AMITY UNIVERSITY MADHYA PRADESH

LAB FILE

JAVA PROGRAMMING LAB

CSE 423

SUBMITTED TO	**SUBMITTED BY**
Mr Manish Khule	**Kumkum Goyal**
Assistant Professor	**A60205223088**
ASET, AUMP	**BTech CSE 4B**

AMITY SCHOOL OF ENGINEERING & TECHNOLOGY

AMITY UNIVERSITY MADHYA PRADESH, GWALIOR

2025

S.NO.	List of practical	Page	Date of Practical	Date of Submission	Signature

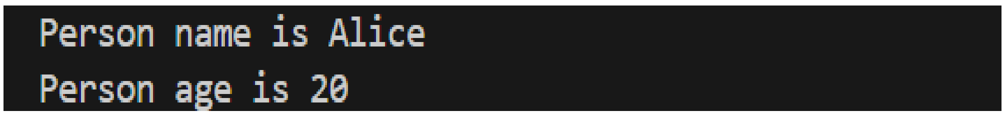
Practical No. 01

Create a Java class Person with attributes and a method to display information. Instantiate and invoke the method.

Code:

```
 public class Person {
  String name;
  int age;
  public void display() {
      System.out.println("Person name is "+ name);
      System.out.println("Person age is "+ age);
  }
  public static void main(String[] args){
      Person p= new Person();
      p.name= "Alice";
      p.age= 20;
      p.display();
  }
}
```

Output:



```
Person name is Alice
Person age is 20
```

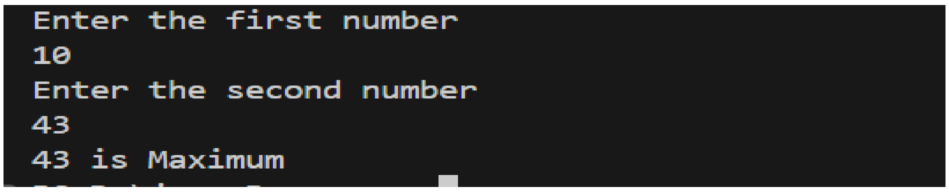
Practical No. 02

Write a program to compare two numbers using if-else and display the maximum.

Code:

```
import java.util.*;
public class CompareNumber {
    public static void main(String[] args){
        Scanner sc= new Scanner(System.in);
        System.out.println("Enter the first number");
        int num1= sc.nextInt();
        System.out.println("Enter the second number");
        int num2= sc.nextInt();

        if( num1 > num2){
            System.out.println(num1 + " is Maximum");
        }
        else if(num2 > num1){
            System.out.println(num2 + " is Maximum");
        }
        else{
            System.out.println("Both numbers are equal");
        }
    }
}
```

Output:



```
Enter the first number
10
Enter the second number
43
43 is Maximum
```

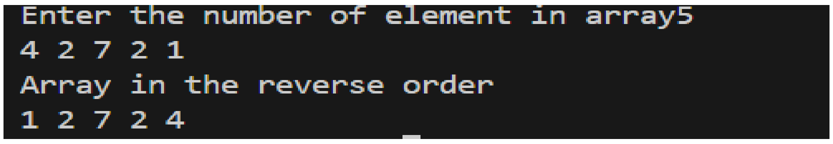
Practical No.03

Write a Java program to read n integers into an array and print them in reverse order.

Code:

```
import java.util.*;
public class ArrayReverse {
  public static void main(String[] args){
  Scanner sc= new Scanner(System.in);
  System.out.print("Enter the number of element in array");
  int n= sc.nextInt();

  int[] arr= new int[n];
  for(int i=0;i<n;i++){
    arr[i]= sc.nextInt();

  }
  System.out.println("Array in the reverse order");
  for (int i= n-1; i>=0;i--){
    System.out.print(arr[i]+ " ");
  }
}
}
```

Output:



```
Enter the number of element in array5
4 2 7 2 1
Array in the reverse order
1 2 7 2 4
```

