MADAN BHANDARI
MEMORIAL COLLEGE

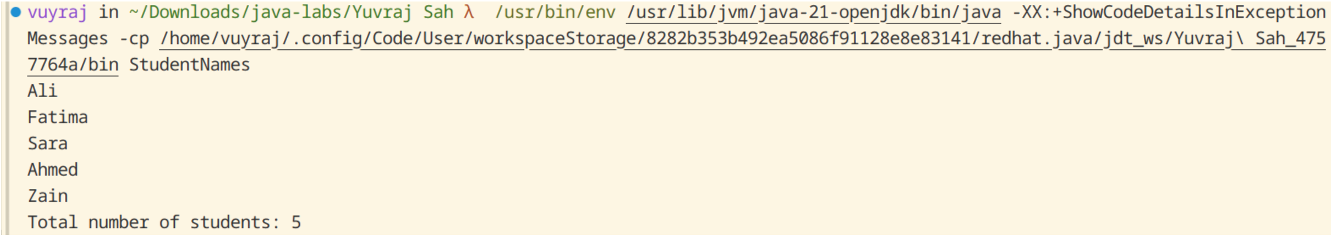
## Lab 1: Java Basics

**Question 1**: Write a Java program that stores the following list of student names in a single-dimensional array: `"Ali", "Fatima", "Sara", "Ahmed", "Zain".`

**Code:**

```
public class StudentNames {
    public static void main(String[] args) {
        String[] students = { "Ali", "Fatima", "Sara", "Ahmed", "Zain" };
        for (String student : students) {
            System.out.println(student);
        }
        System.out.println("Total number of students: " + students.length);
    }
}
```

**Output:**



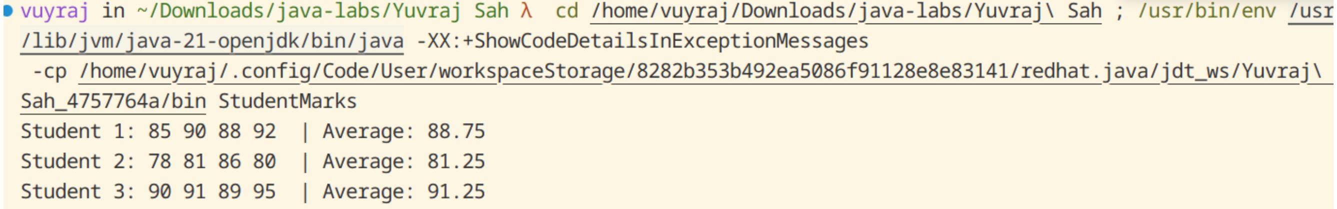
```
● vuyraj in ~/Downloads/java-labs/Yuvraj Sah λ  /usr/bin/env /usr/lib/jvm/java-21-openjdk/bin/java -XX:+ShowCodeDetailsInException
Messages -cp /home/vuyraj/.config/Code/User/workspaceStorage/8282b353b492ea5086f91128e8e83141/redhat.java/jdt_ws/Yuvraj\ Sah_475
7764a/bin StudentNames
Ali
Fatima
Sara
Ahmed
Zain
Total number of students: 5
```

**Question 2:** Write a Java program to create and display a 2D array that represents the marks of 3 students in 4 subjects.

**Code:**

```
 public class StudentMarks {
    public static void main(String[] args) {
        int[][] marks = {{85, 90, 88, 92}, {78, 81, 86, 80},{90, 91, 89, 95}};
        int studentNumber = 1;
        for (int[] studentMarks : marks) {
            System.out.print("Student " + studentNumber + ": ");
            int total = 0;
            for (int mark : studentMarks) {
                System.out.print(mark + " ");
                total += mark;
            }
            double average = (double) total / studentMarks.length;
            System.out.println(" | Average: " + average);
            studentNumber++;
        }  }  }
```

**Output:**



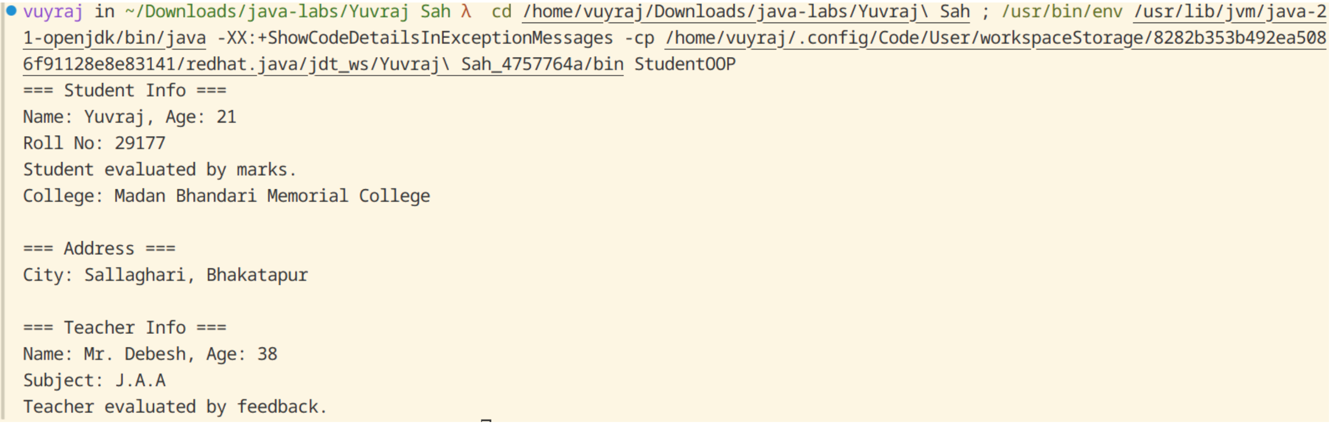
```
vuyraj in ~/Downloads/java-labs/Yuvraj Sah λ  cd /home/vuyraj/Downloads/java-labs/Yuvraj\ Sah ; /usr/bin/env /usr
/lib/jvm/java-21-openjdk/bin/java -XX:+ShowCodeDetailsInExceptionMessages
 -cp /home/vuyraj/.config/Code/User/workspaceStorage/8282b353b492ea5086f91128e8e83141/redhat.java/jdt_ws/Yuvraj\ 
Sah_4757764a/bin StudentMarks
Student 1: 85 90 88 92  | Average: 88.75
Student 2: 78 81 86 80  | Average: 81.25
Student 3: 90 91 89 95  | Average: 91.25
```

**Question 3:** Write a Java Program to demonstrate OOP concepts.

**Code:**

```
interface Evaluatable {
    void evaluate();
}
class Person {
    protected String name;
    protected int age;
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
    public void displayInfo() {
        System.out.println("Name: " + name + ", Age: " + age);
    }
}
class Student extends Person implements Evaluatable {
    final int rollNumber;
    static String college = "Madan Bhandari Memorial College";
    public Student(String name, int age, int rollNumber) {
        super(name, age);
        this.rollNumber = rollNumber;
    }
    @Override
    public void displayInfo() {
        super.displayInfo();
        System.out.println("Roll No: " + rollNumber);
    }
    public void evaluate() {
        System.out.println("Student evaluated by marks.");
    }
    static void printCollege() {
        System.out.println("College: " + college);
    }
    class Address {
        String city;
        Address(String city) {
            this.city = city;
        }
        void show() {
            System.out.println("City: " + city);
        }
    } }
class Teacher extends Person implements Evaluatable {
    String subject;
    public Teacher(String name, int age, String subject) {
        super(name, age);
        this.subject = subject;
    }
    @Override
    public void displayInfo() {
        super.displayInfo();
        System.out.println("Subject: " + subject);
    }
    public void evaluate() {
        System.out.println("Teacher evaluated by feedback.");
    }}
public class StudentOOP {
    public static void main(String[] args) {
        Student s = new Student("Yuvraj", 21, 29177);
        Teacher t = new Teacher("Mr. Debesh", 38, "J.A.A");
        System.out.println("=== Student Info ===");
        s.displayInfo();
        s.evaluate();
        Student.printCollege();
        System.out.println("\n=== Address ===");
        Student.Address addr = s.new Address("Sallaghari, Bhakatapur");
        addr.show();
        System.out.println("\n=== Teacher Info ===");
        t.displayInfo();
        t.evaluate();
    } }
```

**Output:**



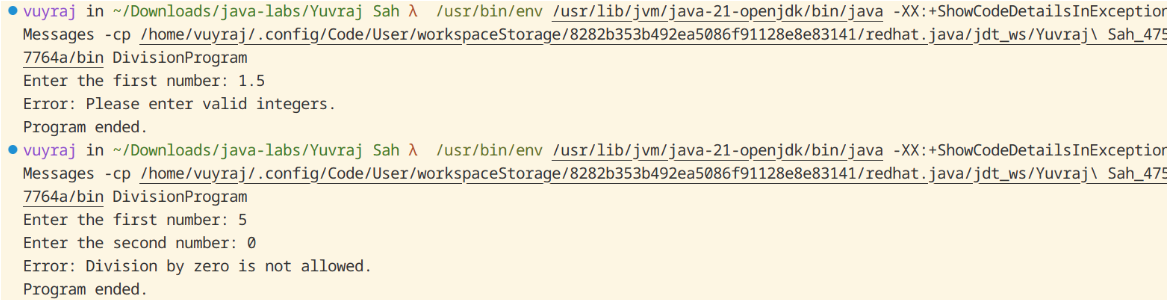
```
● vuyraj in ~/Downloads/java-labs/Yuvraj Sah λ  cd /home/vuyraj/Downloads/java-labs/Yuvraj\ Sah ; /usr/bin/env /usr/lib/jvm/java-21-openjdk/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /home/vuyraj/.config/Code/User/workspaceStorage/8282b353b492ea5086f91128e8e83141/redhat.java/jdt_ws/Yuvraj\ Sah_4757764a/bin StudentOOP
=== Student Info ===
Name: Yuvraj, Age: 21
Roll No: 29177
Student evaluated by marks.
College: Madan Bhandari Memorial College

=== Address ===
City: Sallaghari, Bhakatapur

=== Teacher Info ===
Name: Mr. Debesh, Age: 38
Subject: J.A.A
Teacher evaluated by feedback.
```

**Question 4:** Write a Java program that Divides two digit number and handles the `ArithmeticException & InputMismatchException`.

**Code:**

```
import java.util.Scanner;
public class DivisionProgram {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        try {
            System.out.print("Enter the first number: ");
            int num1 = scanner.nextInt();
            System.out.print("Enter the second number: ");
            int num2 = scanner.nextInt();
            int result = num1 / num2;
            System.out.println("Result: " + result);
        } catch (ArithmeticException e) {
            System.out.println("Error: Division by zero is not allowed.");
        } catch (java.util.InputMismatchException e) {
            System.out.println("Error: Please enter valid integers.");
        }
        System.out.println("Program ended.");
    }
}
```

**Output:**



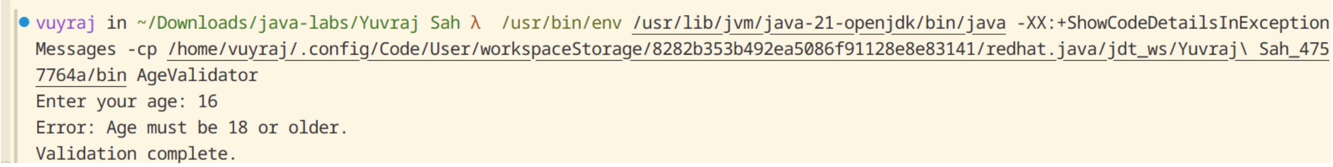
```
● vuyraj in ~/Downloads/java-labs/Yuvraj Sah λ  /usr/bin/env /usr/lib/jvm/java-21-openjdk/bin/java -XX:+ShowCodeDetailsInExceptior
  Messages -cp /home/vuyraj/.config/Code/User/workspaceStorage/8282b353b492ea5086f91128e8e83141/redhat.java/jdt_ws/Yuvraj\ Sah_475
  7764a/bin DivisionProgram
  Enter the first number: 1.5
  Error: Please enter valid integers.
  Program ended.
● vuyraj in ~/Downloads/java-labs/Yuvraj Sah λ  /usr/bin/env /usr/lib/jvm/java-21-openjdk/bin/java -XX:+ShowCodeDetailsInExceptior
  Messages -cp /home/vuyraj/.config/Code/User/workspaceStorage/8282b353b492ea5086f91128e8e83141/redhat.java/jdt_ws/Yuvraj\ Sah_475
  7764a/bin DivisionProgram
  Enter the first number: 5
  Enter the second number: 0
  Error: Division by zero is not allowed.
  Program ended.
```

**Question 5:** Create a custom exception named `InvalidAgeException` that is thrown when the user enters an age below 18.

```
import java.util.Scanner;
class InvalidAgeException extends Exception {
        public InvalidAgeException(String message) {
                super(message);
                } }
public class AgeValidator {
        public static void main(String[] args) {
                Scanner scanner = new Scanner(System.in);
                try {
                        System.out.print("Enter your age: ");
                        int age = scanner.nextInt();
                        if (age < 18) {
```

```
                throw new InvalidAgeException("Age must be 18 or older.");
            }
            System.out.println("Access granted.");
        } catch (InvalidAgeException e) {
            System.out.println("Error: " + e.getMessage());
            }
    System.out.println("Validation complete.");
    }
}
```

**Output:**



```
vuyraj in ~/Downloads/java-labs/Yuvraj Sah λ  /usr/bin/env /usr/lib/jvm/java-21-openjdk/bin/java -XX:+ShowCodeDetailsInException
Messages -cp /home/vuyraj/.config/Code/User/workspaceStorage/8282b353b492ea5086f91128e8e83141/redhat.java/jdt_ws/Yuvraj\ Sah_475
7764a/bin AgeValidator
Enter your age: 16
Error: Age must be 18 or older.
Validation complete.
```

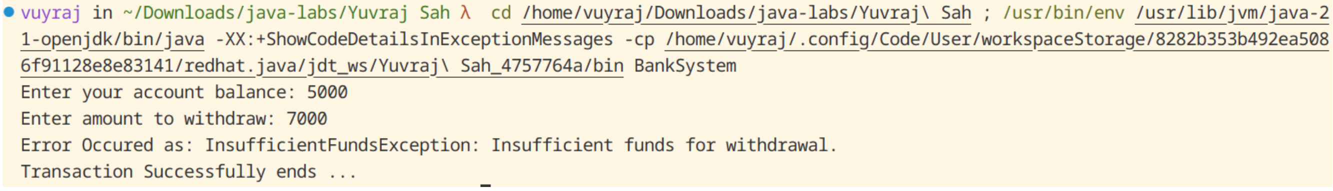
**Question 6:** Write a Java program that simulates a bank withdrawal system with the following behavior:

1. Prompt the user to enter their **account balance** and the **amount to withdraw**.
2. If the withdrawal amount is **greater than the balance**, throw a custom exception `InsufficientFundsException`.
3. Declare the method that performs the withdrawal using `throws`
4. In the `main` method, use `try-catch-finally` to:
    1. Call the withdrawal method
    2. Catch and display any exception messages
5. Always display `"Transaction ended."` in the `finally` block.

**Code:**

```
import java.util.Scanner;
class InsufficientFundsException extends Exception {
    public InsufficientFundsException(String message) {
        super(message);
    }
}

public class BankSystem {
    public static void withdraw(double balance, double amount) throws InsufficientFundsException {
        if (amount > balance) {
            throw new InsufficientFundsException("Insufficient funds for withdrawal.");
        }
        System.out.println("Withdrawal successful. New balance: " + (balance - amount));
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        try {
            System.out.print("Enter your account balance: ");
            double balance = scanner.nextDouble();
            System.out.print("Enter amount to withdraw: ");
            double amount = scanner.nextDouble();
            withdraw(balance, amount);
        } catch (InsufficientFundsException e) {
            System.out.println("Error Occured as: " + e);
        } catch (Exception e) {
            System.out.println("Some Error might have Occured: " + e);
        } finally {
            System.out.println("Transaction Successfully ends ...");
        }
    }
}
```

**Output:**



```
● vuyraj in ~/Downloads/java-labs/Yuvraj Sah λ  cd /home/vuyraj/Downloads/java-labs/Yuvraj\ Sah ; /usr/bin/env /usr/lib/jvm/java-2
  1-openjdk/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /home/vuyraj/.config/Code/User/workspaceStorage/8282b353b492ea508
  6f91128e8e83141/redhat.java/jdt_ws/Yuvraj\ Sah_4757764a/bin BankSystem
  Enter your account balance: 5000
  Enter amount to withdraw: 7000
  Error Occured as: InsufficientFundsException: Insufficient funds for withdrawal.
  Transaction Successfully ends ...
```

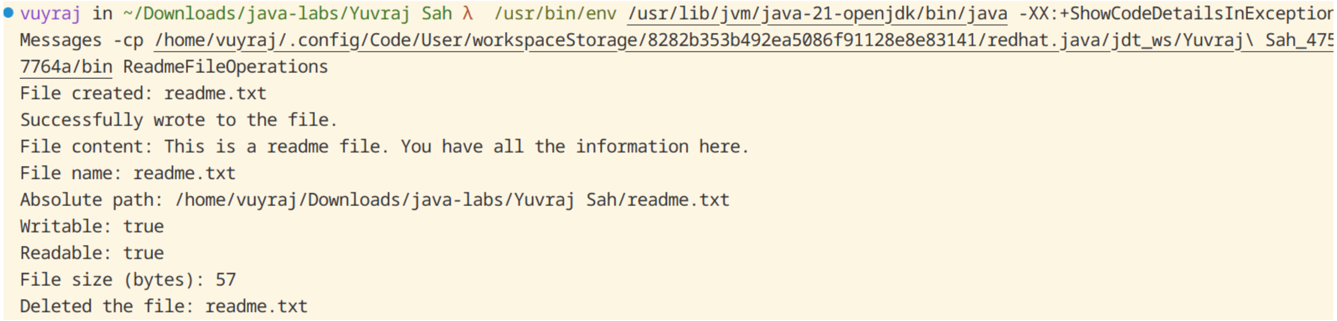
**Question 7:** Write a java program to handle :

1. Create a file named `readme.txt` on your desktop.
2. Write the content `"This is a readme file. You have all the information here."`
3. Read and print the contents of the file.
4. Print the file's absolute path and size in bytes.
5. Delete the file and confirm deletion.

