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

JAVA

EXPERIMENTS

Name:	Prerna Sunil Jadhav
Sap ID:	60004220127
Branch:	Computer Engineering
Semester:	III (DSE)
Course Code:	DJ19CEL306
Course:	Programming Laboratory – I (Java)
Experiments:	11 to 15/16

Name – Prerna Sunil Jadhav **SAP ID - 60004220127**

Experiment No – 11

AIM: To implement exceptions in Java (CO5)

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 (i): Write a Java Program to input the data through command Line and Find out total valid and in-valid integers. (Hint: use exception handling)

args.java

```
Exp11 > args.java > ...
2   public class args{
        Run | Debug
3       public static void main(String[] args){
4           try {
5               System.out.println(x: "Prerna Sunil Jadhav - 60004220127\n");
6               for (int i = 0; i < args.length; i++) {
7                   int number = Integer.parseInt(args[i]);
8                   System.out.println("You entered: "+number);
9               }
10          }
11          catch (NumberFormatException e) {
12              System.out.println(x: "NumberFormatException occurred");
13          }
14      }
15  }
```

Output:

```
C:\Users\Jadhav\Java\jdk1.8.0_291\bin>javac args.java

C:\Users\Jadhav\Java\jdk1.8.0_291\bin>java args 56 5t
Prerna Sunil Jadhav - 60004220127

You entered: 56
NumberFormatException occurred
```

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

CODE (ii): 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.

Code1_MarksException.java 9

Exp11 > Code1_MarksException.java > Code1_MarksException > main(String[])

```
1   package Exp11;
2   import java.util.*;
3
4   class MarksOutOfBoundException extends Exception {
5       MarksOutOfBoundException(String err) {
6           System.out.println(err);
7       }
8   }
9
10  public class Code1_MarksException {
        Run | Debug
11      public static void main(String args[]) {
12          System.out.println(x: "Prerna Jadhav - 60004220127\n");
13          Scanner input = new Scanner(System.in);
14          int choice = 1;
15          while (choice == 1) {
16              try {
17                  int m, m2, m3, seatNo, centerNum;
18                  String name, date;
19                  System.out.println(x: "Enter the Seat Number : ");
20                  seatNo = input.nextInt();
21                  String str1 = input.nextLine();
22                  System.out.println(x: "Enter Name of Student : ");
23                  name = input.nextLine();
24                  System.out.println(x: "Enter the Center Number : ");
25                  centerNum = input.nextInt();
26                  String str = input.nextLine();
27                  System.out.println(x: "Enter Date : ");
28                  date = input.nextLine();
29                  System.out.println(x: "Enter the Marks in Maths : ");
30                  m = input.nextInt();
```

Code1_MarksException.java 9

Exp11 > Code1_MarksException.java > Code1_MarksException > main(String[])

```
31                  System.out.println(x: "Enter the Marks in Chemistry : ");
32                  m2 = input.nextInt();
33                  System.out.println(x: "Enter the Marks in Physics : ");
34                  m3 = input.nextInt();
35              } catch (Exception e) {
36                  System.out.println(e);
37              }
38              System.out.println(x: "\nEnter your choice : \n1.Enter more Student data \n2.Exit ");
39              choice = input.nextInt();
40          }
41          input.close();
42      }
43      public static void main(int seatNo, int centerNo, String date, String name, int marks, int marks2, ir
44              throws MarksOutOfBoundException {
45          if (marks >= 100 || marks <= 0) {
46              throw new MarksOutOfBoundException(
47                      err: "Input marks of all subjects should be greater than 0 and less than 100");
48          } else if (marks2 >= 100 || marks2 <= 0) {
49              throw new MarksOutOfBoundException(
50                      err: "Input marks of all subjects should be greater than 0 and less than 100");
51          } else if (marks3 >= 100 || marks3 <= 0) {
52              throw new MarksOutOfBoundException(
53                      err: "Input marks of all subjects should be greater than 0 and less than 100");
54          } else {
55              System.out.println("\nStudent Details :\nName : " + name + "\nSeat Number: " + seatNo + "\nCe
56                      + centerNo + "\nDate : " + date);
57              System.out.println(
58                      "Marks inMaths : " + marks + "\nMarks in physics : " + marks2 + "\nMarks in chemistry
59          }
60      }
61  }
```