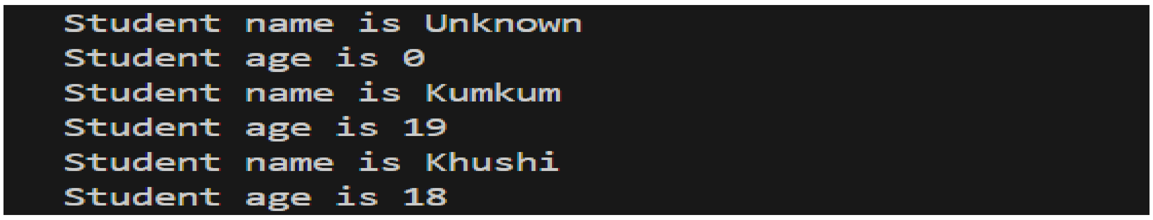
Practical No. 04

Write a program to demonstrate constructor overloading.

Code:

```
public class Student {
    String name;
    int age;
    // Constructor with two parameters
    public Student(String s, int a){
        this.name= s;
        this.age= a;
    }
    // Default constructor
    public Student(){
        name= "Unknown";
        age=0;
    }
    // Constructor with one parameter
    public Student(String n){
        this.name= n;
        age= 18;
    }
    public void display(){
        System.out.println("Student name is " + name);
        System.out.println("Student age is " + age);
    }
    public static void main(String[] args){
        // Using default Constructor
        Student s1= new Student();
        // using constructor with two paramete;
        Student s2= new Student("Kumkum" , 19);
        // USing construtor with one parameter;
        Student s3= new Student("Khushi");
        // Display all the student;

        s1.display();

        s2.display();

        s3.display();

    }

}
```

Output:



```
Student name is Unknown
Student age is 0
Student name is Kumkum
Student age is 19
Student name is Khushi
Student age is 18
```

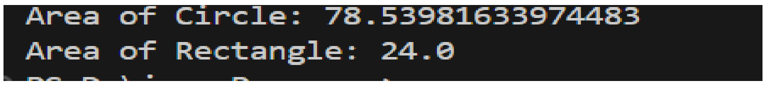
Practical No. 05

Create an abstract class Shape with an abstract method area() and two subclasses: Circle and Rectangle.

Code:

```
abstract class Shape {
    abstract double getarea(){
    }
class Circle extends Shape{
       double radius;
       public Circle(double r){
        this.radius= r;
       }
       double getarea(){
          return Math.PI*radius*radius;
       }
    }
class Rectangle extends Shape{
        double length;
        double width;

        public Rectangle(double l, double w){
            this.length= l;
            this.width= w;
        }
        double getarea(){
            return length*width;
        }
    public static void main(String[] args){
        // Create Circle and Rectangle objects
        Shape circle = new Circle(5.0);
        Shape rectangle = new Rectangle(4.0, 6.0);
        // Call area method

        System.out.println("Area of Circle: " + circle.getarea());

        System.out.println("Area of Rectangle: " + rectangle.getarea());

    }

}
```

Output:



```
Area of Circle: 78.53981633974483
Area of Rectangle: 24.0
```

Practical No. 06

Develop a package mypackage with a class Message that prints a message. Import and use it in another class.

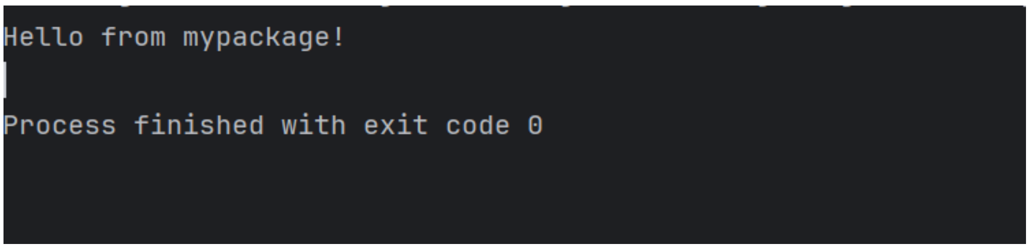
Code:

#File: Message.java

```
package mypackage;

public class Message {
        public void display() {
           System.out.println("Hello from mypackage!");
    }
}
```