**Code:**

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class ReadmeFileOperations {
    public static void main(String[] args) {
        try {
            File myFile = new File("readme.txt");
            if (myFile.createNewFile()) {
                System.out.println("File created: " + myFile.getName());
            } else {
                System.out.println("File already exists.");
            }
        } catch (IOException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
        }
        try {
            FileWriter writer = new FileWriter("readme.txt");
            writer.write("This is a readme file. You have all the information here.");
            writer.close();
            System.out.println("Successfully wrote to the file.");
        } catch (IOException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
        }
        try {
            File myFile = new File("readme.txt");
            Scanner myReader = new Scanner(myFile);
            while (myReader.hasNextLine()) {
                String data = myReader.nextLine();
                System.out.println("File content: " + data);
            }
            myReader.close();
        } catch (FileNotFoundException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
        }
        File file = new File("readme.txt");
        if (file.exists()) {
            System.out.println("File name: " + file.getName());
            System.out.println("Absolute path: " + file.getAbsolutePath());
            System.out.println("Writable: " + file.canWrite());
            System.out.println("Readable: " + file.canRead());
            System.out.println("File size (bytes): " + file.length());
        } else {
            System.out.println("The file does not exist.");
        }
        if (file.delete()) {
            System.out.println("Deleted the file: " + file.getName());
        } else {
            System.out.println("Failed to delete the file.");
        }
    }
}
```

**Output:**



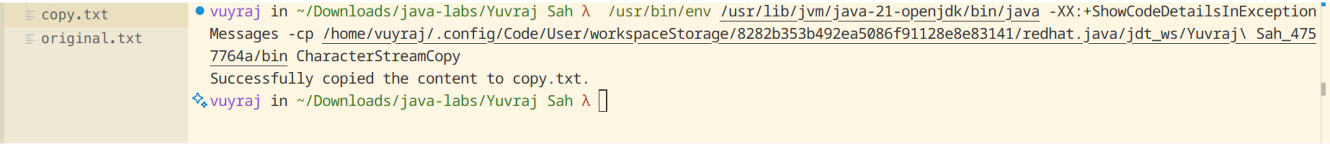
```
● vuyraj in ~/Downloads/java-labs/Yuvraj Sah λ  /usr/bin/env /usr/lib/jvm/java-21-openjdk/bin/java -XX:+ShowCodeDetailsInExceptior
  Messages -cp /home/vuyraj/.config/Code/User/workspaceStorage/8282b353b492ea5086f91128e8e83141/redhat.java/jdt_ws/Yuvraj\ Sah_475
  7764a/bin ReadmeFileOperations
  File created: readme.txt
  Successfully wrote to the file.
  File content: This is a readme file. You have all the information here.
  File name: readme.txt
  Absolute path: /home/vuyraj/Downloads/java-labs/Yuvraj Sah/readme.txt
  Writable: true
  Readable: true
  File size (bytes): 57
  Deleted the file: readme.txt
```

**Question 8:** Write a program to copy text from original.txt file to copy.txt.

**Code:**

```
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
public class CharacterStreamCopy {
    public static void main(String[] args) {
        try {
            File originalFile = new File("original.txt");
            FileWriter writer = new FileWriter(originalFile);
            writer.write("This is a file for copy example.");
            writer.close();
            FileReader reader = new FileReader(originalFile);
            File copyFile = new File("copy.txt");
            FileWriter copyWriter = new FileWriter(copyFile);
            int data;
            while ((data = reader.read()) != -1) {
                copyWriter.write(data);
            }
            reader.close();
            copyWriter.close();
            System.out.println("Successfully copied the content to copy.txt.");
        } catch (IOException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
        }
    }
}
```

**Output:**



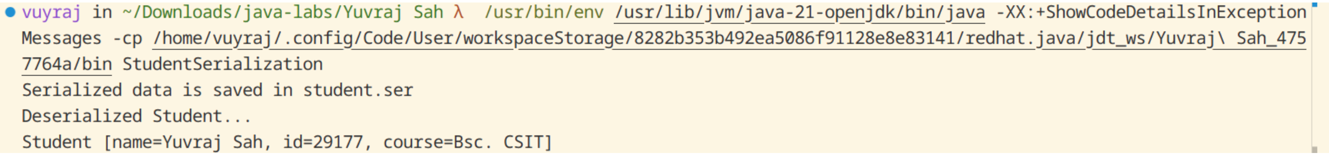
```
● vuyraj in ~/Downloads/java-labs/Yuvraj Sah λ  /usr/bin/env /usr/lib/jvm/java-21-openjdk/bin/java -XX:+ShowCodeDetailsInException
Messages -cp /home/vuyraj/.config/Code/User/workspaceStorage/8282b353b492ea5086f91128e8e83141/redhat.java/jdt_ws/Yuvraj\ Sah_475
7764a/bin CharacterStreamCopy
Successfully copied the content to copy.txt.
vuyraj in ~/Downloads/java-labs/Yuvraj Sah λ
```

**Question 9:** Write a program to Implement Serializtion and Deserialization.

**Code:**

```
import java.io.*;
public class StudentSerialization {
    public static void main(String[] args) {
        class Student implements Serializable {
            private static final long serialVersionUID = 1 L;
            private String name;
            private int id;
            private String course;
        public Student(String name, int id, String course) {
            this.name = name;
            this.id = id;
            this.course = course;
        }
        public String getName() {
            return name;
        }
        public int getId() {
            return id;
        }
        public String getCourse() {
            return course;
        }
        @Override
        public String toString() {
            return "Student [name=" + name + ", id=" + id + ", course=" + course + "]";
        }
    }
    Student student = new Student("Yuvraj Sah", 29177, "Bsc. CSIT");
    try {
        FileOutputStream fileOut = new FileOutputStream("student.ser");
        ObjectOutputStream out = new ObjectOutputStream(fileOut);
        out.writeObject(student);
        out.close();
        fileOut.close();
        System.out.println("Serialized data is saved in student.ser");
    } catch (IOException i) {
        i.printStackTrace();
    }
    Student deserializedStudent = null;
    try {
        FileInputStream fileIn = new FileInputStream("student.ser");
        ObjectInputStream in = new ObjectInputStream(fileIn);
        deserializedStudent = (Student) in .readObject(); in .close();
        fileIn.close();
    } catch (IOException | ClassNotFoundException e) {
        e.printStackTrace();
    }
    if (deserializedStudent != null) {
        System.out.println("Deserialized Student...");
        System.out.println(deserializedStudent);
    }   }}
```

**Output:**



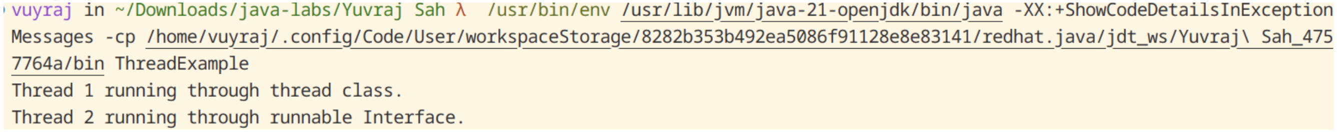
```
● vuyraj in ~/Downloads/java-labs/Yuvraj Sah λ  /usr/bin/env /usr/lib/jvm/java-21-openjdk/bin/java -XX:+ShowCodeDetailsInException
  Messages -cp /home/vuyraj/.config/Code/User/workspaceStorage/8282b353b492ea5086f91128e8e83141/redhat.java/jdt_ws/Yuvraj\ Sah_475
  7764a/bin StudentSerialization
  Serialized data is saved in student.ser
  Deserialized Student...
  Student [name=Yuvraj Sah, id=29177, course=Bsc. CSIT]
```

**Question 10:** Write a Java program that creates two threads, and each thread should print a message.

**Code:**

```
public class ThreadExample {
    static class MyThread extends Thread {
        @Override
        public void run() {
            System.out.println("Thread 1 running through thread class.");
        }
    }
    static class MyRunnable implements Runnable {
        @Override
        public void run() {

            System.out.println("Thread 2 running through runnable Interface.");
        }
    }
    public static void main(String[] args) {
        MyThread thread1 = new MyThread();
        thread1.start();
        MyRunnable myRunnable = new MyRunnable();
        Thread thread2 = new Thread(myRunnable);
        thread2.start();
    }
}
```

**Output:**



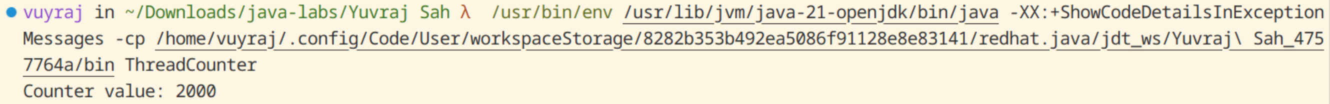
```
vuyraj in ~/Downloads/java-labs/Yuvraj Sah λ  /usr/bin/env /usr/lib/jvm/java-21-openjdk/bin/java -XX:+ShowCodeDetailsInException
Messages -cp /home/vuyraj/.config/Code/User/workspaceStorage/8282b353b492ea5086f91128e8e83141/redhat.java/jdt_ws/Yuvraj\ Sah_475
7764a/bin ThreadExample
Thread 1 running through thread class.
Thread 2 running through runnable Interface.
```

**Question 11:** Create a Java program with two threads that both increment a shared counter. Each thread should increment the counter 1000 times.

**Code:**

```
public class ThreadCounter {
    private static int counter = 0;
    public synchronized static void incrementCounter() {
        counter++;
    }
    public static void main(String[] args) {
        Runnable task = () - > {
            for (int i = 0; i < 1000; i++) {
                incrementCounter();
            }
        };
        Thread thread1 = new Thread(task);
        Thread thread2 = new Thread(task);
        thread1.start();
        thread2.start();
        try {
            thread1.join();
            thread2.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("Counter value: " + counter);
    }
}
```

**Output:**



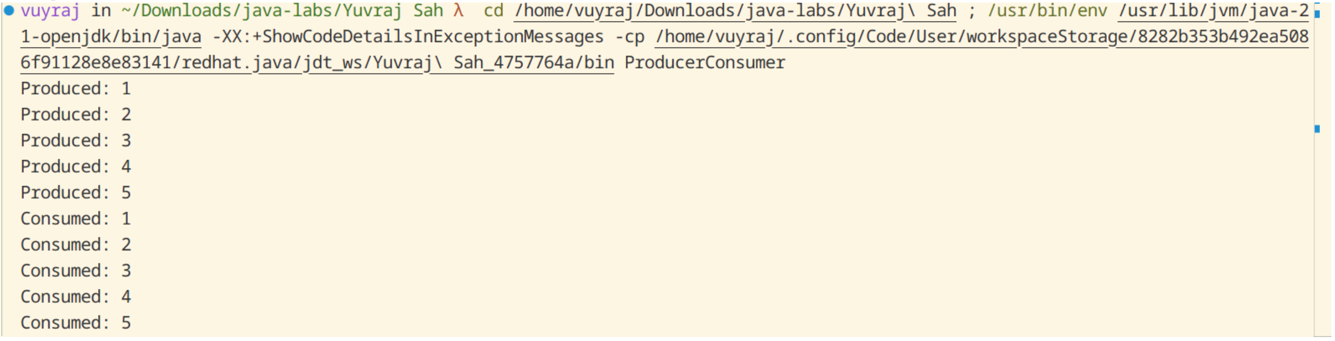
```
● vuyraj in ~/Downloads/java-labs/Yuvraj Sah λ  /usr/bin/env /usr/lib/jvm/java-21-openjdk/bin/java -XX:+ShowCodeDetailsInException
  Messages -cp /home/vuyraj/.config/Code/User/workspaceStorage/8282b353b492ea5086f91128e8e83141/redhat.java/jdt_ws/Yuvraj\ Sah_475
  7764a/bin ThreadCounter
  Counter value: 2000
```

**Question 12:** Write a program to implement `wait()` and `notify()` for inter-thread communication.

**Code:**

```
import java.util.LinkedList;
import java.util.Queue;
public class ProducerConsumer {
    private static Queue < Integer > buffer = new LinkedList < > ();
    private static final int BUFFER_SIZE = 5;
    static class Producer implements Runnable {
        @Override
        public void run() {
            try {
                for (int i = 1; i <= 5; i++) {
                    synchronized(buffer) {
                        while (buffer.size() == BUFFER_SIZE) {
                            buffer.wait();
                        }
                        buffer.add(i);
                        System.out.println("Produced: " + i);
                        buffer.notify();
                    }
                }
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
    static class Consumer implements Runnable {
        @Override
        public void run() {
            try {
                for (int i = 1; i <= 5; i++) {
                    synchronized(buffer) {
                        while (buffer.isEmpty()) {
                            buffer.wait();
                        }
                        int item = buffer.poll();
                        System.out.println("Consumed: " + item);
                        buffer.notify();
                    }
                }
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
    public static void main(String[] args) {
        Thread producerThread = new Thread(new Producer());
        Thread consumerThread = new Thread(new Consumer());
        producerThread.start();
        consumerThread.start();
    }
}
```

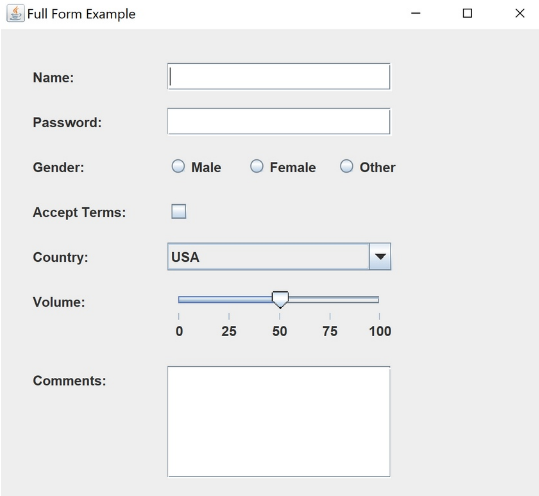
**Output:**



```
● vuyraj in ~/Downloads/java-labs/Yuvraj Sah λ  cd /home/vuyraj/Downloads/java-labs/Yuvraj\ Sah ; /usr/bin/env /usr/lib/jvm/java-2
  1-openjdk/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /home/vuyraj/.config/Code/User/workspaceStorage/8282b353b492ea508
  6f91128e8e83141/redhat.java/jdt_ws/Yuvraj\ Sah_4757764a/bin ProducerConsumer
  Produced: 1
  Produced: 2
  Produced: 3
  Produced: 4
  Produced: 5
  Consumed: 1
  Consumed: 2
  Consumed: 3
  Consumed: 4
  Consumed: 5
```

**Lab 2: Java Swing**

**Question 1:** Add the following components, and structure it like a form (as given in the image below) by manually placing them in the frame: *text field, password field, radio button, checkbox, combo box, slider, text area, and scroll pane.*



Full Form Example

Name:

Password:

Gender: ○ Male ○ Female ○ Other

Accept Terms: □

Country: USA

Volume:

Comments:

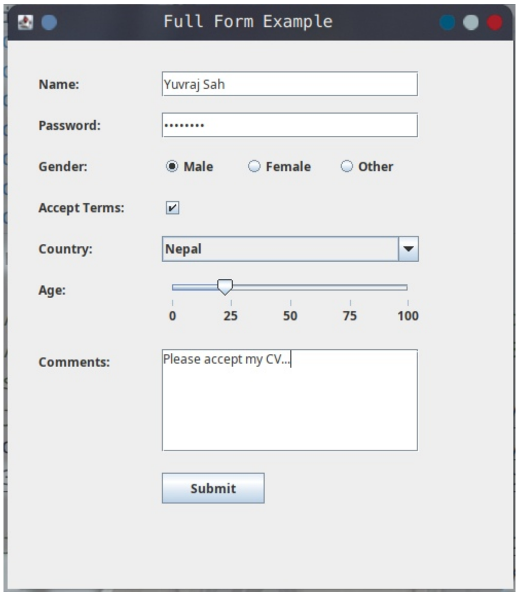
**Code:**

```
import javax.swing.*;
import java.awt.*;
public class FullForm {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Full Form Example");
        frame.setSize(450, 500);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(null);

       JLabel nameLabel = new JLabel("Name:");
        nameLabel.setBounds(30, 30, 100, 25);
        JTextField nameField = new JTextField();
        nameField.setBounds(150, 30, 250, 25);

        JLabel passLabel = new JLabel("Password:");
        passLabel.setBounds(30, 70, 100, 25);
        JPasswordField passField = new JPasswordField();
        passField.setBounds(150, 70, 250, 25);

        JLabel genderLabel = new JLabel("Gender:");
        genderLabel.setBounds(30, 110, 100, 25);
        JRadioButton male = new JRadioButton("Male");
        male.setBounds(150, 110, 70, 25);
```

```
 JRadioButton female = new JRadioButton("Female");
 female.setBounds(230, 110, 80, 25);
 JRadioButton other = new JRadioButton("Other");
 other.setBounds(320, 110, 80, 25);
 ButtonGroup genderGroup = new ButtonGroup();
 genderGroup.add(male);
 genderGroup.add(female);
 genderGroup.add(other);