OUTPUT:

```
Prerna Jadhav - 60004220127

Enter the Seat Number :
9800
Enter Name of Student :
Prerna
Enter the Center Number :
32op
java.util.InputMismatchException

Enter your choice :
1.Enter more Student data
2.Exit
Exception in thread "main" java.util.InputMismatchException
        at java.base/java.util.Scanner.throwFor(Scanner.java:939)
        at java.base/java.util.Scanner.next(Scanner.java:1594)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2258)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2212)
        at Exp11.Code1_MarksException.main(Code1_MarksException.java:39)
```

CONCLUSION: Hereby, implemented exceptions in Java.

Name – Prerna Sunil Jadhav **SAP ID - 60004220127**

Experiment No – 12

AIM: To implement Multithreading

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 (i): 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.

Code1_Multithreading.java

Exp12 > Code1_Multithreading.java > ...

```java
package Exp12;

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(millis: 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(millis: 2000);
            } catch (InterruptedException e) {
            }
        }
        long end = System.currentTimeMillis();
        System.out.println("Total time taken by 7 table:" + (end -
                start));
    }
}
```

Code1_Multithreading.java

Exp12 > Code1_Multithreading.java > ...

```java

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(millis: 2000);
            } catch (InterruptedException e) {
            }
        }
        long end = System.currentTimeMillis();
        System.out.println("Total time taken by 13 table:" + (end -
                start));
    }
}

public class Code1_Multithreading {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Prerna Sunil Jadhav - 60004220127");
        Five f = new Five();
        Seven s = new Seven();
        Thirteen t = new Thirteen();
        f.start();
        s.start();
        t.start();
    }
}
```

Output:

```
Prerna Sunil Jadhav - 60004220127
7*1=7
13*1=13
5*1=5
```

```
5*8=40
13*8=104
7*9=63
5*9=45
13*9=117
Total time taken by 7 table:18091
Total time taken by 5 table:18182
Total time taken by 13 table:18239
```

CONCLUSION: Hereby, implemented multithreading in Java.

AIM: 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: Write java program to implement the concept of Thread Synchronization

Code2_ThreadSync.java

Exp12 > Code2_ThreadSync.java > { } Exp12

```
1   package Exp12;
2   class Movie extends Thread {
3       int vacant = 1, required;
4       Movie(int x) {
5           required = x;
6       }
7       public synchronized void run() {
8           if (required <= vacant) {
9               System.out.println(required + " for " +
10                      Thread.currentThread().getName());
11              try {
12                  Thread.sleep(millis: 100);
13              } catch (Exception e) {
14              }
15              vacant = vacant - required;
16          } else {
17              System.out.println("none for " +
18                      Thread.currentThread().getName());
19          }
20      }
21  }
22  class Code2_ThreadSync {
        Run | Debug
23      public static void main(String z[]) {
24          System.out.println(x: "Prerna Sunil Jadhav - 60004220127");
25          Movie m = new Movie(x: 1);
26          Thread t1 = new Thread(m);
27          Thread t2 = new Thread(m);
28          t1.setName(name: "Prerna");
29          t2.setName(name: "Jadhav");
30          t1.start();
31          t2.start();
32      }
33  }
```

OUTPUT:

```
Prerna Sunil Jadhav - 60004220127
1 for Prerna
none for Jadhav
```

CONCLUSION: Thus, we implemented programs on Multithreading.

Name – Prerna Sunil Jadhav **SAP ID - 60004220127**

Experiment No – 13

AIM: To implement Applets (CO6)

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 .