# File: TestMessage.java

```
import mypackage.Message;

public class TestMessage {
    public static void main(String[] args) {
        Message msg = new Message();
        msg.display();
    }
}
```



```
Hello from mypackage!

Process finished with exit code 0
```

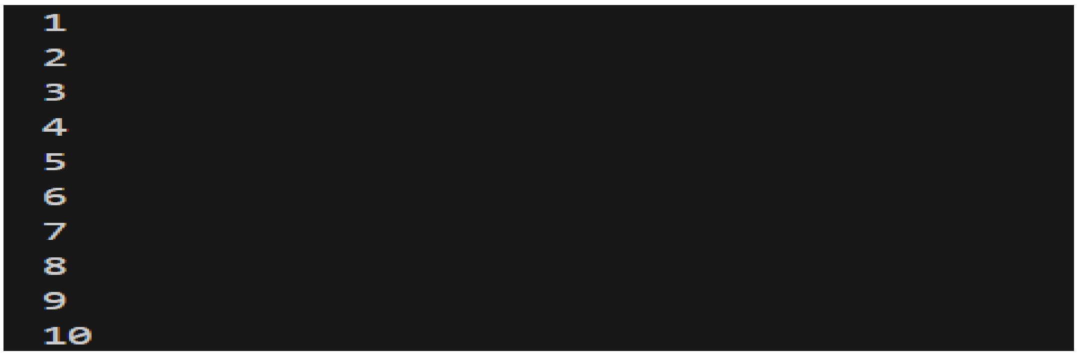
Practical No. 07

Create a class that extends Thread and prints numbers from 1 to 10 with a delay of 500ms between each.

Code:

```
public class ThreaddClass extends Thread {
   public void run(){
      for(int i= 1 ;i <= 10 ;i++){
         System.out.println(i);
      try {
            Thread.sleep(500);
         } catch (InterruptedException e) {
            System.out.println("Thread Interrupted");
         }
      }
   }
public static void main(String[] args){
      ThreaddClass t1= new ThreaddClass();
      t1.start();
   }
}
```

Output:



```
1
2
3
4
5
6
7
8
9
10
```

Practical No. 08

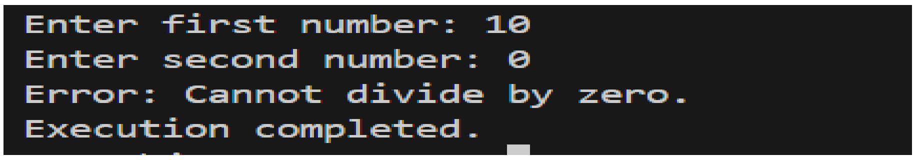
Write a program that demonstrates try-catch-finally using division by zero.

Code:

```
import java.util.Scanner;
public class ExceptionHandlingDemo {
  public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      try {
          // Taking two inputs as strings
          System.out.print("Enter first number: ");
          String input1 = scanner.nextLine();

          System.out.print("Enter second number: ");
          String input2 = scanner.nextLine();

          // Converting string inputs to integers
          int num1 = Integer.parseInt(input1);
          int num2 = Integer.parseInt(input2);

          // Performing division
          int result = num1 / num2;
          System.out.println("Result: " + result);

      } catch (NumberFormatException e) {
          System.out.println("Error: Invalid number format. Please enter valid integers.");
      } catch (ArithmeticException e) {
          System.out.println("Error: Cannot divide by zero.");
      } finally {
          System.out.println("Execution completed.");
      }
      scanner.close();
```

```
    }
}
```

Output:



```
Enter first number: 10
Enter second number: 0
Error: Cannot divide by zero.
Execution completed.
```