 JLabel termsLabel = new JLabel("Accept Terms:");
 termsLabel.setBounds(30, 150, 100, 25);

JCheckBox acceptBox = new JCheckBox();
 acceptBox.setBounds(150, 150, 20, 25);

 JLabel countryLabel = new JLabel("Country:");
 countryLabel.setBounds(30, 190, 100, 25);
 String[] countries = {"USA", "Nepal", "India", "UK", "Canada"};
 JComboBox<String> countryBox = new JComboBox<>(countries);
 countryBox.setBounds(150, 190, 250, 25);

 JLabel AgeLabel = new JLabel("Age:");
 AgeLabel.setBounds(30, 230, 100, 25);
 JSlider AgeSlider = new JSlider(0, 100, 50);
 AgeSlider.setBounds(150, 230, 250, 50);
 AgeSlider.setMajorTickSpacing(25);
 AgeSlider.setPaintTicks(true);
 AgeSlider.setPaintLabels(true);

 JLabel commentsLabel = new JLabel("Comments:");
 commentsLabel.setBounds(30, 300, 100, 25);
 JTextArea commentsArea = new JTextArea();
 JScrollPane scrollPane = new JScrollPane(commentsArea);
 scrollPane.setBounds(150, 300, 250, 100);

JButton submitButton = new JButton("Submit");
 submitButton.setBounds(150, 420, 100, 30);

 frame.add(nameLabel);
 frame.add(nameField);
 frame.add(passLabel);
 frame.add(passField);
 frame.add(genderLabel);
 frame.add(male);
 frame.add(female);
 frame.add(other);
 frame.add(termsLabel);
 frame.add(countryLabel);
 frame.add(countryBox);
```

```
        frame.add(AgeLabel);
        frame.add(AgeSlider);
        frame.add(commentsLabel);
        frame.add(scrollPane);
        frame.add(acceptBox);
        frame.add(submitButton);

       frame.setVisible(true);
    }
}
```

**Output:**



Name: Yuvraj Sah

Password: ••••••••

Gender: ● Male ○ Female ○ Other

Accept Terms: ☑

Country: Nepal

Age:

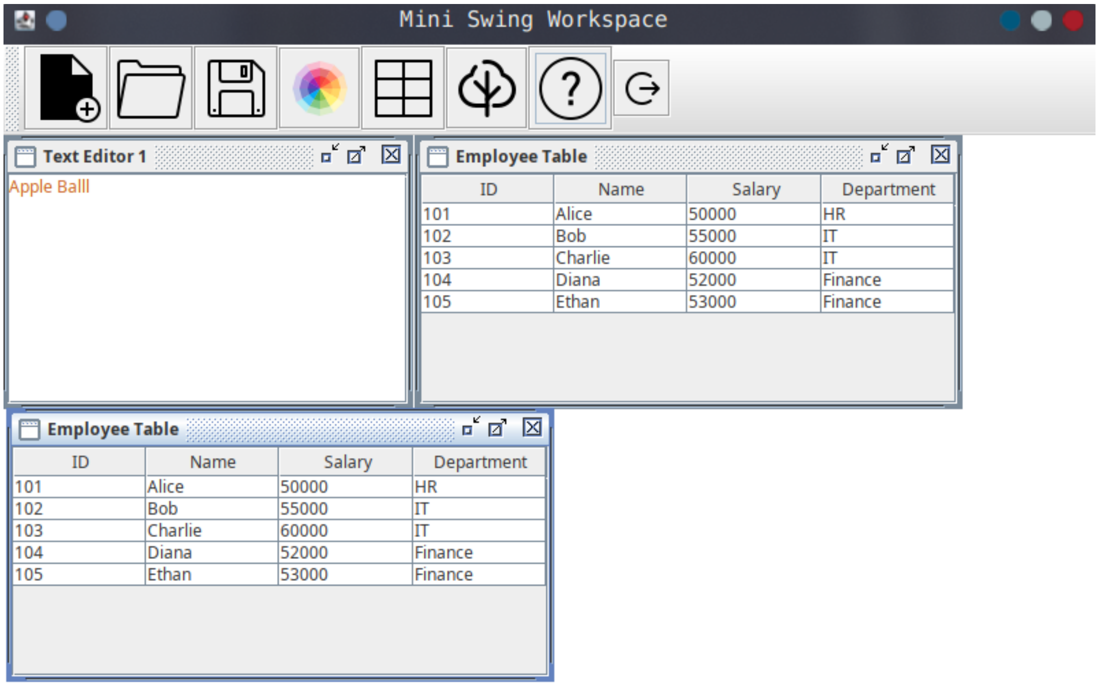
Comments: Please accept my CV...

**Question 2:** Design a multi-feature Java Swing application using toolbars, dialogs, internal frames, tables, trees, file and color choosers, and tooltips. This project introduces real-world GUI development with multiple Swing components in a cohesive mini desktop environment.

**Code:**

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import javax.swing.tree.DefaultMutableTreeNode;
import javax.swing.tree.TreeSelectionModel;
import java.awt.*;
import java.io.*;

public class MiniWorkspace extends JFrame {

    JDesktopPane desktop = new JDesktopPane();
    int frameCount = 0;

    public MiniWorkspace() {
        setTitle("Mini Swing Workspace");
        setSize(800, 600);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        add(desktop);

        // ==== Toolbar ====
        JToolBar toolbar = new JToolBar();

        JButton newBtn = new JButton(new ImageIcon("/home/vuyraj/Downloads/java-labs/Yuvraj Sah/Lab1/icons/new.png")); // 8. Added icon
        newBtn.setToolTipText("Create a new text editor window");
        newBtn.addActionListener(e -> openTextEditor());
        toolbar.add(newBtn);

        JButton openBtn = new JButton(new ImageIcon("/home/vuyraj/Downloads/java-labs/Yuvraj Sah/Lab1/icons/open.png")); // 8. Added icon
        openBtn.setToolTipText("Open file into editor");
        openBtn.addActionListener(e -> openFile());
        toolbar.add(openBtn);

        JButton saveBtn = new JButton(new ImageIcon("/home/vuyraj/Downloads/java-labs/Yuvraj Sah/Lab1/icons/save.png")); // 1. Save functionality
        saveBtn.setToolTipText("Save content from editor");
        saveBtn.addActionListener(e -> saveFile());
        toolbar.add(saveBtn);

        JButton colorBtn = new JButton(new ImageIcon("/home/vuyraj/Downloads/java-labs/Yuvraj Sah/Lab1/icons/color.png")); // 8. Added icon
        colorBtn.setToolTipText("Apply selected color to text");
        colorBtn.addActionListener(e -> chooseColor());
        toolbar.add(colorBtn);

        JButton tableBtn = new JButton(new ImageIcon("/home/vuyraj/Downloads/java-labs/Yuvraj
Sah/Lab1/icons/table.png")); // 8. Added icon
        tableBtn.setToolTipText("View updated employee table");
        tableBtn.addActionListener(e -> openTableFrame());
        toolbar.add(tableBtn);

       JButton treeBtn = new JButton(new ImageIcon("/home/vuyraj/Downloads/java-labs/Yuvraj
Sah/Lab1/icons/tree.png")); // 8. Added icon
        treeBtn.setToolTipText("Show organizational department tree");
        treeBtn.addActionListener(e -> openTreeFrame());
        toolbar.add(treeBtn);

        JButton helpBtn = new JButton(new ImageIcon("/home/vuyraj/Downloads/java-labs/Yuvraj
Sah/Lab1/icons/help.png")); // 5. Help Button
        helpBtn.setToolTipText("Get help about using the application");
        helpBtn.addActionListener(e -> {
            JOptionPane.showMessageDialog(this, """
                     Help Instructions:
                    - New: Create a new editor window
                    - Open: Load text file into editor
                    - Save: Save current text to a file
                    - Color: Change editor text color
                    - Table: View employee records
                    - Tree: View organization tree
                    - Exit: Close application
                    """, "Help", JOptionPane.INFORMATION_MESSAGE);
        });
        toolbar.add(helpBtn);

              JButton exitBtn = new JButton(new ImageIcon("/home/vuyraj/Downloads/java-labs/Yuvraj
Sah/Lab1/icons/exit.png")); // 8. Added icon
        exitBtn.setToolTipText("Exit the application safely");
        exitBtn.addActionListener(e -> exitApp());
        toolbar.add(exitBtn);

        add(toolbar, BorderLayout.NORTH);
    }

    void openTextEditor() {
        JInternalFrame frame = new JInternalFrame("Text Editor " + (++frameCount), true, true, true,
true);
        JTextArea textArea = new JTextArea();
        frame.add(new JScrollPane(textArea));
        frame.setSize(300, 200);
        frame.putClientProperty("textArea", textArea);
        frame.setVisible(true);
        desktop.add(frame);
    }

    void openFile() {
        JFileChooser chooser = new JFileChooser();
        int option = chooser.showOpenDialog(this);
        if (option == JFileChooser.APPROVE_OPTION) {
            File file = chooser.getSelectedFile();
            try {
                BufferedReader reader = new BufferedReader(new FileReader(file));
                JInternalFrame frame = new JInternalFrame("Opened: " + file.getName(), true, true, true,
true);
                JTextArea area = new JTextArea();
                area.read(reader, null);
                reader.close();
                frame.add(new JScrollPane(area));
                frame.setSize(400, 250);
                frame.setLocation(30 * frameCount, 30 * frameCount); // 7. Cascade
                frame.setVisible(true);
                frame.putClientProperty("textArea", area); // Needed for save
                desktop.add(frame);
            } catch (IOException ex) {
                JOptionPane.showMessageDialog(this, "Error reading file", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        }
    }

        // 1. Save Functionality using JFileChooser
    void saveFile() {
        JInternalFrame frame = desktop.getSelectedFrame();
        if (frame != null) {
            JTextArea area = (JTextArea) frame.getClientProperty("textArea");
            if (area != null) {
                JFileChooser chooser = new JFileChooser();
                int option = chooser.showSaveDialog(this);
                if (option == JFileChooser.APPROVE_OPTION) {
                    File file = chooser.getSelectedFile();
                    try (BufferedWriter writer = new BufferedWriter(new FileWriter(file))) {
                        area.write(writer);
                        JOptionPane.showMessageDialog(this, "File saved: " + file.getName());
                    } catch (IOException e) {
                        JOptionPane.showMessageDialog(this, "Failed to save", "Error",
JOptionPane.ERROR_MESSAGE);
                    }
                }
            }
        }
    }
    // 4. Apply selected color to selected editor
    void chooseColor() {
        JInternalFrame frame = desktop.getSelectedFrame();
        if (frame != null) {
            JTextArea area = (JTextArea) frame.getClientProperty("textArea");
            if (area != null) {
                Color selected = JColorChooser.showDialog(this, "Choose Text Color",
area.getForeground());
                if (selected != null) {
                    area.setForeground(selected);
                }
            }
        }
    }

        // 2. Modified table with 2 more rows and "Department" column
    void openTableFrame() {
        String[] cols = {"ID", "Name", "Salary", "Department"};
        Object[][] data = {
            {"101", "Alice", "50000", "HR"},
            {"102", "Bob", "55000", "IT"},
            {"103", "Charlie", "60000", "IT"},
            {"104", "Diana", "52000", "Finance"}, // new
            {"105", "Ethan", "53000", "Finance"}  // new
        };
        JTable table = new JTable(new DefaultTableModel(data, cols));
        JInternalFrame frame = new JInternalFrame("Employee Table", true, true, true, true);
        frame.add(new JScrollPane(table));
        frame.setSize(400, 200);
        frame.setLocation(30 * frameCount, 30 * frameCount); // 7. Cascade
        frame.setVisible(true);
        desktop.add(frame);
    }
    // 3. Added Finance department and employees in tree
    void openTreeFrame() {
        DefaultMutableTreeNode root = new DefaultMutableTreeNode("Departments");

        DefaultMutableTreeNode hr = new DefaultMutableTreeNode("HR");
        hr.add(new DefaultMutableTreeNode("Alice"));

        DefaultMutableTreeNode it = new DefaultMutableTreeNode("IT");
        it.add(new DefaultMutableTreeNode("Bob"));
        it.add(new DefaultMutableTreeNode("Charlie"));

        DefaultMutableTreeNode finance = new DefaultMutableTreeNode("Finance");
        finance.add(new DefaultMutableTreeNode("Diana"));
        finance.add(new DefaultMutableTreeNode("Ethan"));
        root.add(hr);
        root.add(it);
        root.add(finance); // 3. Added

        final JTree tree = new JTree(root);
        tree.setToolTipText("Click a node to see employee info");
        tree.getSelectionModel().setSelectionMode(TreeSelectionModel.SINGLE_TREE_SELECTION);
        tree.addTreeSelectionListener(e -> {
            DefaultMutableTreeNode selected = (DefaultMutableTreeNode)
tree.getLastSelectedPathComponent();
            if (selected != null && selected.isLeaf()) {
                JOptionPane.showMessageDialog(MiniWorkspace.this, "Selected: " + selected.toString());
            }
        });

        JInternalFrame frame = new JInternalFrame("Company Tree", true, true, true, true);
        frame.add(new JScrollPane(tree));
        frame.setSize(250, 300);
        frame.setLocation(30 * frameCount, 30 * frameCount); // 7. Cascade
        frame.setVisible(true);
        desktop.add(frame);
    }

    void exitApp() {
        int confirm = JOptionPane.showConfirmDialog(this, "Are you sure you want to exit?", "Confirm
Exit", JOptionPane.YES_NO_OPTION);
        if (confirm == JOptionPane.YES_OPTION) {
            dispose();
        }
    }

    public static void main(String[] args) {
        MiniWorkspace app = new MiniWorkspace();
        app.setVisible(true);
    }
}
```

**Output:**



Mini Swing Workspace

Text Editor 1

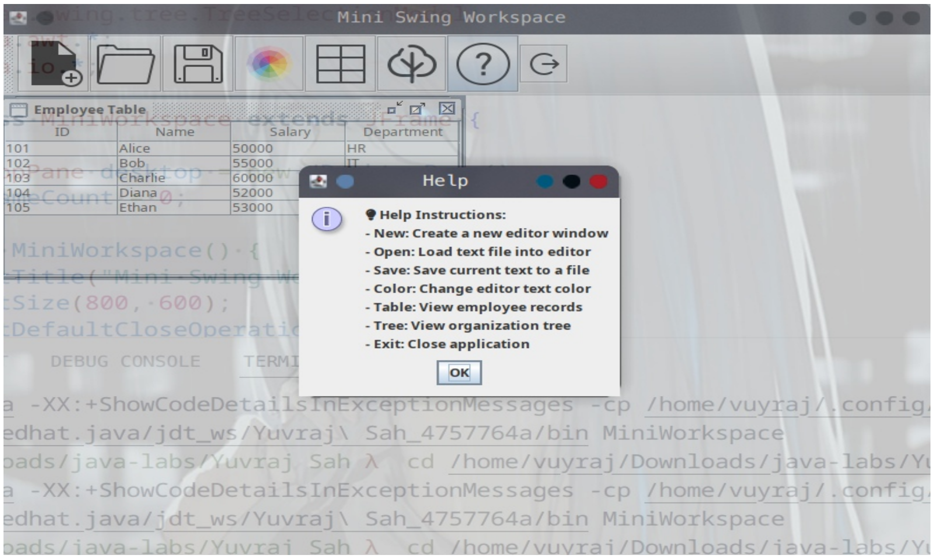
Apple Balll

Employee Table

| ID | Name | Salary | Department |
|---|---|---|---|
| 101 | Alice | 50000 | HR |
| 102 | Bob | 55000 | IT |
| 103 | Charlie | 60000 | IT |
| 104 | Diana | 52000 | Finance |
| 105 | Ethan | 53000 | Finance |

Employee Table

| ID | Name | Salary | Department |
|---|---|---|---|
| 101 | Alice | 50000 | HR |
| 102 | Bob | 55000 | IT |
| 103 | Charlie | 60000 | IT |
| 104 | Diana | 52000 | Finance |
| 105 | Ethan | 53000 | Finance |



Mini Swing Workspace

Employee Table

| ID | Name | Salary | Department |
|---|---|---|---|
| 101 | Alice | 50000 | HR |
| 102 | Bob | 55000 | IT |
| 103 | Charlie | 60000 |  |
| 104 | Diana | 52000 |  |
| 105 | Ethan | 53000 |  |

Help

💡Help Instructions:
- New: Create a new editor window
- Open: Load text file into editor
- Save: Save current text to a file
- Color: Change editor text color
- Table: View employee records
- Tree: View organization tree
- Exit: Close application

```
a -XX:+ShowCodeDetailsInExceptionMessages -cp /home/vuyraj/.config/
edhat.java/jdt_ws/Yuvraj\ Sah_4757764a/bin MiniWorkspace
oads/java-labs/Yuvraj Sah λ  cd /home/vuyraj/Downloads/java-labs/Yu
a -XX:+ShowCodeDetailsInExceptionMessages -cp /home/vuyraj/.config/
edhat.java/jdt_ws/Yuvraj\ Sah_4757764a/bin MiniWorkspace
oads/java-labs/Yuvraj Sah λ  cd /home/vuyraj/Downloads/java-labs/Yu
```

MADAN BHANDARI
MEMORIAL COLLEGE

## Lab 3: Java Swing Layout

**Question 1:** Description of flow, border, grid, gridbag, and group layout in detail.

=> FlowLayout is a simple layout manager that arranges components in a row, from left to right. When the container's width is full, the components wrap to the next line, much like text in a paragraph. It's best used for small toolbars or panels where elements can flow naturally.