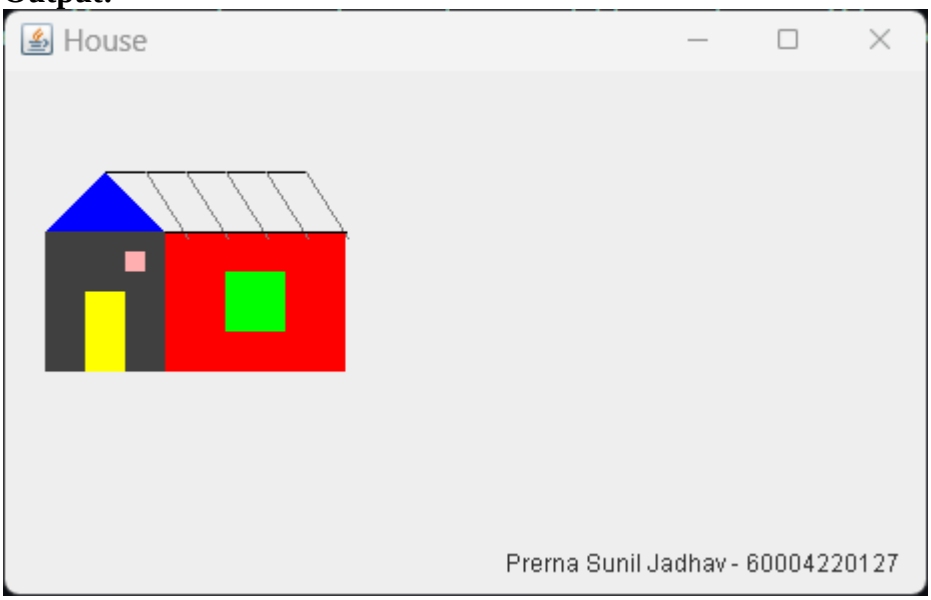
CODE (i): Write a java program to draw the house on an applet

Code1_house.java

Exp13 > Code1_house.java > ...

```java
package Exp13;
import java.awt.Color;
import java.awt.Graphics;
import javax.swing.JFrame;
import javax.swing.JPanel;
public class Code1_house {
    Run | Debug
    public static void main(String[] args) {
        JFrame jf = new JFrame();
        jf.setVisible(b: true);
        jf.setTitle(title: "House");
        jf.setSize(width: 300, height: 300);
        jf.setLocation(x: 300, y: 100);
        jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JPanel p = new JPanel() {
            public void paint(Graphics g) {
                g.drawString(str: "Prerna Sunil Jadhav - 60004220127", x: 250, y: 250);
                g.setColor(Color.DARK_GRAY);
                g.fillRect(x: 20, y: 80, width: 60, height: 70);
                g.setColor(Color.RED);
                g.fillRect(x: 80, y: 80, width: 90, height: 70);
                g.setColor(Color.PINK);
                g.fillRect(x: 60, y: 90, width: 10, height: 10);
                g.setColor(Color.yellow);
                g.fillRect(x: 40, y: 110, width: 20, height: 40);
                g.setColor(Color.BLACK);
                g.drawLine(x1: 50, y1: 50, x2: 150, y2: 50);
                g.setColor(Color.GRAY);
                g.drawLine(x1: 150, y1: 50, x2: 171, y2: 83);
                g.drawLine(x1: 110, y1: 50, x2: 131, y2: 83);
                g.drawLine(x1: 130, y1: 50, x2: 151, y2: 83);
                g.drawLine(x1: 90, y1: 50, x2: 111, y2: 83);
                g.drawLine(x1: 70, y1: 50, x2: 91, y2: 83);
                g.setColor(Color.BLACK);
                g.drawLine(x1: 80, y1: 80, x2: 170, y2: 80);
                g.setColor(Color.GREEN);
                g.fillRect(x: 110, y: 100, width: 30, height: 30);
                g.setColor(Color.BLUE);
                g.fillPolygon(new int[] { 50, 20, 80 }, new int[] { 50, 80,80 }, nPoints: 3);
            }
        };
        jf.add(p);
    }
```