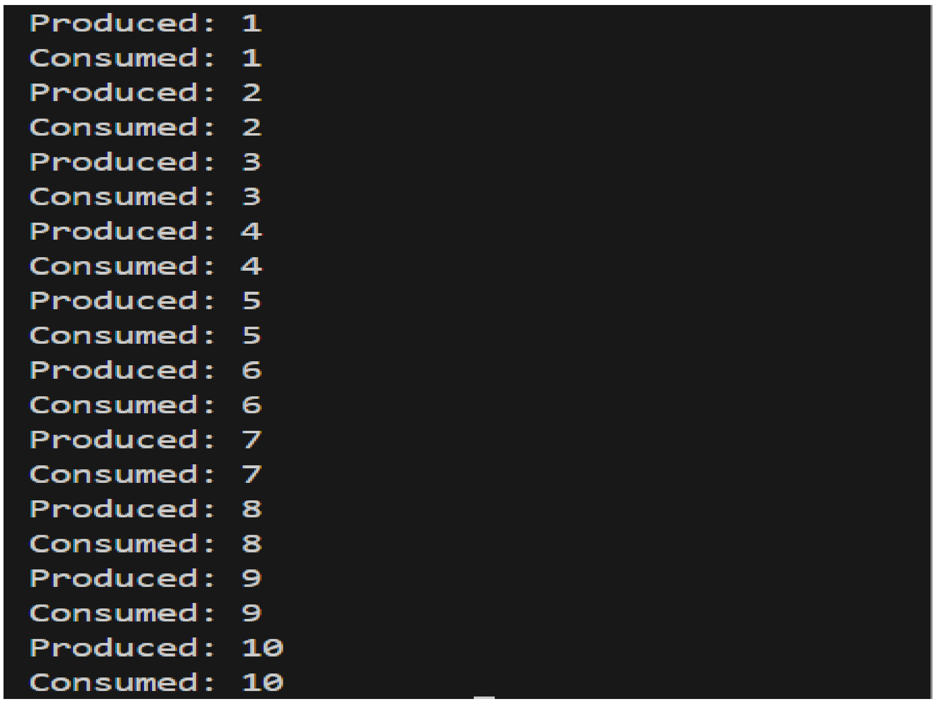
Practical No. 09

Simulate inter-thread communication using wait() and notify() methods in a producer-consumer example.

Code:

```
class Drop {
    private int contents;
    private boolean available = false;
    public synchronized void put(int value) {
        while (available) {
            try {
                wait();
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                System.out.println("Producer interrupted");
            }
        }
        contents = value;
        available = true;
        System.out.println("Produced: " + value);
        notify();
    }

    public synchronized int take() {
        while (!available) {
            try {
                wait();
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                System.out.println("Consumer interrupted");
            }
        }
        available = false;
        System.out.println("Consumed: " + contents);
        notify();
        return contents;
    }
}

// Producer thread
class Producer extends Thread {
    private Drop drop;

    public Producer(Drop drop) {
        this.drop = drop;
    }

    public void run() {
        for (int i = 1; i <= 10; i++) {
            drop.put(i);
            try {
                Thread.sleep(200); // simulate work
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
    }
}

// Consumer thread
class Consumer extends Thread {
    private Drop drop;

    public Consumer(Drop drop) {
        this.drop = drop;
    }

    public void run() {
        for (int i = 1; i <= 10; i++) {
            int value = drop.take();
            // process value...
            try {
                Thread.sleep(300); // simulate work
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
    }
}

public class ProducerConsumerDemo {
    public static void main(String[] args) {
        Drop drop = new Drop();
        new Producer(drop).start();
        new Consumer(drop).start();
    }
}
```

Output:



```
Produced:  1
Consumed:  1
Produced:  2
Consumed:  2
Produced:  3
Consumed:  3
Produced:  4
Consumed:  4
Produced:  5
Consumed:  5
Produced:  6
Consumed:  6
Produced:  7
Consumed:  7
Produced:  8
Consumed:  8
Produced:  9
Consumed:  9
Produced:  10
Consumed:  10
```

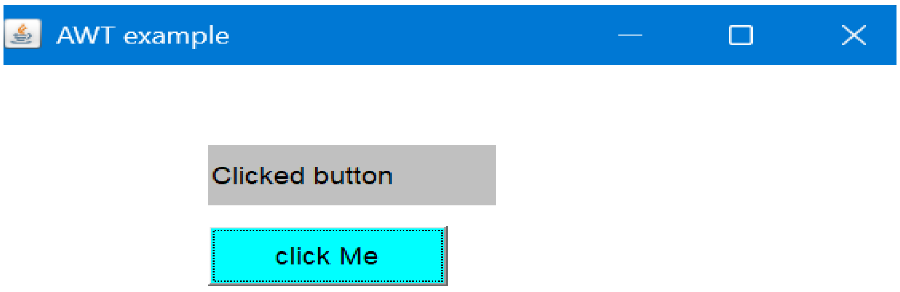
Practical No. 10

Create a basic AWT application with a frame containing a button and a label.