BorderLayout divides the container into five regions: north (top), south (bottom), east (right), west (left), and center. Each region can hold only one component. The center region takes up all remaining space, while the others adjust around it. This layout is commonly used in the main content area of frames.

GridLayout arranges components in a grid format with equal-sized cells. You define the number of rows and columns, and each component fills one cell. This layout ensures all components have the same size, which is useful for creating calculator-like interfaces or evenly spaced buttons.

GridBagLayout is one of the most flexible and complex layout managers. It allows components to span multiple rows or columns, and you can control their size, alignment, padding, and positioning using constraints. It's ideal for advanced forms and custom layouts where components vary in size.

GroupLayout is mainly used in GUI design tools like NetBeans. It aligns components horizontally and vertically in groups, allowing developers to organize UI elements in a structured way. It's powerful for creating professional-looking forms, especially when auto-generated by design tools.

**Question 2:** What happens when no layout is used?

=> In Java Swing, every container (like `JFrame`, `JPanel`, etc.) has a default layout manager assigned to it. For example, a `JFrame`'s content pane uses BorderLayout by default, and a `JPanel` uses FlowLayout unless specified otherwise. These default layouts help automatically arrange components in a predictable way, making GUI development easier and more adaptive.

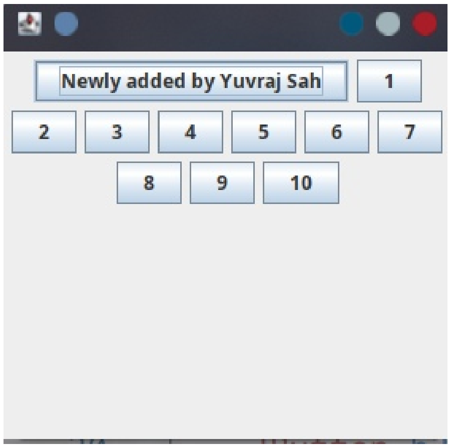
However, when you set the layout to `null`, you're explicitly telling the container not to use any layout manager. This is called absolute positioning, where the developer must manually set the x and y position, as well as width and height, for every component using specific coordinates.

**Question 3:** Write a Program to demonstrate Flow Layout in java Swing.

**Code:**

```
import java.awt.*;
import javax.swing.*;
public class FlowLayoutExample
{
JFrame frameObj;
FlowLayoutExample()
{
   frameObj = new JFrame();
    JButton by = new JButton("Newly added by Yuvraj Sah");
   JButton b1 = new JButton("1");
   JButton b2 = new JButton("2");
   JButton b3 = new JButton("3");
   JButton b4 = new JButton("4");
   JButton b5 = new JButton("5");
   JButton b6 = new JButton("6");
   JButton b7 = new JButton("7");
   JButton b8 = new JButton("8");
   JButton b9 = new JButton("9");
   JButton b10 = new JButton("10");
   frameObj.add(by); frameObj.add(b1); frameObj.add(b2); frameObj.add(b3); frameObj.add(b4);
   frameObj.add(b5); frameObj.add(b6);  frameObj.add(b7);  frameObj.add(b8);
   frameObj.add(b9);  frameObj.add(b10);
   frameObj.setLayout(new FlowLayout());
   frameObj.setSize(300, 300);
   frameObj.setVisible(true);
}
public static void main(String argvs[])
{
   new FlowLayoutExample();
}
}
```

**Output:**

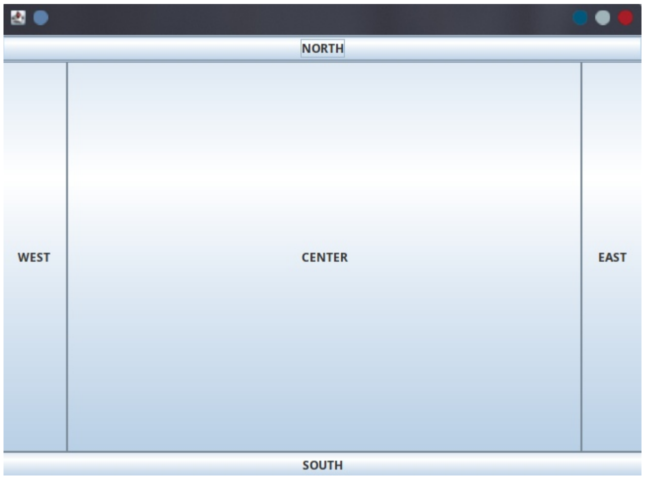


**Question 4:** Write a Program to demonstrate Border Layout in java Swing.

**Code:**

```
import java.awt.*;
import javax.swing.*;
public class Border {
    JFrame f;
    Border() {
        f = new JFrame();
        JButton b1 = new JButton("NORTH");
        JButton b3 = new JButton("EAST");
        JButton b4 = new JButton("WEST");
        JButton b5 = new JButton("CENTER");
        JButton b6 = new JButton("SOUTH");;
        f.add(b1, BorderLayout.NORTH);
        f.add(b3, BorderLayout.EAST);
        f.add(b4, BorderLayout.WEST);
        f.add(b5, BorderLayout.CENTER);
        f.add(b6, BorderLayout.SOUTH);
        f.setSize(300, 300);
        f.setVisible(true);
    }
    public static void main(String[] args) {
        new Border();
    }
}
```

**Output:**

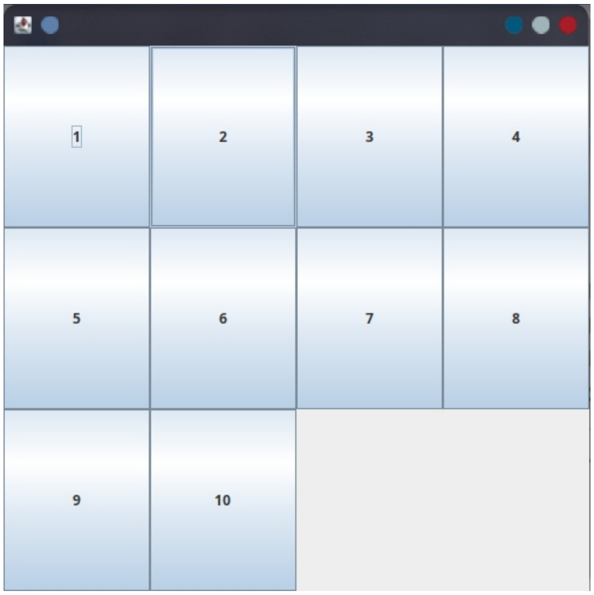


**Question 5:** Write a Program to demonstrate Grid Layout in java Swing.

**Code:**

```
import java.awt.GridLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
public class GridLayoutDemo {
    public static void main(String[] args) {
        JFrame frame = new JFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(500, 500);
        frame.setLayout(new GridLayout(3, 4, 0, 0));
        frame.add(new JButton("1"));
        frame.add(new JButton("2"));
        frame.add(new JButton("3"));
        frame.add(new JButton("4"));
        frame.add(new JButton("5"));
        frame.add(new JButton("6"));
        frame.add(new JButton("7"));
        frame.add(new JButton("8"));
        frame.add(new JButton("9"));
        frame.add(new JButton("10"));
        frame.setVisible(true);
    }
}
```

**Output:**

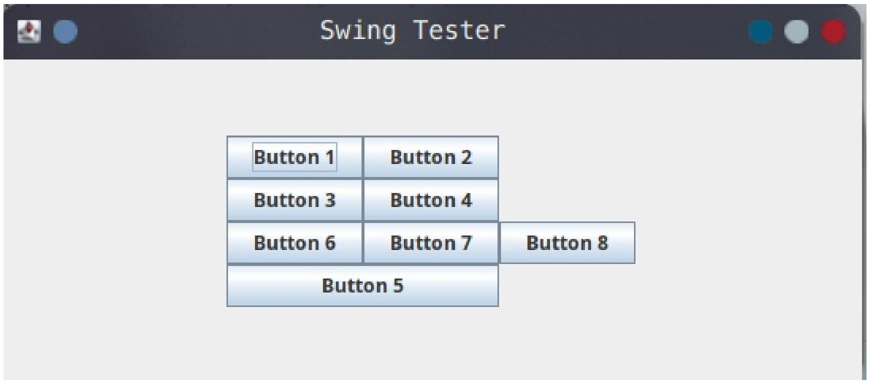


**Question 6:** Write a Program to demonstrate Grid Bag Layout in java Swing.

**Code:**

```
import java.awt.BorderLayout;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
public class GridBagDemo {
   public static void main(String[] args) {
      createWindow();
   }
   private static void createWindow() {
      JFrame frame = new JFrame("Swing Tester");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      createUI(frame);
      frame.setSize(540, 180);
      frame.setLocationRelativeTo(null);
      frame.setVisible(true);
   }
   private static void createUI(final JFrame frame) {
    JPanel panel = new JPanel();
    GridBagLayout layout = new GridBagLayout();
    panel.setLayout(layout);

    GridBagConstraints gbc = new GridBagConstraints();
    gbc.fill = GridBagConstraints.HORIZONTAL;
    gbc.gridx = 0;
    gbc.gridy = 0;
    panel.add(new JButton("Button 1"),gbc);

    gbc.gridx = 1;
    gbc.gridy = 0;
    panel.add(new JButton("Button 2"),gbc);

    gbc.fill = GridBagConstraints.HORIZONTAL;
    gbc.gridx = 0;
    gbc.gridy = 1;
    panel.add(new JButton("Button 3"),gbc);

    gbc.gridx = 1;
    gbc.gridy = 1;
    panel.add(new JButton("Button 4"),gbc);
    gbc.gridx = 0;
    gbc.gridy = 2;

    gbc.fill = GridBagConstraints.HORIZONTAL;
   panel.add(new JButton("Button 6"),gbc);
    gbc.gridx = 1;
    gbc.gridy = 2;

    panel.add(new JButton("Button 7"),gbc);
    gbc.gridx = 2;
    gbc.gridy = 2;

    panel.add(new JButton("Button 8"),gbc);
    gbc.gridx = 0;
    gbc.gridy = 3;

    gbc.fill = GridBagConstraints.HORIZONTAL;
    gbc.gridwidth = 2;
    panel.add(new JButton("Button 5"),gbc);

    frame.getContentPane().add(panel, BorderLayout.CENTER);
  }
}
```

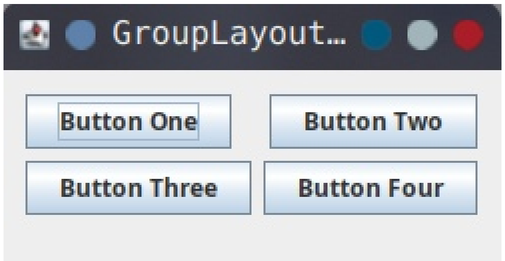
**Output:**



**Question 7:** Write a Program to demonstrate Group Layout in java Swing.
**Code:**

```
import java.awt.Container;
import javax.swing.GroupLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import static javax.swing.GroupLayout.Alignment.*;
public class GroupDemo {
    public static void main(String[] args) {
        JFrame frame = new JFrame("GroupLayoutExample");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container myPanel = frame.getContentPane();

        GroupLayout groupLayout = new GroupLayout(myPanel);
        groupLayout.setAutoCreateGaps(true);
        groupLayout.setAutoCreateContainerGaps(true);
        myPanel.setLayout(groupLayout);

        JButton b1 = new JButton("Button One");
        JButton b2 = new JButton("Button Two");
        JButton b3 = new JButton("Button Three");
        JButton b4 = new JButton("Button Four");

        groupLayout.setHorizontalGroup(groupLayout.createSequentialGroup()
            .addGroup(groupLayout.createParallelGroup(LEADING).addComponent(b1).addComponent(b3))
            .addGroup(groupLayout.createParallelGroup(TRAILING).addComponent(b2).addComponent(b4)));

        groupLayout.setVerticalGroup(groupLayout.createSequentialGroup()
            .addGroup(groupLayout.createParallelGroup(BASELINE).addComponent(b1).addComponent(b2))
            .addGroup(groupLayout.createParallelGroup(BASELINE).addComponent(b3).addComponent(b4)));

        frame.pack();
        frame.setVisible(true);
    }
}
```

**Output:**



**Lab 4: Java Event Handling**

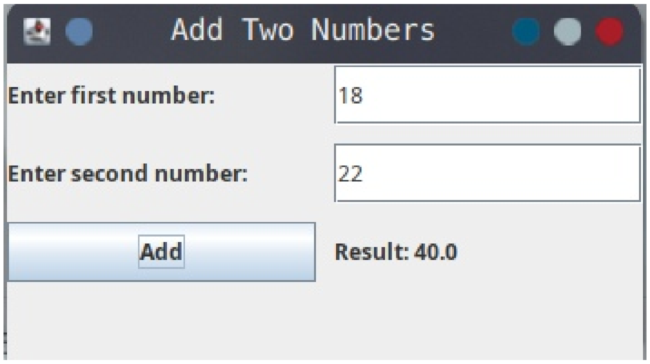
**Question 1:** Write a program using swing components to add two numbers. Use text fields for inputs and a label for output. Your program should display the result when the user presses a button. Your GUI should be properly organized and clear to the user.

**Code:**

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

public class ATNGUI {

    public static void main(String[] args) {

        JFrame frame = new JFrame("Add Two Numbers");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(350, 200);
        frame.setLayout(new GridLayout(4, 2, 10, 10));

        JLabel label1 = new JLabel("Enter first number:");
        JTextField textField1 = new JtextField();

        JLabel label2 = new JLabel("Enter second number:");
        JTextField textField2 = new JtextField();

        JButton addButton = new JButton("Add");
        JLabel resultLabel = new JLabel("Result: ");

        addButton.addActionListener(new ActionListener() {

      public void actionPerformed(ActionEvent e) {
                try {
                    double num1 = Double.parseDouble(textField1.getText());
                    double num2 = Double.parseDouble(textField2.getText());
                    double sum = num1 + num2;
                    resultLabel.setText("Result: " + sum);
                }
```

```
            catch (NumberFormatException ex) {
                    resultLabel.setText("Please enter valid numbers.");
                }
            }
        });

        frame.add(label1);
        frame.add(textField1);

        frame.add(label2);
        frame.add(textField2);

        frame.add(addButton);
        frame.add(resultLabel);

        frame.setLocationRelativeTo(null);
        frame.setVisible(true);
    }
}
```

**Output:**



Enter first number: 18

Enter second number: 22

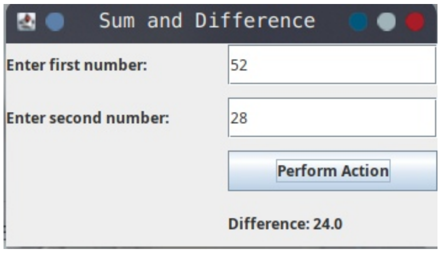
Result: 40.0

**Question 2:** Write a GUI program using components to find sum and difference of two numbers. Use two text fields for giving input and a label for output. The program should display sum if user presses mouse and difference if user release mouse.

**Code:**

```
import javax.swing.*;
import java.awt.*;
public class plusminus {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Sum and Difference");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(350, 200);
        frame.setLayout(new GridLayout(4, 2, 10, 10));
        JLabel label1 = new JLabel("Enter first number:");
        JTextField textField1 = new JTextField();
        JLabel label2 = new JLabel("Enter second number:");
        JTextField textField2 = new JTextField();
        JButton actionButton = new JButton("Perform Action");
        JLabel resultLabel = new JLabel("Result: ");
        actionButton.addMouseListener(new MouseAdapter() {
            public void mousePressed(MouseEvent e) {  try {
                    double num1 = Double.parseDouble(textField1.getText());
                    double num2 = Double.parseDouble(textField2.getText());
                    resultLabel.setText("Sum: " + (num1 + num2));
                } catch (NumberFormatException ex) {
                    resultLabel.setText("Invalid input.");
                } }
            public void mouseReleased(MouseEvent e) {   try {
                    double num1 = Double.parseDouble(textField1.getText());
                    double num2 = Double.parseDouble(textField2.getText());
                    resultLabel.setText("Difference: " + (num1 - num2));
                } catch (NumberFormatException ex) {
                    resultLabel.setText("Invalid input.");
                }     }      });
        frame.add(label1);   frame.add(textField1);
        frame.add(label2);   frame.add(textField2);
        frame.add(new JLabel());   frame.add(actionButton);
        frame.add(new JLabel());    frame.add(resultLabel);
        frame.setLocationRelativeTo(null);
        frame.setVisible(true);      } }
```

**Output:**



Enter first number: 52

Enter second number: 28

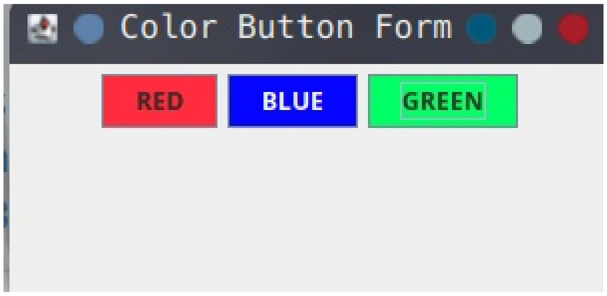
Difference: 24.0

**Question 3:** Using swing components, design a form with three buttons with captions “RED,” “BLUE,” and “GREEN,” respectively. Then write a program to handle the event such that when the user clicks the button, the color of that button will be the same as its caption.

**Code:**

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class Color {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Color Button Form");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 150);
        frame.setLayout(new FlowLayout());
        JButton redButton = new JButton("RED");
        JButton blueButton = new JButton("BLUE");
        JButton greenButton = new JButton("GREEN");
        redButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                redButton.setBackground(java.awt.Color.RED);
            } });
        blueButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                blueButton.setBackground(java.awt.Color.BLUE);
                blueButton.setForeground(java.awt.Color.WHITE);
            }   });
        greenButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                greenButton.setBackground(java.awt.Color.GREEN);
            }});
        frame.add(redButton);
        frame.add(blueButton);
        frame.add(greenButton);
        frame.setLocationRelativeTo(null);
        frame.setVisible(true);
    }
}
```

**Output:**

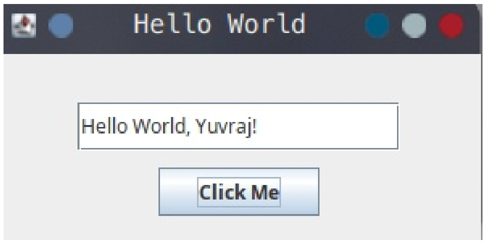


**Question 4:** Write a simple GUI program that displays “Hello World” in a text field. The program should display output if user clicks a button.