Output:



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 (ii): Write java program to create an advertisement banner on an applet using multithreading

Code2_Banner.java

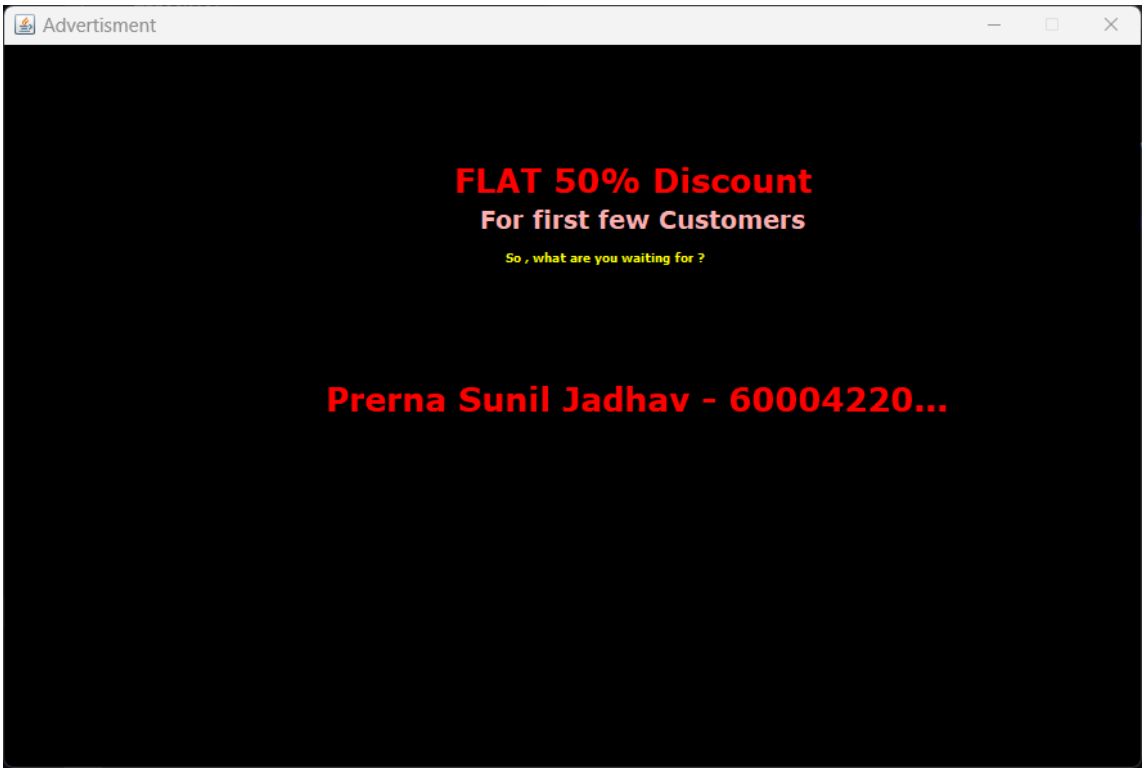
Exp13 > Code2_Banner.java > MyFrame > MyFrame()

```
1   package Exp13;
2   import javax.swing.*;
3   import java.awt.*;
4   class MyFrame extends JFrame implements Runnable {
5       Container c;
6       JLabel title, ad, ad1, name;
7       public MyFrame() {
8           setTitle(title: "Advertisment");
9           setBounds(x: 300, y: 90, width: 900, height: 600);
10          setDefaultCloseOperation(EXIT_ON_CLOSE);
11          setResizable(resizable: false);
```

```
        setVisible(b: true);
        c = getContentPane();
        c.setBackground(Color.black);
        c.setLayout(mgr: null);
        title = new JLabel(text: "SALE SALE SALE !!!");
        title.setSize(width: 300, height: 50);
        title.setLocation(x: 350, y: 30);
        title.setForeground(Color.white);
        title.setFont(new Font(name: "Verdana", Font.BOLD, size: 38));
        c.add(title);
        name = new JLabel(text: "Prerna Sunil Jadhav - 60004220127");
        name.setSize(width: 500, height: 50);
        name.setForeground(Color.red);
        name.setFont(new Font(name: "Verdana", Font.BOLD, size: 25));
        name.setLocation(x: 250, y: 250);
        c.add(name);
        ad = new JLabel(text: "FLAT 50% Discount");
        ad.setSize(width: 400, height: 30);
        ad.setForeground(Color.red);
        ad.setFont(new Font(name: "Verdana", Font.BOLD, size: 25));
        ad.setLocation(x: 350, y: 90);
        c.add(ad);
        ad1 = new JLabel(text: "For first few Customers");
        ad1.setSize(width: 400, height: 30);
        ad1.setForeground(Color.pink);
        ad1.setFont(new Font(name: "Verdana", Font.BOLD, size: 19));
        ad1.setLocation(x: 370, y: 120);
        c.add(ad1);
        ad1 = new JLabel(text: "So , what are you waiting for ?");
        ad1.setSize(width: 400, height: 30);
        ad1.setForeground(Color.yellow);
        ad1.setFont(new Font(name: "Verdana", Font.BOLD, size: 9));
        ad1.setLocation(x: 390, y: 150);
        c.add(ad1);
        new Thread(this).start();
    }

    public void run() {
        try {
            while (true) {
                if (title.getText() == null) {
                    title.setText(text: "SALE SALE SALE !!!");
                    Thread.sleep(millis: 500);
                } else {
                    title.setText(text: null);
                    Thread.sleep(millis: 500);
```

```
57                          Thread.sleep(millis: 500);
58                      }
59                  }
60              } catch (InterruptedException ex) {
61              }
62          }
63      }
64      public class Code2_Banner {
            Run | Debug
65          public static void main(String[] args) {
66              new MyFrame();
67          }
68      }
```