Code:

```
import java.awt.*;
import java.awt.event.*;
public class MyAwtApp extends Frame implements ActionListener{
  Button b;
  Label l;
   public MyAwtApp(){
     setTitle("AWT example");
     setVisible(true);
     setSize(400, 300);
     setLayout(null);

     b= new Button("click Me");
     b.setBounds(100, 110, 100, 30);
     b.setBackground(Color.CYAN);
     add(b);

     l= new Label("click the button");
     l.setBounds(100, 70, 120, 30);
     l.setBackground(Color.LIGHT_GRAY);
     add(l);

     b.addActionListener(this);

     addWindowListener(new WindowAdapter() {
       public void windowClosing(WindowEvent we) {
         dispose();
       }
     });
  }
  public void actionPerformed(ActionEvent e){
     l.setText("Clicked button");
  }
 public static void main(String[] args){
   new MyAwtApp();
 }
  }
```

Output:



Practical No. 11

Design a Swing application with a form containing text fields for name, email, and a submit button.

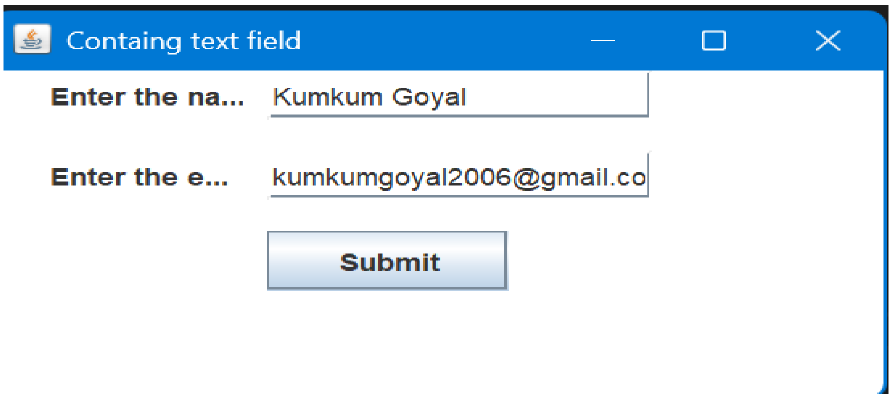
Code:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
public class SwingForm extends Frame implements ActionListener{
  JTextField Namefield , Emailfield;
  JLabel Namelabel, Emaillabel;
  JButton submitButton;

  public SwingForm(){
    setTitle("Containg text field");
    setSize(300, 200);
    setLayout(null);



    Namelabel = new JLabel("Enter the name");
    Namelabel.setBounds(30, 30, 80, 25);
    add(Namelabel);

    Namefield= new JTextField();
    Namefield.setBounds(120, 30, 160, 25);
    add(Namefield);

    Emaillabel = new JLabel("Enter the email");
    Emaillabel.setBounds(30, 70, 80, 25);
```

```
        add(Emaillabel);

        Emailfield= new JTextField();
        Emailfield.setBounds(120, 70, 160, 25);
        add(Emailfield);

        submitButton = new JButton("Submit");
        submitButton.setBounds(120, 110, 100, 30);
        add(submitButton);
        submitButton.addActionListener(this);