**Code:**

```
import javax.swing.*;
import java.awt.event.*;
public class HelloWorldGUI {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Hello World");
        frame.setSize(300, 150);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(null);
        JTextField textField = new JTextField();
        textField.setBounds(50, 30, 200, 30);
        frame.add(textField);
        JButton button = new JButton("Click Me");
        button.setBounds(100, 70, 100, 30);
        frame.add(button);
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                textField.setText("Hello World, Yuvraj!");
            }
        });
        frame.setLocationRelativeTo(null);
        frame.setVisible(true);
    }
}
```

**Output:**



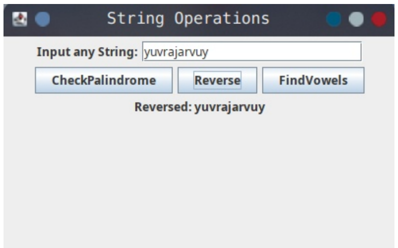
Hello World, Yuvraj!

**Question 5:** Design a GUI form using swing with a text field, a text label for displaying the input message “Input any String”, and three buttons with caption CheckPalindrome, Reverse, FindVowels.

**Code:**

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class StringsOpr {
   public static void main(String[] args) {
      JFrame frame = new JFrame("String Operations");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setSize(400, 250);
      frame.setLayout(new FlowLayout());
      JLabel promptLabel = new JLabel("Input any String:");
      JTextField inputField = new JTextField(20);
      JLabel resultLabel = new JLabel("Result will appear here.");
      JButton palindromeBtn = new JButton("CheckPalindrome");
      JButton reverseBtn = new JButton("Reverse");
      JButton vowelsBtn = new JButton("FindVowels");
      frame.add(promptLabel); frame.add(inputField);  frame.add(palindromeBtn);
      frame.add(reverseBtn);  frame.add(vowelsBtn);  frame.add(resultLabel);
      palindromeBtn.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent e) {
            String input = inputField.getText();
            String reversed = new StringBuilder(input).reverse().toString();
            if (input.equalsIgnoreCase(reversed)) {
               resultLabel.setText("It's a palindrome.");
            } else {
               resultLabel.setText("Not a palindrome.");
            } }});
      reverseBtn.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent e) {
            String input = inputField.getText();
            String reversed = new StringBuilder(input).reverse().toString();
            resultLabel.setText("Reversed: " + reversed);
         }   });
      vowelsBtn.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent e) {
            String input = inputField.getText();
            StringBuilder vowels = new StringBuilder();
            for (char c: input.toLowerCase().toCharArray()) {
               if ("aeiou".indexOf(c) != -1) {
                  vowels.append(c).append(" ");
               }     }
            resultLabel.setText("Vowels: " + vowels.toString());
         } });
      frame.setVisible(true);
   }}
```

**Output:**



Input any String: yuvrajarvuy

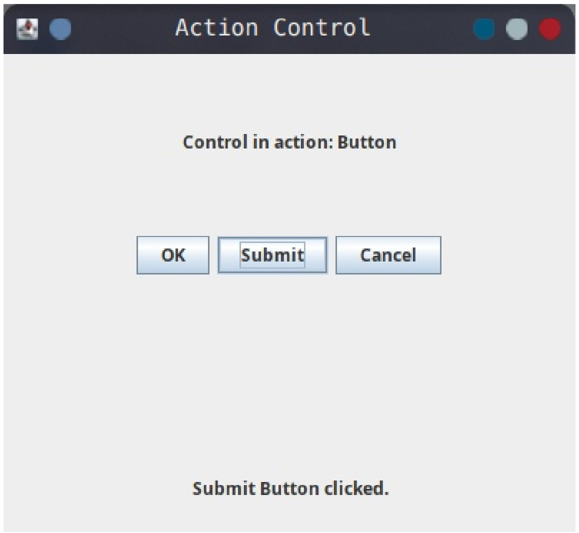
Reversed: yuvrajarvuy

**Question 6:** Write a Java Swing program to demonstrate event handling using multiple buttons (OK, Submit, Cancel) and display the action performed in a status label.

**Code:**

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

public class ActionControl {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public ActionControl() {
        prepareGUI();
    }

    public static void main(String[] args) {
        ActionControl ActionControl = new ActionControl();
        ActionControl.showEventDemo();
    }

    private void prepareGUI() {
        mainFrame = new JFrame("Action Control");
        mainFrame.setSize(400, 400);
        mainFrame.setLayout(new GridLayout(3, 1));

        headerLabel = new JLabel("", JLabel.CENTER);
        statusLabel = new JLabel("", JLabel.CENTER);
        statusLabel.setSize(350, 100);

        mainFrame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent windowEvent) {
                System.exit(0);
            }
        });

        controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());

        mainFrame.add(headerLabel);
        mainFrame.add(controlPanel);
        mainFrame.add(statusLabel);
        mainFrame.setVisible(true);
    }

    private void showEventDemo() {
        headerLabel.setText("Control in action: Button");
        JButton okButton = new JButton("OK");
        JButton submitButton = new JButton("Submit");
        JButton cancelButton = new JButton("Cancel");

        okButton.setActionCommand("OK");
        submitButton.setActionCommand("Submit");
        cancelButton.setActionCommand("Cancel");

        okButton.addActionListener(new ButtonClickListener());
        submitButton.addActionListener(new ButtonClickListener());
        cancelButton.addActionListener(new ButtonClickListener());

        controlPanel.add(okButton);
        controlPanel.add(submitButton);
        controlPanel.add(cancelButton);

        mainFrame.setVisible(true);
    }

    private class ButtonClickListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            String command = e.getActionCommand();

            if (command.equals("OK")) {
                statusLabel.setText("OK Button clicked.");
            } else if (command.equals("Submit")) {
                statusLabel.setText("Submit Button clicked.");
            } else if (command.equals("Cancel")) {
                statusLabel.setText("Cancel Button clicked.");
            }
        }
    }
}
```

**Output:**



Control in action: Button

Submit Button clicked.

**Question 7 :** What is an event in the context of Java GUI programming?
An event in Java GUI programming refers to any user interaction with a graphical component, such as clicking a button, typing in a text field, or selecting a menu item. These interactions generate event objects that notify the program to respond accordingly.

**Question 7 :** How many types of events are there in Java, and how are they categorized?
Java events are mainly categorized into low-level and semantic events. Low-level events include component-level interactions like mouse events (MouseEvent) and key presses (KeyEvent). Semantic events represent user actions like button clicks (ActionEvent), item selections (ItemEvent), or window changes (WindowEvent).

**Question 8:** What is event handling in Java, and why is it important?
Event handling is the process of capturing and responding to user-generated events in a Java program. It allows developers to make GUIs interactive, enabling the application to react to user input like clicks or keystrokes in a meaningful way.

**Question 9:** What is the Event Delegation Model in Java?
The Event Delegation Model is Java's approach to handling events efficiently. It involves three components: the event source (the GUI component), the event object (carries event details), and the event listener (responds to the event). This model improves performance and modularity by separating event generation from event handling logic.

**Question 10:** What are callback methods in Java event handling?
Callback methods are predefined methods in listener interfaces that get automatically called when an event occurs. When a component triggers an event, the Java runtime calls the matching callback method allowing the program to respond to user actions.

**Question 11:** What are the typical steps involved in handling an event in Java?

The Steps involved in handling an event in Java are :-
(1) Create a listener class that implements the relevant listener interface.

(2) Register the listener with a component using methods like addActionListener().

(3) Implement the callback method to define the response. For example, clicking a button triggers actionPerformed() in the registered listener.

**Question 12:** What are listener interfaces and adapter classes in Java?
Listener interfaces define methods for responding to specific events, like MouseListener for mouse actions. Adapter classes are convenience classes that provide empty implementations of listener interfaces with multiple methods, letting developers override only the ones they need, e.g., MouseAdapter.

**Question 13:** What are action commands in Java, and how are they used in event handling?
Action commands are custom string identifiers assigned to components like buttons using setActionCommand(). In a shared listener, getActionCommand() helps identify which component triggered the event, allowing different responses for each component within the same listener.

**Question 14:** How is event handling implemented in Java using the AWT/Swing Event Delegation Model?
In the Event Delegation Model, specific listener interfaces handle different types of events:

- Action Events: Handled by ActionListener with actionPerformed(ActionEvent e)
- Key Events: Handled by KeyListener with keyPressed(), keyReleased(), keyTyped()
- Focus Events: Handled by FocusListener with focusGained(), focusLost()
- Mouse Events: Handled by MouseListener and MouseMotionListener, with methods like mouseClicked(), mouseEntered(), mouseDragged()
- Window Events: Handled by WindowListener with methods like windowClosing(), windowOpened()
- Item Events: Handled by ItemListener using itemStateChanged(ItemEvent e)
  Each event type is linked to its source component and the registered listener responds via these interface methods.

MADAN BHANDARI
MEMORIAL COLLEGE

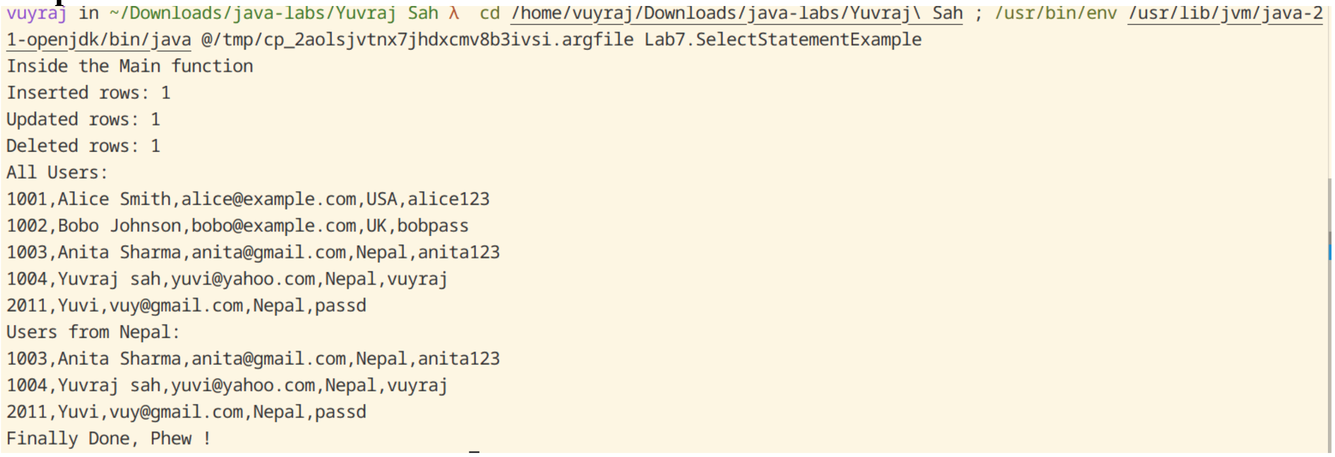
**Lab 7: Java JDBC**

**Question 1:**Write a JDBC program in Java to perform insert, update, delete, and select operations on a table, including a parameterized query to filter users by country.

**Code:**

```
package Lab7;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class SelectStatementExample {
    public static void main(String[] args) {
        System.out.println("Inside the Main function");
        String url = "jdbc:mysql://localhost:3306/testdb";
        String dusername = "root";
        String dpassword = "root";

        try (Connection connection = DriverManager.getConnection(url, dusername, dpassword)) {
            Statement stmt = connection.createStatement();

            // Insert
            String insertSQL = "INSERT INTO Users (id, name, email, country, password) VALUES (2011, 'Yuvi', 'yu@gmail.com', 'Nepal', 'passd')";
            int insertCount = stmt.executeUpdate(insertSQL);
            System.out.println("Inserted rows: " + insertCount);

            // Update
            String updateSQL = "UPDATE Users SET email = 'vuy@gmail.com' WHERE id = 2011";
            int updateCount = stmt.executeUpdate(updateSQL);
            System.out.println("Updated rows: " + updateCount);

            // Delete
            String deleteSQL = "DELETE FROM Users WHERE id = 1011";
            int deleteCount = stmt.executeUpdate(deleteSQL);
            System.out.println("Deleted rows: " + deleteCount);

            // Select all users
            String selectAllSQL = "SELECT id, name, email, country, password FROM Users";
            try (ResultSet rs = stmt.executeQuery(selectAllSQL)) {
                System.out.println("All Users:");
                printUsers(rs);
            }
            // Select users from Nepal using PreparedStatement
            String selectNepalSQL = "SELECT id, name, email, country, password FROM Users WHERE
country = ?";
            try (PreparedStatement pstmt = connection.prepareStatement(selectNepalSQL)) {
                pstmt.setString(1, "Nepal");
                try (ResultSet rsNepal = pstmt.executeQuery()) {
                    System.out.println("Users from Nepal:");
                    printUsers(rsNepal);
                }
            }
        } catch (SQLException e) {
            System.out.println("An error occurred while connecting to MySQL database");
            printSQLException(e);
        } finally {
            System.out.println("Finally Done, Phew ! ");
        }
    }
    private static void printUsers(ResultSet rs) throws SQLException {
        while (rs.next()) {
            int id = rs.getInt("id");
            String name = rs.getString("name");
            String email = rs.getString("email");
            String country = rs.getString("country");
            String password = rs.getString("password");
            System.out.println(id + "," + name + "," + email + "," + country + "," + password);
        }
    }
    public static void printSQLException(SQLException ex) {
        for (Throwable e : ex) {
            if (e instanceof SQLException) {
                e.printStackTrace(System.err);
                System.err.println("SQLState: " + ((SQLException) e).getSQLState());
                System.err.println("Error Code: " + ((SQLException) e).getErrorCode());
                System.err.println("Message: " + e.getMessage());
                Throwable t = ex.getCause();
                while (t != null) {
                    System.err.println("Cause: " + t);
                    t = t.getCause();
                }
            }
        }
    }
}
```

**Output:**



```
vuyraj in ~/Downloads/java-labs/Yuvraj Sah λ  cd /home/vuyraj/Downloads/java-labs/Yuvraj\ Sah ; /usr/bin/env /usr/lib/jvm/java-2
1-openjdk/bin/java @/tmp/cp_2aolsjvtnx7jhdxcmv8b3ivsi.argfile Lab7.SelectStatementExample
Inside the Main function
Inserted rows: 1
Updated rows: 1
Deleted rows: 1
All Users:
1001,Alice Smith,alice@example.com,USA,alice123
1002,Bobo Johnson,bobo@example.com,UK,bobpass
1003,Anita Sharma,anita@gmail.com,Nepal,anita123
1004,Yuvraj sah,yuvi@yahoo.com,Nepal,vuyraj
2011,Yuvi,vuy@gmail.com,Nepal,passd
Users from Nepal:
1003,Anita Sharma,anita@gmail.com,Nepal,anita123
1004,Yuvraj sah,yuvi@yahoo.com,Nepal,vuyraj
2011,Yuvi,vuy@gmail.com,Nepal,passd
Finally Done, Phew !
```

**Question 2:** Discuss JDBC (Java Database Connectivity)

JDBC (Java Database Connectivity) is a standard Java API used to connect and interact with relational databases like MySQL, Oracle, etc. It allows Java applications to execute SQL queries, retrieve results, and perform data manipulation (insert, update, delete). JDBC provides interfaces for database operations, making it easier to switch databases without changing application logic.

**Question 3:** Explain Steps in JDBC

Steps involved in JDBC programming are:

1. Import JDBC packages: Start by importing classes from the java.sql package. These include Connection, DriverManager, Statement, ResultSet, and others that are needed to interact with the database.
2. Register JDBC driver: In older Java versions, the database driver had to be registered manually using Class.forName(). In newer versions, this step is optional as the driver is automatically loaded if it's in the classpath.
3. Establish connection: Use DriverManager.getConnection() with the database URL, username, and password to establish a connection to the database.
4. Create statement: Once connected, create a Statement or PreparedStatement object using the connection. These objects are used to send SQL commands to the database.
5. Execute SQL query: Use executeQuery() to run SELECT queries that return data, and executeUpdate() to run INSERT, UPDATE, or DELETE queries that modify data.
6. Process results: If the query returns data (like a SELECT), use a ResultSet to loop through the rows and retrieve data using methods like getInt(), getString(), etc.
7. Close connections: After the operations are done, close the ResultSet, Statement, and Connection objects to free up system resources. Using try-with-resources makes this automatic and safer.

MADAN BHANDARI
MEMORIAL COLLEGE

**Lab 8: Java Network Programming**

**Question 1:** Write a Java TCP client-server program where the client sends two integers to the server, and the server returns and displays their sum.