OUTPUT:



CONCLUSION: Thus, we implemented programs on applets.

Name – Prerna Sunil Jadhav **SAP ID - 60004220127**

Experiment No – 14

AIM: Designing Graphical User Interfaces in Java using AWT and Event handling (CO6)

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 (i): Write java program to create a registration form using AWT..

Code1_Registration.java 1

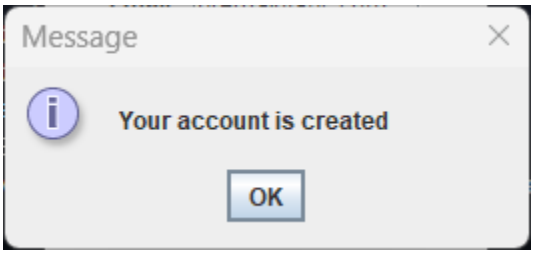
Exp14 > Code1_Registration.java > Code1_Registration > Cod

```
1   package Exp14;
2   import java.awt.*;
3   import javax.swing.*;
4   import java.awt.event.*;
5   class Code1_Registration extends JFrame implements ActionListener {
6       Container c;
7       JLabel lFName, lLName, lPassword, lEmail;
8       JTextField txtFName, txtLName;
9       JTextField txtEmail;
10      JPasswordField txtPassword;
11      JButton btnSubmit, btnClear, btnExit;
12      String strFName, strLName, strPassword, strEmail;
13      Code1_Registration() {
14          c = getContentPane();
15          c.setLayout(new FlowLayout());
16          lFName = new JLabel(text: "First Name: ");
17          lLName = new JLabel(text: "Last Name: ");
18          lPassword = new JLabel(text: "Password: ");
19          lEmail = new JLabel(text: "Email: ");
20          txtFName = new JTextField(columns: 10);
21          txtLName = new JTextField(columns: 10);
22          txtEmail = new JTextField(columns: 10);
23          txtPassword = new JPasswordField(columns: 10);
24          txtPassword.setEchoChar(c: '*');
25          btnSubmit = new JButton(text: "Submit");
26          btnClear = new JButton(text: "Clear");
27          btnExit = new JButton(text: "Exit");
28          c.add(lFName);
29          c.add(txtFName);
30          c.add(lLName);
31          c.add(txtLName);
```