        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we) {
                dispose();
            }
        });
        setVisible(true);
    }
public void actionPerformed(ActionEvent e) {
        String name = Namefield.getText();
        String email = Emailfield.getText();

        if (!name.isEmpty() && !email.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Submitted: " + name + ", " + email);
        } else {
            JOptionPane.showMessageDialog(this, "Please fill all fields.", "Warning",
JOptionPane.WARNING_MESSAGE);
        }
    }

    public static void main(String[] args) {
        new SwingForm();
    }
```

```
}
```

Output:


Containg text field

Enter the na... Kumkum Goyal

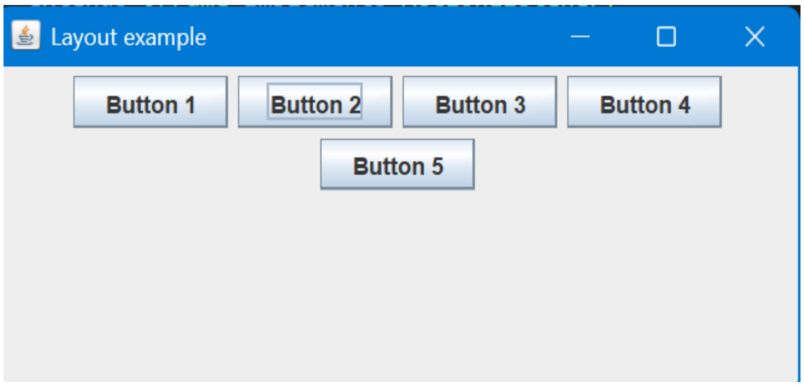
Enter the e... kumkumgoyal2006@gmail.co

Practical No. 12

Implement a Java Swing GUI with multiple buttons and layout managers (e.g., GridLayout or FlowLayout).

Code:

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.*;

public class LayoutExample  extends JFrame implements ActionListener{

   JButton b1, b2 , b3 , b4,  b5;

  public LayoutExample(){

  setTitle("Layout example");

  setLayout(new FlowLayout());

  setSize(300, 200);


  b1= new JButton("Button 1");

  b2= new JButton("Button 2");

  b3= new JButton("Button 3");

  b4= new JButton("Button 4");

  b5= new JButton("Button 5");


  add(b1);

  add(b2);

  add(b3);

  add(b4);

  add(b5);


  b1.addActionListener(this);

  b2.addActionListener(this);

  b3.addActionListener(this);

  b4.addActionListener(this);

  setVisible(true);
}
public void actionPerformed(ActionEvent e) {
        JButton clicked = (JButton) e.getSource();
        JOptionPane.showMessageDialog(this, clicked.getText() + " was clicked!");
    }
 public static void main(String[] args){
    new LayoutExample();
 }
}
```