**Code:**

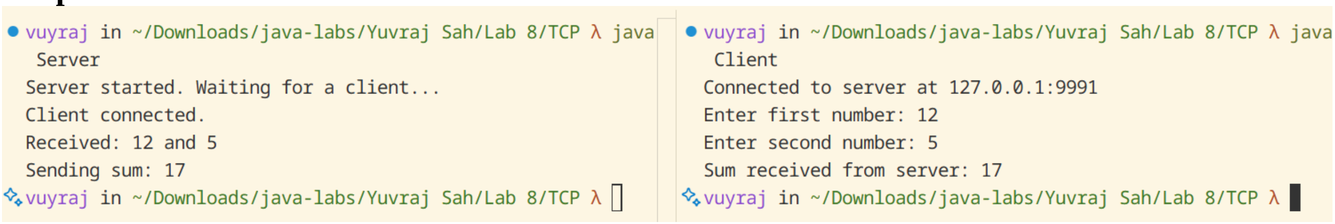
**Server.java**

```
import java.net.*;
import java.io.*;
public class Server {
    private ServerSocket server = null;
    private Socket socket = null;
    private DataInputStream in = null;
    private DataOutputStream out = null;
    public Server(int port) {
        try {
            server = new ServerSocket(port);
            System.out.println("Server started. Waiting for a client...");
            socket = server.accept();
            System.out.println("Client connected."); in = new DataInputStream(socket.getInputStream());
            out = new DataOutputStream(socket.getOutputStream());
            int num1 = in .readInt();
            int num2 = in .readInt();
            System.out.println("Received: " + num1 + " and " + num2);
            int sum = num1 + num2;
            System.out.println("Sending sum: " + sum);
            out.writeInt(sum); in .close();
            out.close();
            socket.close();
            server.close();
        } catch (IOException i) {
            System.err.println("Server error: " + i.getMessage());
        }
    }
    public static void main(String[] args) {
        new Server(9991);
    }
}
```

**Client.java**

```
import java.net.*;
import java.io.*;
public class Client {
    private Socket socket = null;
    private DataInputStream input = null;
    private DataOutputStream out = null;
    private DataInputStream inFromServer = null;
    public Client(String address, int port) {
        try {
            socket = new Socket(address, port);
            System.out.println("Connected to server at " + address + ":" + port);
            input = new DataInputStream(System.in);
            out = new DataOutputStream(socket.getOutputStream());
            inFromServer = new DataInputStream(socket.getInputStream());
            System.out.print("Enter first number: ");
            int num1 = Integer.parseInt(input.readLine());
            System.out.print("Enter second number: ");
            int num2 = Integer.parseInt(input.readLine());
            out.writeInt(num1);
            out.writeInt(num2);
            int sum = inFromServer.readInt();
            System.out.println("Sum received from server: " + sum);
            input.close();
            out.close();
            inFromServer.close();
            socket.close();
        } catch (Exception e) {
            System.err.println("Client error: " + e.getMessage());
        }
    }
    public static void main(String[] args) {
        new Client("127.0.0.1", 9991);
    }
}
```

**Output:**



```
● vuyraj in ~/Downloads/java-labs/Yuvraj Sah/Lab 8/TCP λ java
  Server
 Server started. Waiting for a client...
 Client connected.
 Received: 12 and 5
 Sending sum: 17
 vuyraj in ~/Downloads/java-labs/Yuvraj Sah/Lab 8/TCP λ
```

```
● vuyraj in ~/Downloads/java-labs/Yuvraj Sah/Lab 8/TCP λ java
  Client
 Connected to server at 127.0.0.1:9991
 Enter first number: 12
 Enter second number: 5
 Sum received from server: 17
 vuyraj in ~/Downloads/java-labs/Yuvraj Sah/Lab 8/TCP λ
```

**Question 2:** Write a Java UDP program where the sender sends two messages, 'Welcome java' and 'Bye Java', and the receiver receives and prints both messages in Unicasting mode.
**Code:**

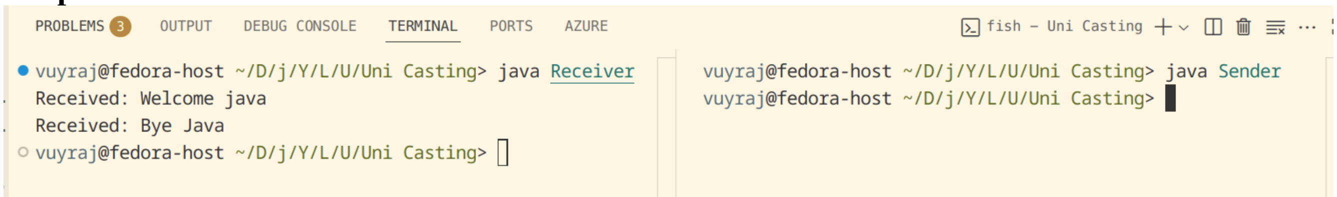
Reciever.java

```
import java.net.*;
public class Receiver {
    public static void main(String[] args) throws Exception {
        DatagramSocket ds = new DatagramSocket(3000);
        byte[] buf = new byte[1024];
        DatagramPacket dp1 = new DatagramPacket(buf, buf.length);
        ds.receive(dp1);
        String msg1 = new String(dp1.getData(), 0, dp1.getLength());
        System.out.println("Received: " + msg1);
        DatagramPacket dp2 = new DatagramPacket(buf, buf.length);
        ds.receive(dp2);
        String msg2 = new String(dp2.getData(), 0, dp2.getLength());
        System.out.println("Received: " + msg2);
        ds.close();
    }
}
```

Sender.java

```
import java.net.*;
public class Sender {
    public static void main(String[] args) throws Exception {
        DatagramSocket ds = new DatagramSocket();
        InetAddress ip = InetAddress.getByName("127.0.0.1");
        String msg1 = "Welcome java";
        DatagramPacket dp1 = new DatagramPacket(msg1.getBytes(), msg1.length(), ip, 3000);
        ds.send(dp1);
        String msg2 = "Bye Java";
        DatagramPacket dp2 = new DatagramPacket(msg2.getBytes(), msg2.length(), ip, 3000);
        ds.send(dp2);
        ds.close();
    }
}
```

**Output:**



```
vuyraj@fedora-host ~/D/j/Y/L/U/Uni Casting> java Receiver
Received: Welcome java
Received: Bye Java
vuyraj@fedora-host ~/D/j/Y/L/U/Uni Casting>
```

```
vuyraj@fedora-host ~/D/j/Y/L/U/Uni Casting> java Sender
vuyraj@fedora-host ~/D/j/Y/L/U/Uni Casting>
```

**Question 3:** Write a Java program using Multicasting to send two messages from a sender, including 'Bye Multicast', and receive and display both messages on the receiver side.

**Code:**

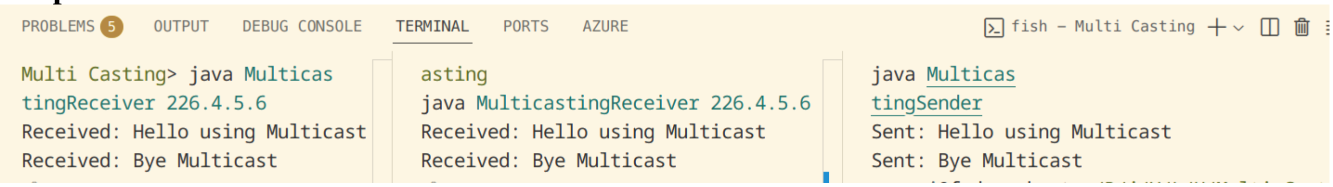
MulticastingSender.java

```
import java.io.IOException;
import java.net.InetAddress;
import java.net.MulticastSocket;
import java.net.DatagramPacket;
public class MulticastingSender {
    public static void main(String[] args) throws IOException {
        String group = "226.4.5.6";
        int port = 5000;
        MulticastSocket ms = new MulticastSocket();
        String message1 = "Hello using Multicast";
        DatagramPacket dp1 = new DatagramPacket(
            message1.getBytes(),
            message1.length(),
            InetAddress.getByName(group),
            port
        );
        ms.send(dp1);
        System.out.println("Sent: " + message1);
        String message2 = "Bye Multicast";
        DatagramPacket dp2 = new DatagramPacket(
            message2.getBytes(),
            message2.length(),
            InetAddress.getByName(group),
            port
        );
        ms.send(dp2);
        System.out.println("Sent: " + message2);
        ms.close();
    }
}
```

MulticastingReceiver.java

```
import java.io.IOException;
import java.net.InetAddress;
import java.net.MulticastSocket;
import java.net.DatagramPacket;
public class MulticastingReceiver {
    public static void main(String[] args) throws IOException {
        if (args.length < 1) {
            System.out.println("Usage: java MulticastingReceiver <multicast-group-ip>");
            return;
        }
        String group = args[0]; // e.g., "226.4.5.6"
        int port = 5000;
        MulticastSocket ms = new MulticastSocket(port);
        InetAddress groupAddress = InetAddress.getByName(group);
        ms.joinGroup(groupAddress);
        byte[] buf = new byte[1024];
        DatagramPacket dp = new DatagramPacket(buf, buf.length);
        for (int i = 0; i < 2; i++) {
            ms.receive(dp);
            String message = new String(dp.getData(), 0, dp.getLength());
            System.out.println("Received: " + message);
        }
        ms.leaveGroup(groupAddress);
        ms.close();
    }
}
```

**Output:**



```
Multi Casting> java Multicas
tingReceiver 226.4.5.6
Received: Hello using Multicast
Received: Bye Multicast
```

```
asting
java MulticastingReceiver 226.4.5.6
Received: Hello using Multicast
Received: Bye Multicast
```

```
java Multicas
tingSender
Sent: Hello using Multicast
Sent: Bye Multicast
```

**Question 4:** What is socket programming used for?

Socket programming is used to enable communication between two devices over a network. It allows data to be sent and received between a client and a server using protocols like TCP or UDP. Sockets help create applications like chat systems, file transfer tools, and network services.

**Question 5:** Differentiate between TCP and UDP.

TCP (Transmission Control Protocol) is a connection-oriented protocol that ensures reliable and ordered delivery of data. It is used when accuracy is important, like in file transfers. UDP (User Datagram Protocol) is connectionless, faster, and does not guarantee delivery or order. It is used for applications like video streaming or online games where speed is more important than accuracy.

**Question 6:** Which package contains all of the Java networking classes and interfaces?

Java networking classes and interfaces are in the java.net package. It includes classes like Socket, ServerSocket, DatagramSocket, DatagramPacket, and InetAddress, and interfaces like SocketImplFactory and ContentHandlerFactory. These are used to implement both TCP and UDP communications.

**Question 7:** Describe the classes used in TCP communications.

In TCP communication, the Socket class is used by the client to connect to the server. The server uses ServerSocket to listen for incoming connections. Once a connection is established, both can use input and output streams to send and receive data.

**Question 8:** Describe the classes used in UDP communications.

In UDP communication, the DatagramSocket class is used to send and receive data. The data itself is sent in the form of DatagramPacket objects. Unlike TCP, there is no connection establishment step, packets are just sent to specific IP addresses and ports.

**Question 9:** Discuss the steps for creating clients and servers and how they communicate using TCP.
In TCP, the server first creates a ServerSocket on a specific port and waits for clients. The client creates a Socket to connect to the server. Once connected, both sides use streams to exchange data. The communication continues until one side closes the connection.

**Question 10:** Differentiate between Unicasting and Multicasting in UDP communication.
Unicasting is sending a UDP packet to a single specific receiver using DatagramSocket. Multicasting is sending a message to a group of receivers using the MulticastSocket class and a multicast IP address. Multicasting is useful in scenarios like video conferencing or live streaming to multiple users at once. Unicasting delivers to one recipient, while multicasting can deliver to many.

**Question 11:** Demonstrate use of URL, URLConnection  classes along with commonly used methods of the classes.

**Code:**

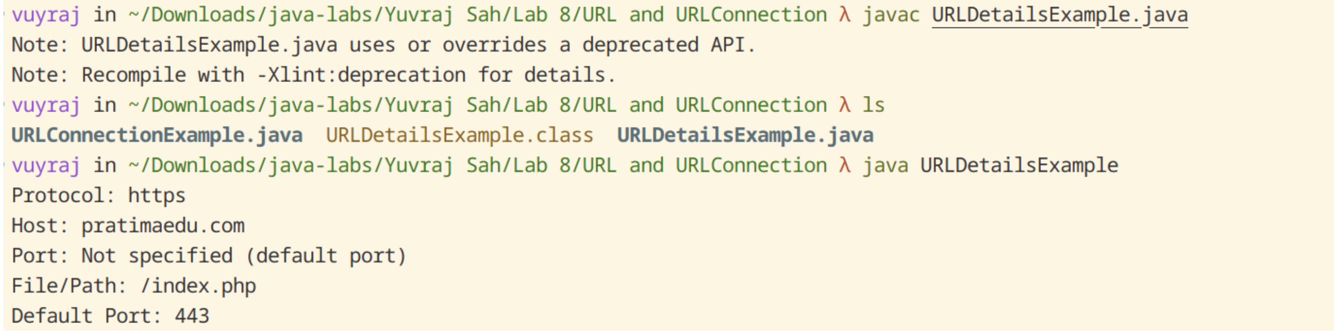
URLdetailsExample.java

```
import java.net.*;
public class URLDetailsExample {
    public static void main(String[] args) {
        try {
            URL url = new URL("https://pratimaedu.com/index.php");
            String protocol = url.getProtocol();
            System.out.println("Protocol: " + protocol);
            String host = url.getHost();
            System.out.println("Host: " + host);
            int port = url.getPort();
            System.out.println("Port: " + (port == -1 ? "Not specified (default port)" : port));
            String file = url.getFile();
            System.out.println("File/Path: " + file);
            int defaultPort = url.getDefaultPort();
            System.out.println("Default Port: " + defaultPort);
        } catch (MalformedURLException e) {
            System.out.println("Malformed URL: " + e.getMessage());
        }
    }
}
```

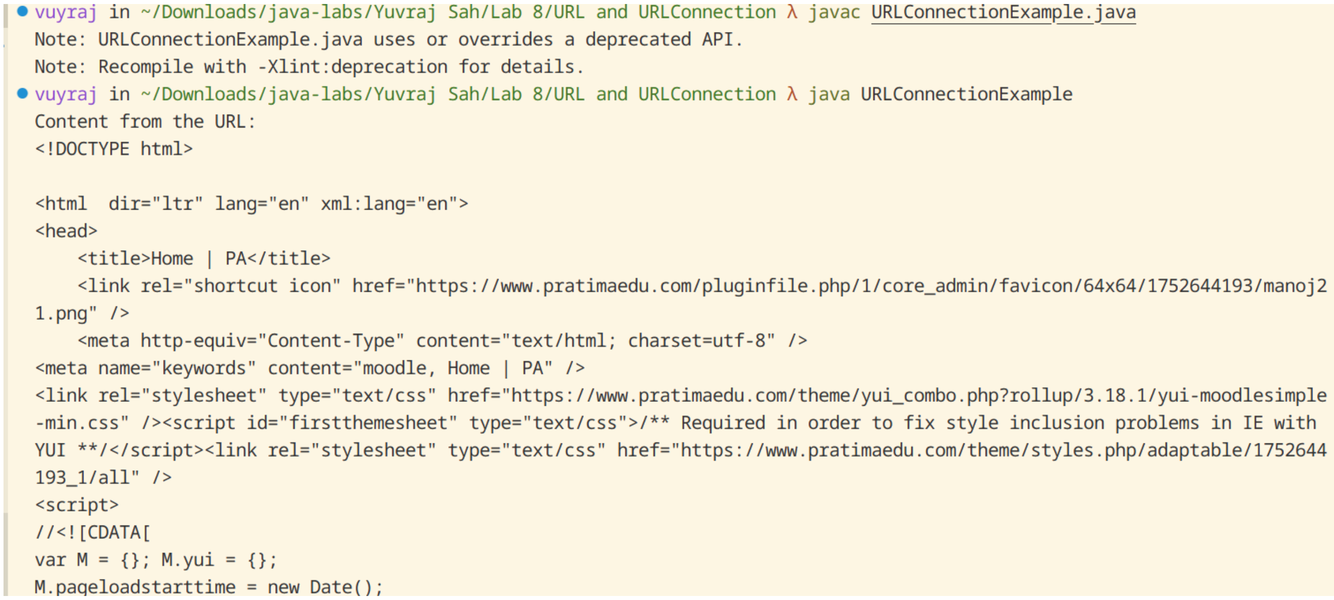
URLConnectionExample.java

```
import java.net.*;
import java.io.*;
public class URLConnectionExample {
    public static void main(String[] args) {
        try {
            URL url = new URL("https://www.pratimaedu.com");
            URLConnection urlConnection = url.openConnection();
            BufferedReader in = new BufferedReader(
                    new InputStreamReader(urlConnection.getInputStream()));
            String inputLine;
            StringBuilder content = new StringBuilder();
            while ((inputLine = in.readLine()) != null) {
                content.append(inputLine).append("\n");
            }
            in.close();
            System.out.println("Content from the URL:");
            System.out.println(content.toString());
        } catch (MalformedURLException e) {
            System.out.println("Malformed URL: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("I/O error: " + e.getMessage());
        }
    }
}
```