Exp14 > Code1_Registration.java > Code1_Registration > actionPerformed(Act

```
        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(anObject: "") || strLName.equals(anObject: "") ||
                    strPassword.equals(anObject: "") ||
                    strEmail.equals(anObject: "")) {
                JOptionPane.showMessageDialog(c, message: "Input field is empty!!");
                txtFName.requestFocus();
            } else {
                JOptionPane.showMessageDialog(c, message: "Your account is created ");
                txtFName.setText(t: "");
                txtPassword.setText(t: "");
                System.exit(status: 0);
            }
        } else if (ae.getSource() == btnClear) {
            txtFName.setText(t: "");
            txtPassword.setText(t: "");
            txtLName.setText(t: "");
            txtEmail.setText(t: "");
            txtFName.requestFocus();
        } else {

            System.exit(status: 0);
        }
    }
    Run | Debug
    public static void main(String z[]) {
        //Prerna Sunil Jadhav - 60004221027
        Code1_Registration frm = new Code1_Registration();
        frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frm.setBounds(x: 200, y: 200, width: 400, height: 300);
        frm.setVisible(b: true);
        frm.setTitle(title: "REGISTER");
    }
}
```

OUTPUT:



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 (ii): 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

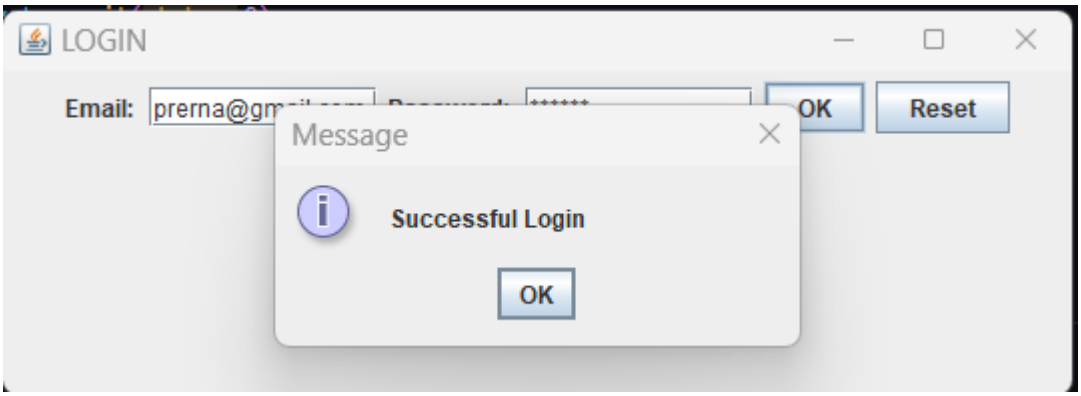
Code2_Login.java 1

Exp14 > Code2_Login.java > Code2_Login > actionPerfor

```java
package Exp14;

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

class Code2_Login extends JFrame implements ActionListener {
    Container c;
    JLabel lPassword, lEmail;
    JTextField txtEmail;
    JPasswordField txtPassword;
    JButton btnSubmit, btnClear;
    String strPassword, strEmail;

    Code2_Login() {
        c = getContentPane();
        c.setLayout(new FlowLayout());
        // labels
        lPassword = new JLabel(text: "Password: ");
        lEmail = new JLabel(text: "Email: ");
        // text fields
        txtEmail = new JTextField(columns: 10);
        txtPassword = new JPasswordField(columns: 10);
        txtPassword.setEchoChar(c: '*');
        // buttons
        btnSubmit = new JButton(text: "OK");
        btnClear = new JButton(text: "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(anObject: "Prerna") &&
                    strEmail.equals(anObject: "prerna@gmail.com")) {
                JOptionPane.showMessageDialog(c, message: "Successful Login");
                System.exit(status: 0);
            } else {
                JOptionPane.showMessageDialog(c, message: "Unsuccessful Login");
                txtEmail.setText(t: "");
                txtPassword.setText(t: "");
            }
```

```
51          } else if (ae.getSource() == btnClear) {
52              txtPassword.setText(t: "");
53              txtEmail.setText(t: "");
54          }
55      }
56
        Run | Debug
57      public static void main(String z[]) {
58          Code2_Login frm = new Code2_Login();
59          frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
60          frm.setBounds(x: 200, y: 200, width: 550, height: 200);
61          frm.setVisible(b: true);
62          frm.setTitle(title: "LOGIN");
63      }
64  }
```

