 250, 25); btn2.addActionListener(this);
c.add(btn2);
// button 3
btn3 = new JButton("Button 3 "); btn3.setBounds(40, 130, 250, 25); btn3.addActionListener(this);
c.add(btn3);
// button 4
```

```
btn4 = new JButton("Button 4 "); btn4.setBounds(40, 180, 250, 25); btn4.addActionListener(this);
c.add(btn4);
// message
msg = new JLabel(""); msg.setBounds(40, 350, 200, 25);

c.add(msg); setVisible(true);
}


public void actionPerformed(ActionEvent e) {


if (e.getSource() == btn1) { msg.setText("Button 1 clicked");
} else if (e.getSource() == btn2) { msg.setText("Button 2 clicked");
} else if (e.getSource() == btn3) { msg.setText("Button 3 clicked");
} else if (e.getSource() == btn4) { msg.setText("Button 4 clicked");
} else {
msg.setText("No button clicked");
}
}
}
public class Main {
public static void main(String args[]) {


MyFrame frame = new MyFrame();
}
}
```

Output:



CONCLUSION: Thus we implemented simple swing applications and complex GUI using Java Swing classes.