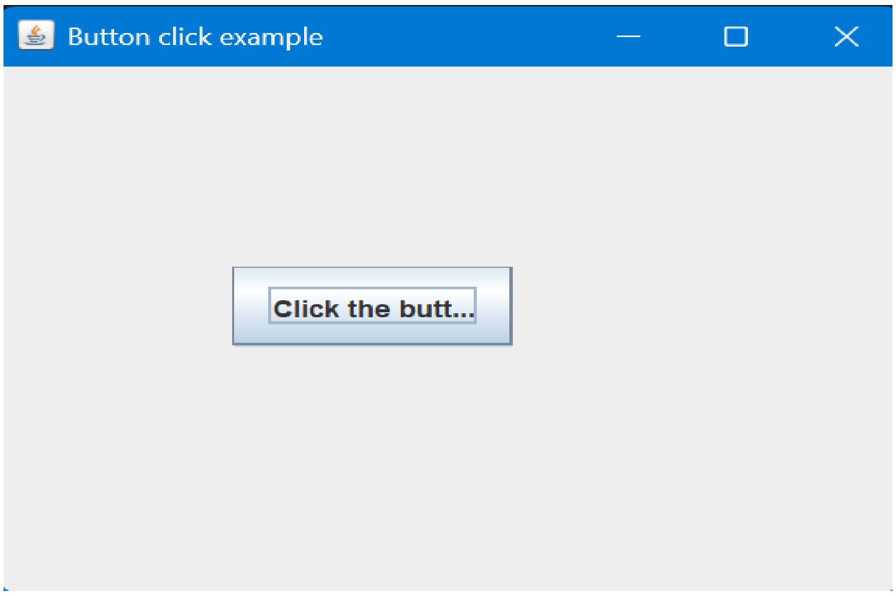
Practical No. 13

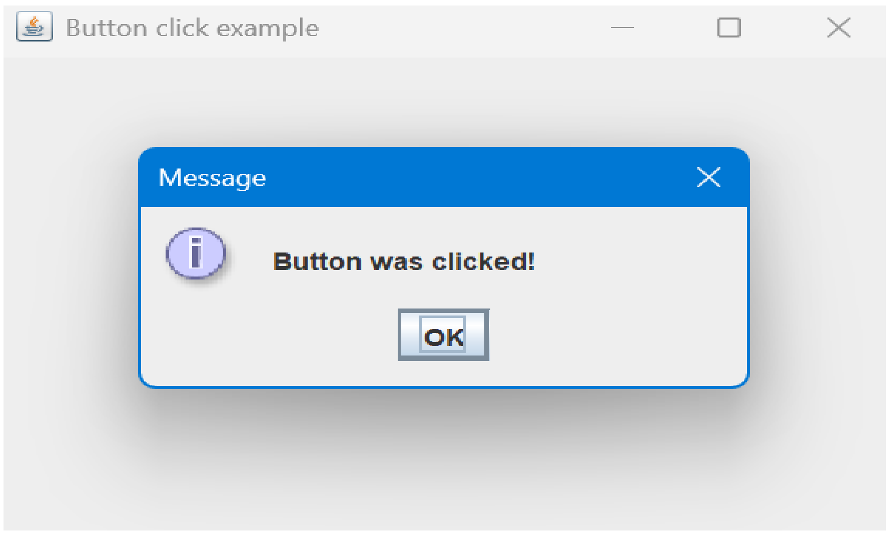
Develop a Java application using ActionListener to display a message when a button is clicked.

Code:

```
import javax.swing.*;
import java.awt.event.*;
public class ButtonClickExample implements ActionListener{
  JFrame frame;
  JButton button;
  public ButtonClickExample(){
    frame= new JFrame("Button click example");
    frame.setSize(400, 300);
    frame.setLayout(null);

    button= new JButton("Click the button");
    button.setBounds(100,100,120,40);
    button.addActionListener(this);
    frame.add(button);
    frame.setVisible(true);
  }

      public void actionPerformed(ActionEvent e){
        JOptionPane.showMessageDialog(frame, "Button was clicked!");

      }
    public static void main(String[] args){
      new ButtonClickExample();
    }
  }
```

Output:





Button was clicked!

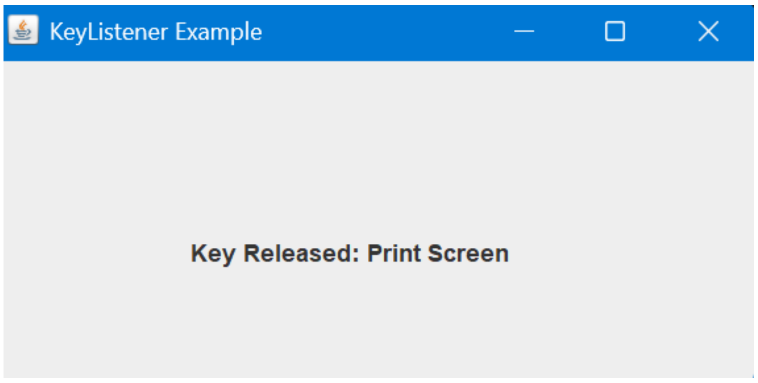
Practical No. 14

Write a program to detect and print key presses using KeyListener.

Code:

```
import javax.swing.*;
import java.awt.event.*;
public class KeyPressExample extends JFrame implements KeyListener {
    private final JLabel label;
    public KeyPressExample() {
        super("KeyListener Example");
        setSize(400, 200);
        setLayout(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);

        label = new JLabel("Press any key...");
        label.setBounds(100, 80, 200, 30);
        add(label);

        addKeyListener(this);
        setFocusable(true);
        setVisible(true);

        // Ensure the frame actually has focus for key events
        requestFocusInWindow();
    }
    public void keyTyped(KeyEvent e) {
        label.setText("Key Typed: " + e.getKeyChar());
    }
    public void keyPressed(KeyEvent e) {
        label.setText("Key Pressed: " + KeyEvent.getKeyText(e.getKeyCode()));
    }
    public void keyReleased(KeyEvent e) {
        label.setText("Key Released: " + KeyEvent.getKeyText(e.getKeyCode()));
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(KeyPressExample::new);
    }
}
```