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 – Prerna Sunil Jadhav **SAP ID - 60004220127**

Experiment No – 15

AIM: Develop simple swing applications and complex GUI using Java Swing classes. (CO6)

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 (i): 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.

Code1_Register.java

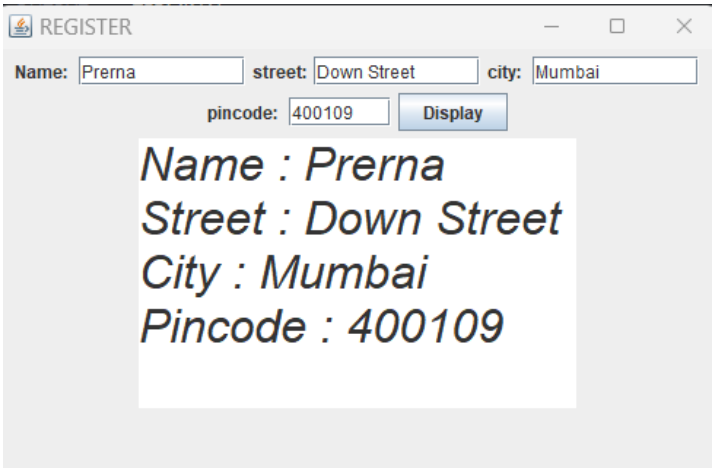
Exp15 > Code1_Register.java > Code1_Register > actionPerformed

```
1   package Exp15;
2
3   import java.awt.*;
4   import javax.swing.*;
5   import java.awt.event.*;
6
7   class Code1_Register extends JFrame implements ActionListener {
8       Container c;
9       JLabel name, street, city, pincode;
10      JTextField tname, tstreet, tcity, tpincode;
11      JTextArea tout;
12      JButton MyInfo, btnClear, btnExit;
13      String sname, sstreet, scity, spincode;
14
15      Code1_Register() {
16          c = getContentPane();
17          c.setLayout(new FlowLayout());
18          // label
19          name = new JLabel(text: "Name: ");
20          street = new JLabel(text: "street:");
21          city = new JLabel(text: "city: ");
22          pincode = new JLabel(text: "pincode: ");
```

```
        // text fields
        tname = new JTextField(columns: 10);
        tstreet = new JTextField(columns: 10);
        tcity = new JTextField(columns: 10);
        tpincode = new JTextField(columns: 6);
        // buttons
        MyInfo = new JButton(text: "Display");
        c.add(name);
        c.add(tname);
        c.add(street);
        c.add(tstreet);
        c.add(city);
        c.add(tcity);
        c.add(pincode);
        c.add(tpincode);
        c.add(MyInfo);
        tout = new JTextArea();
        tout.setFont(new Font(name: "Arial", Font.ITALIC, size: 32));
        tout.setSize(width: 300, height: 400);
        tout.setLocation(x: 100, y: 500);
        tout.setLineWrap(wrap: true);
        tout.setEditable(b: false);
        c.add(tout);
        MyInfo.addActionListener(this);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == MyInfo) {
            sname = tname.getText();
            scity = tcity.getText();
            sstreet = tstreet.getText();
            spincode = tpincode.getText();
            if (sname.equals(anObject: "") || sstreet.equals(anObject: "") ||
                    scity.equals(anObject: "") ||
                    spincode.equals(anObject: "")) {
                JOptionPane.showMessageDialog(c, message: "Input field is empty !!");
                tname.requestFocus();
            } else {
                String data = "Name : " + tname.getText() + "\n" +
                        "Street : " + tstreet.getText() + "\n" + "City : "
                        + tcity.getText() + "\n" + "Pincode : " +
                        tpincode.getText() + "\n";
                tout.setText(data);
                tout.setEditable(b: false);
            }
```

```
68          } else if (e.getSource() == btnClear) {
69              tname.setText(t: "");
70              tpincode.setText(t: "");
71              tstreet.setText(t: "");
72              tcity.setText(t: "");
73              tname.requestFocus();
74          } else {
75              System.exit(status: 0);
76          }
77      }
        Run | Debug
78      public static void main(String[] args) {
79          Code1_Register frm = new Code1_Register();
80          frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
81          frm.setBounds(x: 200, y: 200, width: 400, height: 300);
82          frm.setVisible(b: true);
83          frm.setTitle(title: "REGISTER");
84      }
85  }
```