**Output:**



```
vuyraj in ~/Downloads/java-labs/Yuvraj Sah/Lab 8/URL and URLConnection λ javac URLDetailsExample.java
Note: URLDetailsExample.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
vuyraj in ~/Downloads/java-labs/Yuvraj Sah/Lab 8/URL and URLConnection λ ls
URLConnectionExample.java  URLDetailsExample.class  URLDetailsExample.java
vuyraj in ~/Downloads/java-labs/Yuvraj Sah/Lab 8/URL and URLConnection λ java URLDetailsExample
Protocol: https
Host: pratimaedu.com
Port: Not specified (default port)
File/Path: /index.php
Default Port: 443
```



```
● vuyraj in ~/Downloads/java-labs/Yuvraj Sah/Lab 8/URL and URLConnection λ javac URLConnectionExample.java
  Note: URLConnectionExample.java uses or overrides a deprecated API.
  Note: Recompile with -Xlint:deprecation for details.
● vuyraj in ~/Downloads/java-labs/Yuvraj Sah/Lab 8/URL and URLConnection λ java URLConnectionExample
  Content from the URL:
  <!DOCTYPE html>

  <html  dir="ltr" lang="en" xml:lang="en">
  <head>
      <title>Home | PA</title>
      <link rel="shortcut icon" href="https://www.pratimaedu.com/pluginfile.php/1/core_admin/favicon/64x64/1752644193/manoj2
  1.png" />
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <meta name="keywords" content="moodle, Home | PA" />
  <link rel="stylesheet" type="text/css" href="https://www.pratimaedu.com/theme/yui_combo.php?rollup/3.18.1/yui-moodlesimple
  -min.css" /><script id="firstthemesheet" type="text/css">/** Required in order to fix style inclusion problems in IE with
  YUI **/</script><link rel="stylesheet" type="text/css" href="https://www.pratimaedu.com/theme/styles.php/adaptable/1752644
  193_1/all" />
  <script>
  //<![CDATA[
  var M = {}; M.yui = {};
  M.pageloadstarttime = new Date();
```

MADAN BHANDARI MEMORIAL COLLEGE

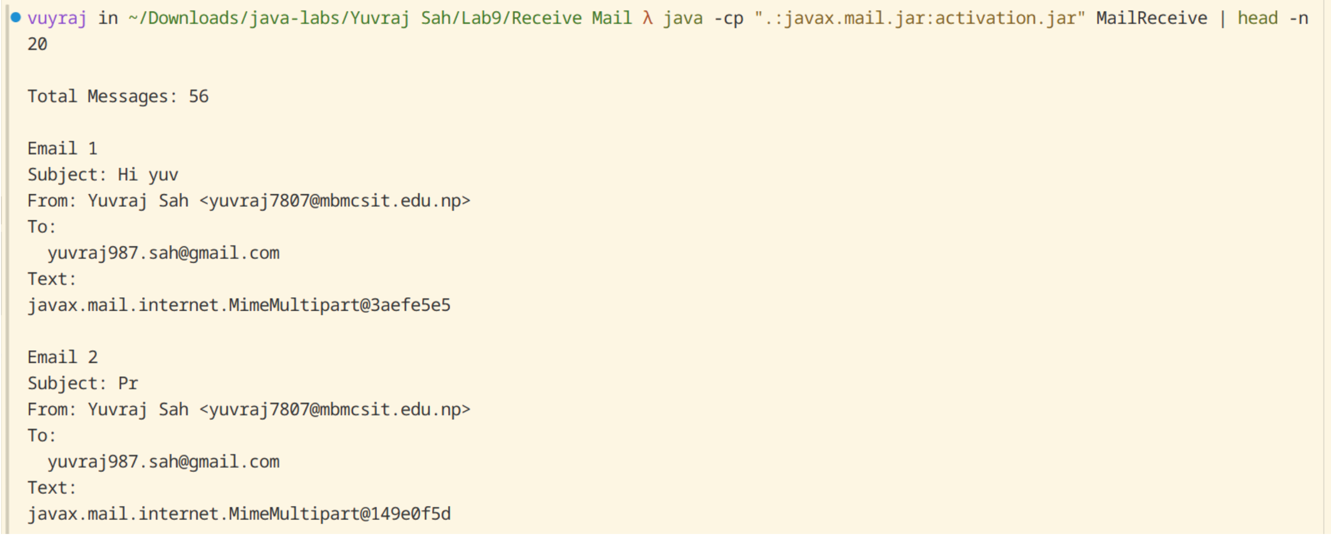
**Lab 9: Java Mail API**

**Question 1:** Write a Java program using JavaMail API to connect to a POP3 mail server and display details of the first few received emails.

**Code:**

```
import java.util.*;
import javax.mail.*;
public class MailReceive {
    public static void receiveEmail(String pop3Host, String storeType, String user, String password) {
        Properties props = new Properties();
        props.put("mail.pop3.host", pop3Host);
        props.put("mail.pop3.port", "995");
        props.put("mail.pop3.starttls.enable", "true");
        props.put("mail.store.protocol", "pop3s");
        Session session = Session.getInstance(props);
        try {
            Store mailStore = session.getStore(storeType);
            mailStore.connect(pop3Host, user, password);
            Folder folder = mailStore.getFolder("INBOX");
            folder.open(Folder.READ_ONLY);
            Message[] emailMessages = folder.getMessages();
            System.out.println("Total Messages: " + emailMessages.length);
            for (int i = 0; i < Math.min(5, emailMessages.length); i++) {
                Message message = emailMessages[i];
                Address[] toAddress = message.getRecipients(Message.RecipientType.TO);
                System.out.println("\nEmail " + (i + 1));
                System.out.println("Subject: " + message.getSubject());
                System.out.println("From: " + message.getFrom()[0]);
                System.out.println("To:");
                for (Address addr : toAddress) {
                    System.out.println("  " + addr.toString());
                }
                System.out.println("Text:\n" + message.getContent().toString());
            }
            folder.close(false);
            mailStore.close();
        } catch (Exception e) {
            System.err.println("Error in receiving email.");
            e.printStackTrace();
        }
    }
    public static void main(String[] args) {
        String pop3Host = "pop.gmail.com";
        String mailStoreType = "pop3s";
        final String userName = "yuvraj987.sah@gmail.com";
        final String password = "fjhhlgzlhkzxbisf";
        receiveEmail(pop3Host, mailStoreType, userName, password);
    }
}
```

**Output:**



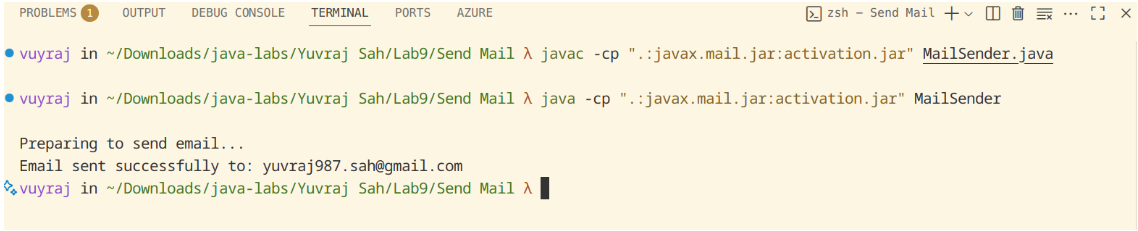
```
● vuyraj in ~/Downloads/java-labs/Yuvraj Sah/Lab9/Receive Mail λ java -cp ".:javax.mail.jar:activation.jar" MailReceive | head -n
20

Total Messages: 56

Email 1
Subject: Hi yuv
From: Yuvraj Sah <yuvraj7807@mbmcsit.edu.np>
To:
  yuvraj987.sah@gmail.com
Text:
javax.mail.internet.MimeMultipart@3aefe5e5

Email 2
Subject: Pr
From: Yuvraj Sah <yuvraj7807@mbmcsit.edu.np>
To:
  yuvraj987.sah@gmail.com
Text:
javax.mail.internet.MimeMultipart@149e0f5d
```

**Question 2:** Write a Java program using JavaMail API to send an email with a text message and a file attachment through Gmail SMTP.

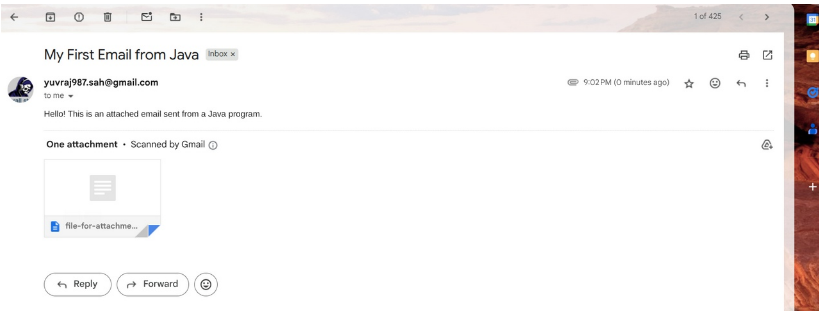
**Code:**

```
import java.util.Properties;
import javax.mail.*;
import javax.mail.internet.*;
public class MailSender {
    public static void main(String[] args) {
        String recipient = "yuvraj987.sah@gmail.com";  // Replace with recipient email
        String senderEmail = "yuvraj987.sah@gmail.com"; // Replace with your email
        String appPassword = "fjhhlgzlhkzxbisf";   // Use an app-specific password
        try {
            sendMail(senderEmail, appPassword, recipient);
        } catch (Exception e) {
            System.err.println("Failed to send email:");
            e.printStackTrace();
        }
    }
    public static void sendMail(String senderEmail, String password, String recipientEmail) throws
Exception {
        System.out.println("Preparing to send email...");
        Properties props = new Properties();
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.starttls.enable", "true");
        props.put("mail.smtp.host", "smtp.gmail.com");
        props.put("mail.smtp.port", "587");
        Session session = Session.getInstance(props, new Authenticator() {
            protected PasswordAuthentication getPasswordAuthentication() {
                return new PasswordAuthentication(senderEmail, password);
            } });
        Message message = new MimeMessage(session);
        message.setFrom(new InternetAddress(senderEmail));
        message.setRecipient(Message.RecipientType.TO, new InternetAddress(recipientEmail));
        message.setSubject("My First Email from Java");
        MimeBodyPart textPart = new MimeBodyPart();
        textPart.setText("Hello! This is an attached email sent from a Java program.");
        MimeBodyPart attachmentPart = new MimeBodyPart();
        String filename = "/home/vuyraj/Downloads/java-labs/Yuvraj Sah/Lab9/Send Mail/file-for-
attachment.txt"; // Change to your file path
        attachmentPart.attachFile(filename);
        Multipart multipart = new MimeMultipart();
        multipart.addBodyPart(textPart);
        multipart.addBodyPart(attachmentPart);
        message.setContent(multipart);
        Transport.send(message);
        System.out.println("Email sent successfully to: " + recipientEmail);
    }
}
```

## Output:



```
● vuyraj in ~/Downloads/java-labs/Yuvraj Sah/Lab9/Send Mail λ javac -cp ".:javax.mail.jar:activation.jar" MailSender.java

● vuyraj in ~/Downloads/java-labs/Yuvraj Sah/Lab9/Send Mail λ java -cp ".:javax.mail.jar:activation.jar" MailSender

  Preparing to send email...
  Email sent successfully to: yuvraj987.sah@gmail.com
  vuyraj in ~/Downloads/java-labs/Yuvraj Sah/Lab9/Send Mail λ ▌
```



My First Email from Java

yuvraj987.sah@gmail.com
to me

9:02 PM (0 minutes ago)

Hello! This is an attached email sent from a Java program.

One attachment • Scanned by Gmail

file-for-attachme...

**Question 3:** What is Java Mail API?
Java Mail API is a set of classes and interfaces provided by Java to allow sending and receiving email from Java applications. It supports standard mail protocols like SMTP, POP3, and IMAP. Developers use this API to build email-related features like notifications, mail clients, and automated mail sending.

**Question 4:** What are the two jar libraries used for mail API?
The two required libraries are javax.mail.jar and activation.jar. The first provides classes to send and receive emails while the second is used to handle MIME-type attachments and data.

**Question 5:** What are the classes used?
Commonly used classes in the Java Mail API include Session for configuring email settings, Message and MimeMessage for creating email content, Transport for sending the message, Store and Folder for receiving messages, and Multipart and BodyPart for handling attachments and complex email content.

**Question 6:** Steps in sending mail using mail API.

- Set up mail properties like SMTP host, port number, authentication, and TLS settings.
- Create a Session object using the sender's email and password with authentication enabled.
- Create a MimeMessage object to represent the email.
- Set the sender's and recipient's email addresses using the setFrom and setRecipient methods.
- Set the subject and content of the email using setSubject and setText or setContent.
- If needed, create a MimeBodyPart for attachments and combine it with the message using a Multipart object.
- Send the email using Transport.send() method.
- Print confirmation or handle exceptions as necessary.

**Question 7:** Steps in receiving mail

- Set up mail properties for the POP3 or IMAP server (such as host, port, and protocol).
- Create a `Session` object using the configured properties.
- Get a `Store` object and connect to the mail server using the username and password.
- Access the desired folder (usually "INBOX") using a `Folder` object.
- Open the folder in read-only or read-write mode.
- Use `getMessages()` to fetch all available messages.
- Loop through the messages to read details like subject, sender, and content.
- Close the folder after processing the emails.
- Close the mail store to end the session.

MADAN BHANDARI
MEMORIAL COLLEGE

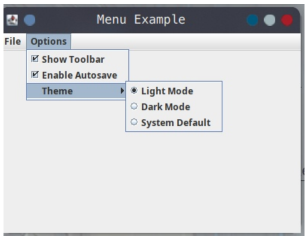
**Lab 5: Java Swing Menu**

**Question 1:** Create a Java Swing application with a menu bar containing File and Options menus, submenu items with icons, checkboxes, radio buttons.

**Code:**

```
import javax.swing.*;
import java.awt.*;
public class MenuExample {

 public static void main(String[] args) {
        JFrame frame = new JFrame("Menu Example");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 300);

        JMenuBar menuBar = new JMenuBar();
        JMenu fileMenu = new JMenu("File");
        JMenuItem newItem = new JMenuItem("New");
        JMenuItem openItem = new JMenuItem("Open", new ImageIcon("open.png")); // Icon for Open
        JMenuItem saveItem = new JMenuItem("Save");
        saveItem.setEnabled(false); // Disabled
        JMenuItem exitItem = new JMenuItem("Exit");

        JMenu recentFiles = new JMenu("Recent Files");
        JMenuItem file1 = new JMenuItem("File1.txt");
        JMenuItem file2 = new JMenuItem("File2.txt");

        recentFiles.add(file1);
        recentFiles.add(file2);
        fileMenu.add(newItem);
        fileMenu.add(openItem);
        fileMenu.add(saveItem);
        fileMenu.addSeparator(); // Separator before Exit
        fileMenu.add(recentFiles);
        fileMenu.add(exitItem);
        JMenu optionsMenu = new JMenu("Options");

       JCheckBoxMenuItem showToolbar = new JCheckBoxMenuItem("Show Toolbar", true);
        JCheckBoxMenuItem enableAutosave = new JCheckBoxMenuItem("Enable Autosave", true);
        JMenu themeMenu = new JMenu("Theme");
        JRadioButtonMenuItem lightMode = new JRadioButtonMenuItem("Light Mode");
        JRadioButtonMenuItem darkMode = new JRadioButtonMenuItem("Dark Mode");
        JRadioButtonMenuItem systemDefault = new JRadioButtonMenuItem("System Default", true);
        ButtonGroup themeGroup = new ButtonGroup();
        themeGroup.add(lightMode);
        themeGroup.add(darkMode);
        themeGroup.add(systemDefault);
        themeMenu.add(lightMode);
        themeMenu.add(darkMode);
        themeMenu.add(systemDefault);
        optionsMenu.add(showToolbar);
        optionsMenu.add(enableAutosave);
        optionsMenu.add(themeMenu);
        menuBar.add(fileMenu);
        menuBar.add(optionsMenu);
        frame.setJMenuBar(menuBar);

       frame.setVisible(true);
    }
}
```

**Output:**



**Question 2:** Create a Java Swing application with a menu bar containing File and Options menus, submenu items with icons, checkboxes, radio buttons, action listeners using anonymous inner classes, and keyboard shortcuts.

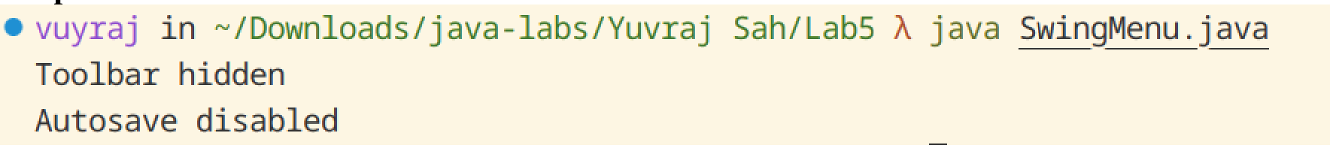
**Code:**

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

public class SwingMenu {

  public SwingMenu() {

        JFrame frame = new JFrame("Swing Menu Starter");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(500, 400);
```

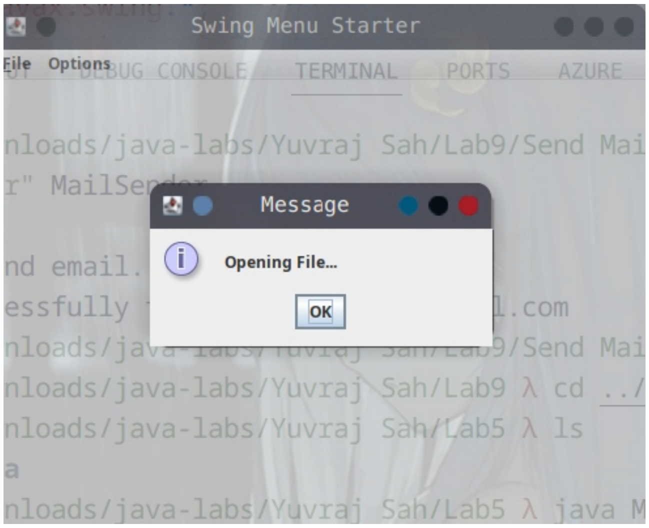
```
 JMenuBar menuBar = new JMenuBar();
 JMenu fileMenu = new JMenu("File");
 fileMenu.setMnemonic(KeyEvent.VK_F);  // Alt + F
 JMenuItem newItem = new JMenuItem("New");
newItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_N,ActionEvent.CTRL_MASK));
 JMenuItem openItem = new JMenuItem("Open", new ImageIcon("open.png"));
 JMenuItem saveItem = new JMenuItem("Save");
 saveItem.setEnabled(false);
 JMenuItem exitItem = new JMenuItem("Exit");
 exitItem.setMnemonic(KeyEvent.VK_E);
 exitItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_Q,ActionEvent.CTRL_MASK));
 JMenu recentFiles = new JMenu("Recent Files");
 JMenuItem file1 = new JMenuItem("File1.txt");
 JMenuItem file2 = new JMenuItem("File2.txt");

recentFiles.add(file1);
 recentFiles.add(file2);
 fileMenu.add(newItem);
 fileMenu.add(openItem);
 fileMenu.add(saveItem);
 fileMenu.addSeparator();
 fileMenu.add(recentFiles);
 fileMenu.add(exitItem);

 JMenu optionsMenu = new JMenu("Options");
 JCheckBoxMenuItem showToolbar = new JCheckBoxMenuItem("Show Toolbar", true);
 JCheckBoxMenuItem enableAutosave = new JCheckBoxMenuItem("Enable Autosave", true);
 JMenu themeMenu = new JMenu("Theme");
 JRadioButtonMenuItem lightMode = new JRadioButtonMenuItem("Light Mode");
 JRadioButtonMenuItem darkMode = new JRadioButtonMenuItem("Dark Mode");
 JRadioButtonMenuItem systemDefault = new JRadioButtonMenuItem("System Default", true);
 ButtonGroup themeGroup = new ButtonGroup();
 themeGroup.add(lightMode);
 themeGroup.add(darkMode);
 themeGroup.add(systemDefault);
 themeMenu.add(lightMode);
 themeMenu.add(darkMode);
 themeMenu.add(systemDefault);
 optionsMenu.add(showToolbar);
 optionsMenu.add(enableAutosave);
 optionsMenu.add(themeMenu);
 menuBar.add(fileMenu);
 menuBar.add(optionsMenu);
 frame.setJMenuBar(menuBar);

  newItem.addActionListener(new ActionListener() {
     public void actionPerformed(ActionEvent e) {
        JOptionPane.showMessageDialog(frame, "New File Created");
     }
 });

 openItem.addActionListener(new ActionListener() {
     public void actionPerformed(ActionEvent e) {
        JOptionPane.showMessageDialog(frame, "Opening File...");
     }
 });

 exitItem.addActionListener(new ActionListener() {
     public void actionPerformed(ActionEvent e) {
        System.exit(0);
     }
 });

 showToolbar.addActionListener(new ActionListener() {
     public void actionPerformed(ActionEvent e) {
        if (showToolbar.isSelected())
            System.out.println("Toolbar shown");
        else
            System.out.println("Toolbar hidden");
     }
 });

 enableAutosave.addActionListener(new ActionListener() {
     public void actionPerformed(ActionEvent e) {
        if (enableAutosave.isSelected())
            System.out.println("Autosave enabled");
        else
            System.out.println("Autosave disabled");
     }
 });

 lightMode.addActionListener(new ActionListener() {
     public void actionPerformed(ActionEvent e) {
        JOptionPane.showMessageDialog(frame, "Light Mode selected");
     }
 });

 darkMode.addActionListener(new ActionListener() {
     public void actionPerformed(ActionEvent e) {
        JOptionPane.showMessageDialog(frame, "Dark Mode selected");
     }
 });

 systemDefault.addActionListener(new ActionListener() {
     public void actionPerformed(ActionEvent e) {
        JOptionPane.showMessageDialog(frame, "System Default selected");
     }
 });
```

```
        frame.setVisible(true);
    }

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

**Output:**



```
vuyraj in ~/Downloads/java-labs/Yuvraj Sah/Lab5 λ java SwingMenu.java
Toolbar hidden
Autosave disabled
```



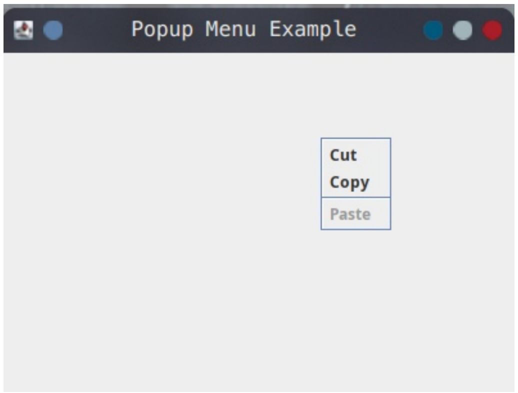
Opening File...

**Question 3:** Create a Java Swing application that shows a popup menu with Cut, Copy, and disabled Paste options on right-click, and displays a dialog when a menu item is selected.

**Code:**

```
import javax.swing.*;
import java.awt.event.*;
public class PopupMenu {
    public PopupMenu() {
        JFrame frame = new JFrame("Popup Menu Example");
        frame.setSize(400, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JPopupMenu popupMenu = new JPopupMenu();
        JMenuItem cutItem = new JMenuItem("Cut");
        JMenuItem copyItem = new JMenuItem("Copy");
        JMenuItem pasteItem = new JMenuItem("Paste");
        pasteItem.setEnabled(false);
        popupMenu.add(cutItem);
        popupMenu.add(copyItem);
        popupMenu.addSeparator();
        popupMenu.add(pasteItem);
        cutItem.addActionListener(e ->
            JOptionPane.showMessageDialog(frame, "Cut selected")
        );
        copyItem.addActionListener(e ->
            JOptionPane.showMessageDialog(frame, "Copy selected")
        );
        pasteItem.addActionListener(e ->
            JOptionPane.showMessageDialog(frame, "Paste selected")
        );
        frame.addMouseListener(new MouseAdapter() {
            public void mousePressed(MouseEvent e) {
                showPopup(e);
            }
            public void mouseReleased(MouseEvent e) {
                showPopup(e);
            }
            private void showPopup(MouseEvent e) {
                if (e.isPopupTrigger()) {
                    popupMenu.show(e.getComponent(), e.getX(), e.getY());
                }
            }
        });
        frame.setVisible(true);
    }
    public static void main(String[] args) {
        new PopupMenu();
    }
}
```

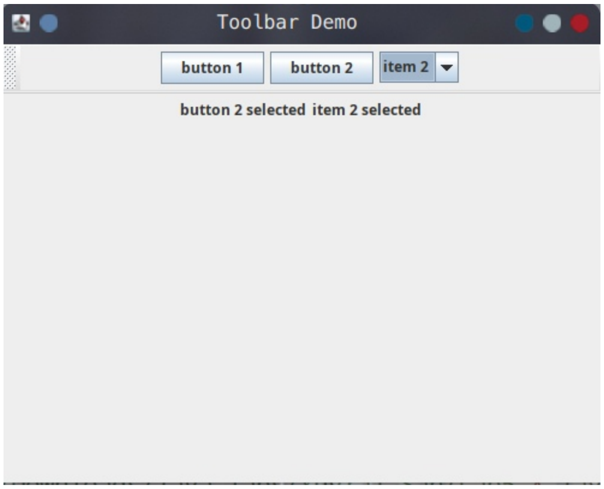
**Output:**



**Question 4:** Write a Java Swing program that uses a toolbar with two buttons and a combo box, and displays the selected button or item in labels.
**Code:**

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

public class ToolbarDemoStarter extends JFrame implements ActionListener, ItemListener {

    static JToolBar toolBar;
    static JButton button1, button2;
    static JComboBox<String> comboBox;
    static JLabel label1, label2;

    public static void main(String[] args) {
        ToolbarDemoStarter demo = new ToolbarDemoStarter();

        JFrame frame = new JFrame("Toolbar Demo");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(new BorderLayout());

        toolBar = new JToolBar();

        JPanel panel = new JPanel();

        button1 = new JButton("button 1");
        button2 = new JButton("button 2");

        button1.addActionListener(demo);
        button2.addActionListener(demo);

        comboBox = new JComboBox<>(new String[] { "item 1", "item 2" });
        comboBox.addItemListener(demo);

        panel.add(button1);
        panel.add(button2);
        panel.add(comboBox);
        toolBar.add(panel);

        JPanel labelPanel = new JPanel();
        label1 = new JLabel("nothing selected");
        label2 = new JLabel("nothing selected");

        labelPanel.add(label1);
        labelPanel.add(label2);


        frame.add(toolBar, BorderLayout.NORTH);
        frame.add(labelPanel, BorderLayout.CENTER);

        frame.setSize(500, 400);
        frame.setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {
        label1.setText(e.getActionCommand() + " selected");
    }
    public void itemStateChanged(ItemEvent e) {
        if (e.getStateChange() == ItemEvent.SELECTED) {
            label2.setText(e.getItem().toString() + " selected");
        }
    }
}
```

**Output:**



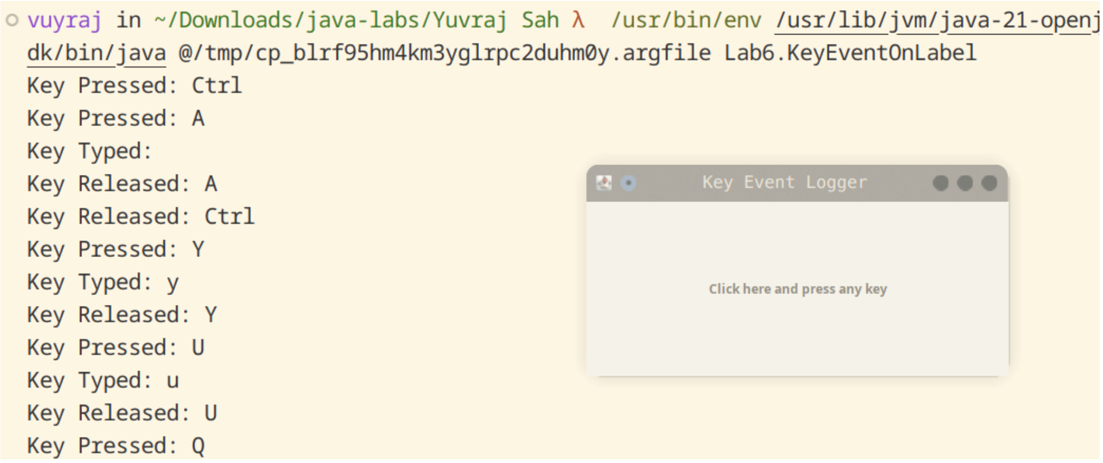
button 2 selected item 2 selected

**Lab 6: Java Swing Event**

**Question 1:** Write a swing program that logs key events to the console.
**Code:**

```
package Lab6;
import javax.swing.*;
import java.awt.event.*;
public class KeyEventOnLabel {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Key Event Logger");
        frame.setSize(400, 200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JLabel label = new JLabel("Click here and press any key", SwingConstants.CENTER);
        label.setFocusable(true);
        label.addKeyListener(new KeyListener() {
            public void keyPressed(KeyEvent e) {
                System.out.println("Key Pressed: " + KeyEvent.getKeyText(e.getKeyCode()));
            }
            public void keyReleased(KeyEvent e) {
                System.out.println("Key Released: " + KeyEvent.getKeyText(e.getKeyCode()));
            }
            public void keyTyped(KeyEvent e) {
                System.out.println("Key Typed: " + e.getKeyChar());
            }
        });
        frame.add(label);
        frame.setVisible(true);
        SwingUtilities.invokeLater(() -> label.requestFocusInWindow());
    }
}
```

**Output:**



```
○ vuyraj in ~/Downloads/java-labs/Yuvraj Sah λ  /usr/bin/env /usr/lib/jvm/java-21-openj
dk/bin/java @/tmp/cp_blrf95hm4km3yglrpc2duhm0y.argfile Lab6.KeyEventOnLabel
Key Pressed: Ctrl
Key Pressed: A
Key Typed: 
Key Released: A
Key Released: Ctrl
Key Pressed: Y
Key Typed: y
Key Released: Y
Key Pressed: U
Key Typed: u
Key Released: U
Key Pressed: Q
```

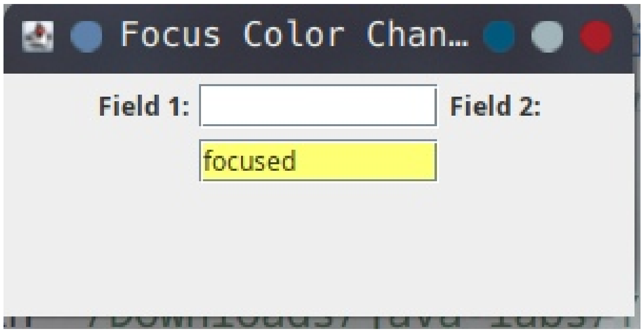
**Question 2:** Create two text fields. When one of the text fields gains focus, change its background color to yellow. When it loses focus, change it back to white.

**Code:**

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

public class FocusEventdemo {

public static void main(String[] args) {

       JFrame frame = new JFrame("Focus Color Change");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(300, 150);
    frame.setLayout(new FlowLayout());

    JTextField textField1 = new JTextField(10);
    JTextField textField2 = new JTextField(10);

       FocusListener focusListener = new FocusAdapter() {
       public void focusGained(FocusEvent e) {
          ((JTextField) e.getSource()).setBackground(Color.YELLOW);
       }

        public void focusLost(FocusEvent e) {
          ((JTextField) e.getSource()).setBackground(Color.WHITE);
       }
    };

    textField1.addFocusListener(focusListener);
    textField2.addFocusListener(focusListener);
```

```
        frame.add(new JLabel("Field 1:"));
        frame.add(textField1);
        frame.add(new JLabel("Field 2:"));
        frame.add(textField2);
        frame.setVisible(true);
    }
}
```

**Output:**



Focus Color Chan…

Field 1:

Field 2:

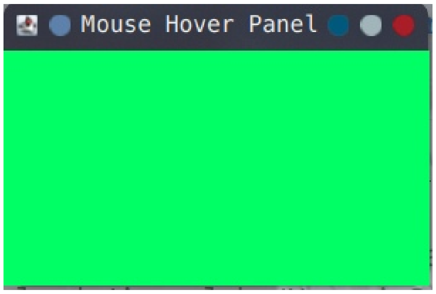
focused

**Question 3:** Create a grey panel that changes its background color to green when the mouse enters the panel and changes back to its original color when the mouse exits.

**Code:**

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class PanelMouseHover {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Mouse Hover Panel");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 200);

        JPanel panel = new JPanel();
        Color originalColor = Color.LIGHT_GRAY;

        panel.setBackground(originalColor);

        panel.addMouseListener(new MouseAdapter() {

            public void mouseEntered(MouseEvent e) {
                panel.setBackground(Color.GREEN);
            }

            public void mouseExited(MouseEvent e) {
                panel.setBackground(originalColor);
            }
        });
        frame.add(panel);
        frame.setVisible(true);
    }
}
```

**Output:**

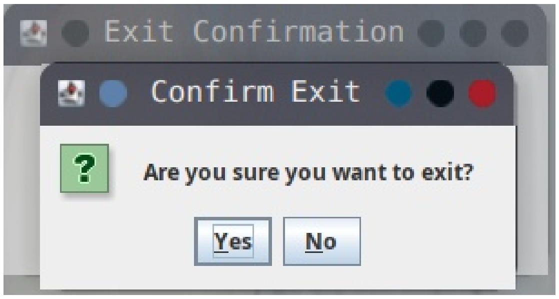


**Question 4:** Create a window that displays a confirmation dialog asking "Are you sure you want to exit?" when the user attempts to close the window.

**Code:**

```
import javax.swing.*;
import java.awt.event.*;
public class ExitConfirmationWindow {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Exit Confirmation");
        frame.setSize(300, 150);
        frame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        frame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                int response = JOptionPane.showConfirmDialog(
                    frame,
                    "Are you sure you want to exit?",
                    "Confirm Exit",
                    JOptionPane.YES_NO_OPTION,
                    JOptionPane.QUESTION_MESSAGE
                );

                if (response == JOptionPane.YES_OPTION) {
                    frame.dispose(); // Close the window
                }
            }
        });
        frame.setVisible(true);
    }
}
```

**Output:**



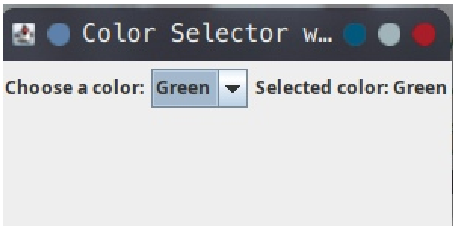
Are you sure you want to exit?

**Question 5:** Create a combo box with a list of colors (e.g., "Red", "Blue", "Green"). Display the selected color name in a label when a selection is made.

**Code:**

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class ColorComboBoxWithItemEvent {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Color Selector with Item Event");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 150);
        frame.setLayout(new FlowLayout());
        String[] colors = { "Red", "Blue", "Green" };
        JComboBox<String> colorComboBox = new JComboBox<>(colors);
        JLabel selectedLabel = new JLabel("Selected color: ");
        colorComboBox.addItemListener(new ItemListener() {
            public void itemStateChanged(ItemEvent e) {
                if (e.getStateChange() == ItemEvent.SELECTED) {
                    String selectedColor = (String) e.getItem();
                    selectedLabel.setText("Selected color: " + selectedColor);
                }
            }
        });
        frame.add(new JLabel("Choose a color:"));
        frame.add(colorComboBox);
        frame.add(selectedLabel);
        frame.setVisible(true);
    }
}
```

**Output:**



Choose a color: Green Selected color: Green