Output:



KeyListener Example

```
Key Released: Print Screen
```

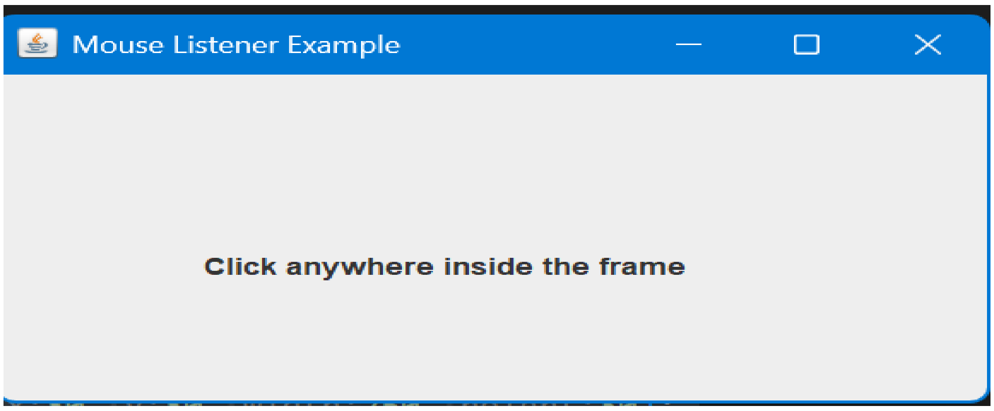
Practical No. 15

Create a mouse event handling example using MouseListener to change label text on mouse click.

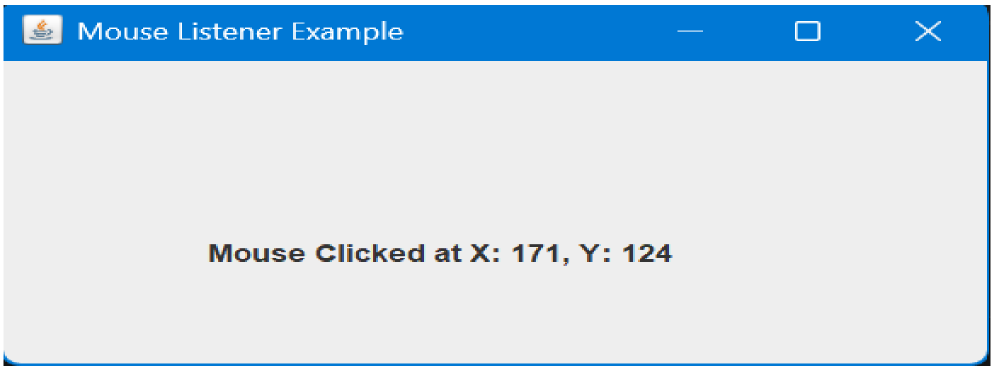
Code:

```
import javax.swing.*;
import java.awt.event.*;
public class MouseClick extends JFrame implements MouseListener {
  JLabel label;
  public MouseClick() {
    setTitle("MouseListener Example");
    setSize(400, 200);
    setLayout(null);
    setDefaultCloseOperation(EXIT_ON_CLOSE);

    label = new JLabel("Click anywhere inside the frame");
    label.setBounds(80, 80, 250, 30);
    add(label);

    // Add MouseListener to the JFrame
    addMouseListener(this);
    setVisible(true);
  }
  // Triggered when mouse is clicked (pressed and released)
  public void mouseClicked(MouseEvent e) {
    label.setText("Mouse Clicked at (" + e.getX() + ", " + e.getY() + ")");
  }
  // The rest are required to be overridden but can be left empty
  public void mousePressed(MouseEvent e) {}
  public void mouseReleased(MouseEvent e) {}
  public void mouseEntered(MouseEvent e) {}
  public void mouseExited(MouseEvent e) {}
  public static void main(String[] args) {
        SwingUtilities.invokeLater(MouseClickExample::new);
    }
}
```

Output:



```
Click anywhere inside the frame
```



```
Mouse Clicked at X: 171, Y: 124
```