OUTPUT:



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 (ii): 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

Code2_Buttons.java

Exp15 > Code2_Buttons.java > Code2_Buttons > Code2_Buttor

```
1   package Exp15;
2
3   import java.awt.*;
4   import javax.swing.*;
5   import java.awt.event.*;
6
7   public class Code2_Buttons extends JFrame implements ActionListener {
8       Container c;
9       JButton btnHi, btnHowareyou, btnPrerna, btnGladtomeet;
10      JLabel label;
11
12      Code2_Buttons() {
13          c = getContentPane();
14          c.setLayout(mgr: null);
15          btnHi = new JButton(text: "Hi");
16          btnPrerna = new JButton(text: "Prerna");
17          btnGladtomeet = new JButton(text: "Glad to meet");
18          btnHowareyou = new JButton(text: "How are you?");
19          label = new JLabel(text: " ");
20          label.setSize(width: 200, height: 60);
21          btnHi.setLocation(x: 100, y: 50);
22          btnPrerna.setLocation(x: 100, y: 110);
23          btnGladtomeet.setLocation(x: 100, y: 170);
24          btnHowareyou.setLocation(x: 100, y: 230);
25          btnHi.setSize(width: 100, height: 50);
26          btnPrerna.setSize(width: 100, height: 50);
27          btnGladtomeet.setSize(width: 100, height: 50);
28          btnHowareyou.setSize(width: 100, height: 50);
29          c.add(btnHi);
30          c.add(btnPrerna);
31          c.add(btnGladtomeet);
32          c.add(btnHowareyou);
33          c.add(label);
34          btnHi.addActionListener(this);
35          btnPrerna.addActionListener(this);
36          btnGladtomeet.addActionListener(this);
```

```
37          btnHowareyou.addActionListener(this);
38      }
39
        Run | Debug
40      public static void main(String[] args) {
41          Code2_Buttons frm = new Code2_Buttons();
42          frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
43          frm.setBounds(x: 200, y: 200, width: 400, height: 500);
44          frm.setVisible(b: true);
45          frm.setTitle(title: "click me");
46      }
47
48      public void actionPerformed(ActionEvent e) {
49          if (e.getSource() == btnHi) {
50              label.setText(text: "Hi");
51              label.setLocation(x: 220, y: 50);
52          }
53          if (e.getSource() == btnPrerna) {
54              label.setText(text: "Prerna?");
55              label.setLocation(x: 220, y: 110);
56          }
57          if (e.getSource() == btnGladtomeet) {
58              label.setText(text: "Glad to meet");
59              label.setLocation(x: 220, y: 160);
60          }
61          if (e.getSource() == btnHowareyou) {
62              label.setText(text: "How are you");
63              label.setLocation(x: 220, y: 230);
64          }
65      }
66  }
```

OUTPUT: