

## JAVA LAB PROGRAMS

1. Declare a class Named Teacher. The class will have all the data members(First\_Name, Last\_Name,Emp\_ ID, Dept\_Name) as per your convenient.The class will have constructors. Write a function to read the values of the class variables. The values of the variable will be stored in a FILE (text file). The values will be stored in a structured format of your own choice.Further, read the content of the FILE and display the content in an ordered form (First Name, LastName).

Concept Learning:

1. FILE manipulation
2. Use try catch blocks
3. Use multiple try catch block
4. Finally statement

Try to have your own Exception

**Code:**

### Teacher.java

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

public class Teacher {
    private String firstName;
    private String lastName;
    private String empld;
    private String deptName;

    public Teacher(String firstName, String lastName, String empld, String deptName) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.empld = empld;
        this.deptName = deptName;
    }

    public String getFirstName() {
        return firstName;
    }
}
```

```

public void setFirstName(String firstName) {
    this.firstName = firstName;
}

public String getLastName() {
    return lastName;
}

public void setLastName(String lastName) {
    this.lastName = lastName;
}

public String getEmpId() {
    return empId;
}

public void setEmpId(String empId) {
    this.empId = empId;
}

public String getDeptName() {
    return deptName;
}

public void setDeptName(String deptName) {
    this.deptName = deptName;
}

public void saveToFile(String filename) throws IOException {
    FileWriter writer = new FileWriter(filename);
    writer.write(this.firstName + "," + this.lastName + "," + this.empId + "," +
this.deptName);
    writer.close();
}

public static ArrayList<Teacher> readFromFile(String filename) throws IOException {
    ArrayList<Teacher> teachers = new ArrayList<>();
    File file = new File(filename);
    Scanner scanner = new Scanner(file);
    while (scanner.hasNextLine()) {
        String line = scanner.nextLine();
        String[] parts = line.split(",");
        Teacher teacher = new Teacher(parts[0], parts[1], parts[2], parts[3]);
    }
}

```

```

        teachers.add(teacher);
    }
    scanner.close();
    return teachers;
}

public String toString() {
    return this.firstName + " " + this.lastName;
}
}

```

---

### MainClass1.java

```

import java.io.IOException;
import java.util.ArrayList;

public class MainClass1{
    public static void main(String[] args) throws IOException {
        // Create a Teacher object
        Teacher teacher = new Teacher("Sree", "V", "1234", "CSE");

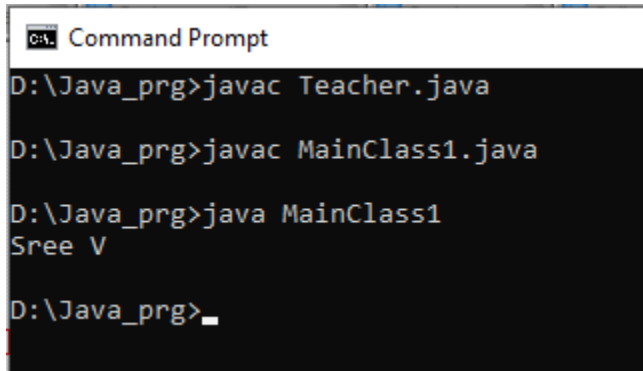
        // Save the Teacher object to a file
        teacher.saveToFile("teacher.txt");

        // Read the Teacher object from the file
        ArrayList<Teacher> teachers = Teacher.readFromFile("teacher.txt");

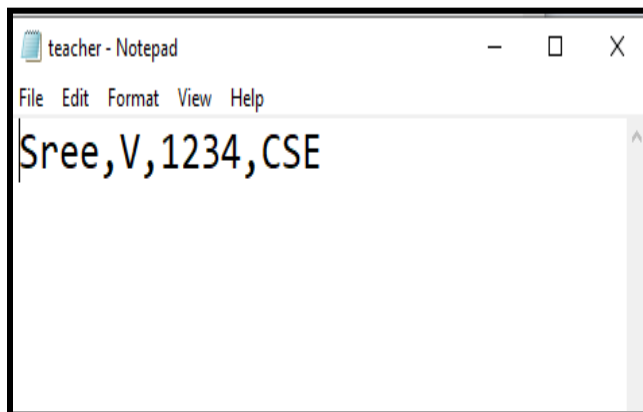
        // Print the list of Teacher objects
        for (Teacher t : teachers) {
            System.out.println(t.toString());
        }
    }
}

```

## Output:



```
Command Prompt
D:\Java_prg>javac Teacher.java
D:\Java_prg>javac MainClass1.java
D:\Java_prg>java MainClass1
Sree V
D:\Java_prg>
```



```
teacher - Notepad
File Edit Format View Help
Sree,V,1234,CSE
```

2. Create three classes Named Student, Teacher, Parents. Student and Teacher class inherits Thread class and Parent class implements Runnable interface. These three classes have run methods with statements. The task of the teacher class of the first assignment has to be synchronized. Similarly, the other two classes should have run methods with few valid statements under synchronized.

**Code:**

```
class Student extends Thread {
    public void run() {
        synchronized (System.out) {
            System.out.println("Student is attending class.");
        }
    }
}


class Teacher extends Thread {
    public void run() {
        synchronized (System.out) {
            System.out.println("Teacher is giving an assignment.");
        }
    }
}

class Parent implements Runnable {
    public void run() {
        synchronized (System.out) {
            System.out.println("Parent is attending a parent-teacher conference.");
        }
    }
}

class MainClass2{
    public static void main(String[] args) {
        Student o1 = new Student();
        Teacher o2 = new Teacher();
        Parent o3 = new Parent();

        o1.start();
        o2.start();
        o3.run();
    }
}
```

## Output:

 Command Prompt

```
D:\Java_prg>javac MainClass2.java
```

```
D:\Java_prg>java MainClass2
```

```
Parent is attending a parent-teacher conference.
```

```
Teacher is giving an assignment.
```

```
Student is attending class.
```

```
D:\Java_prg>_
```

3. Create two classes Named Student and Teacher with required data members. Assume that the information about the Student and Teacher is stored in a text file. Read n and m number of Student and Teacher information from the File. Store the information in Array list of type Student and Teacher Array List<Student> and Array List<Teacher>. Print the information of Teacher who taught OOPS and Maths. Use Iterator and other functions of util in your program.

**Code:**

```
import java.util.ArrayList;
import java.util.Scanner;
import java.io.File;
import java.io.FileNotFoundException;

class Student {
    private String name;
    private int rollNo;
    private float cgpa;

    public Student(int rollNo, String name, float cgpa) {
        this.name = name;
        this.rollNo = rollNo;
        this.cgpa = cgpa;
    }

    public String toString() {
        return "Student:[" + rollNo + " " + name + " " + cgpa + "];"
    }

}

class Teacher {
    private String name;
    private String subject;

    public Teacher(String name, String subject) {
        this.name = name;
        this.subject = subject;
    }

    public String toString() {
        return "Teacher:[" + name + " " + " " + subject + "];"
    }

}
```

```

public class MainClass3 {
    public static void main(String[] args) {
        ArrayList<Student> s = new ArrayList<Student>();
        ArrayList<Teacher> t = new ArrayList<Teacher>();
        try {
            File file = new File("students_and_teachers.txt");
            Scanner scanner = new Scanner(file);

            for(int i = 0; i < 4; i++) {
                int rollNo = scanner.nextInt();
                String name = scanner.next();
                float cgpa = scanner.nextFloat();
                Student stu = new Student(rollNo,name,cgpa);
                s.add(stu);
            }

            for (int i = 0; i < 3; i++) {
                String name = scanner.next();
                String subject = scanner.next();

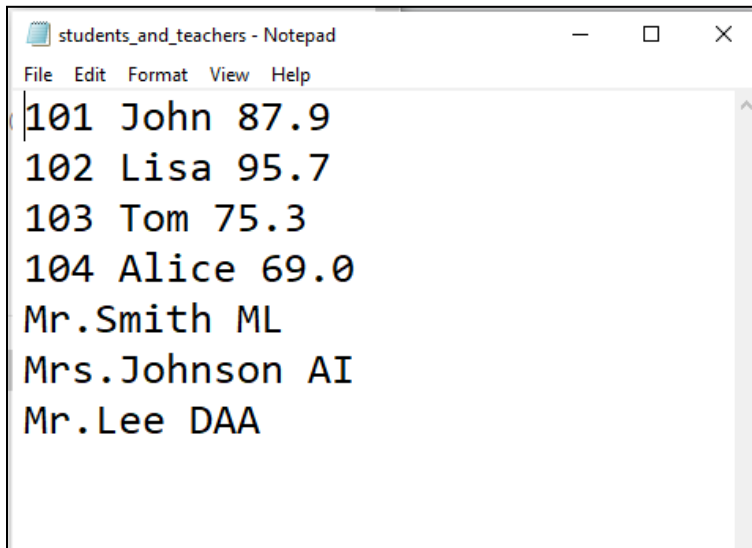
                Teacher teach = new Teacher(name, subject);
                t.add(teach);
            }
            scanner.close();
        } catch (FileNotFoundException e) {
            System.out.println("File not found.");
        }
        // Print the student and teacher lists
        System.out.println("Students:");
        for (Student s1 : s) {
            System.out.println(s1);
        }

        System.out.println("Teachers:");
        for (Teacher t1 : t) {
            System.out.println(t1);
        }
    }
}

```

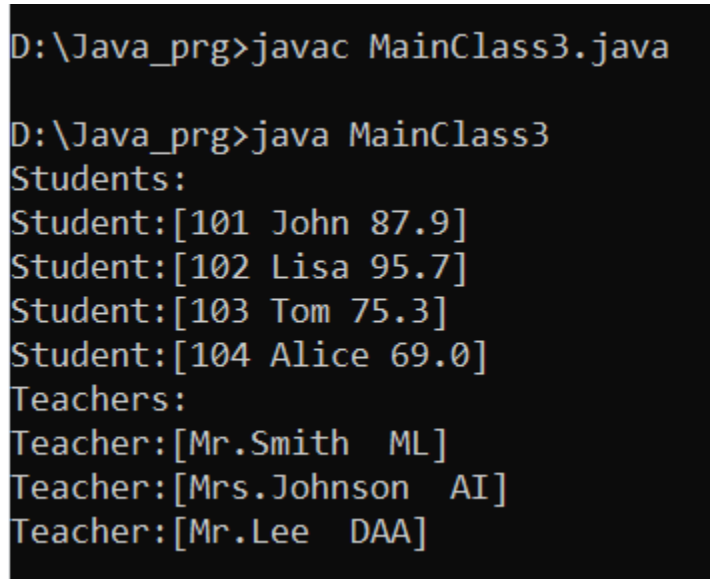


Save the file as: **students\_and\_teachers.txt**



```
students_and_teachers - Notepad
File Edit Format View Help
101 John 87.9
102 Lisa 95.7
103 Tom 75.3
104 Alice 69.0
Mr.Smith ML
Mrs.Johnson AI
Mr.Lee DAA
```

**Output:**



```
D:\Java_prg>javac MainClass3.java

D:\Java_prg>java MainClass3
Students:
Student:[101 John 87.9]
Student:[102 Lisa 95.7]
Student:[103 Tom 75.3]
Student:[104 Alice 69.0]
Teachers:
Teacher:[Mr.Smith ML]
Teacher:[Mrs.Johnson AI]
Teacher:[Mr.Lee DAA]
```

4. Watch any of the favorite movie of your choice (any language is fine, preferably English). Create a Text file to store at least 10 meaningful dialogs from the movie and store it in a text file. Process the file to remove the stop words (eg. the, is, was, ..... ) and create another file to have clean text (word).

Code:

```
import java.io.*;
import java.util.*;

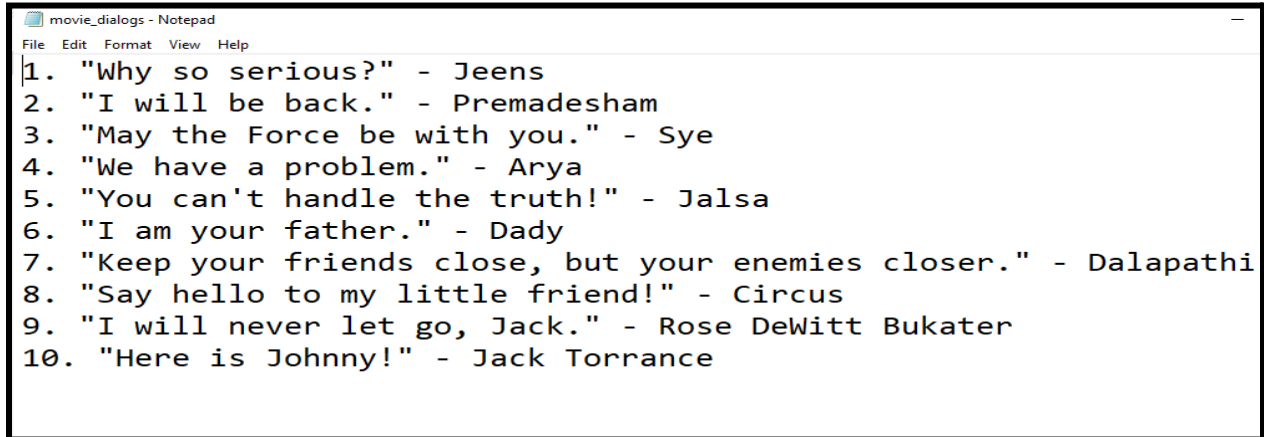
public class MainClass4 {
    public static void main(String[] args) {
        // Set of common stop words
        Set<String> stopWords = new HashSet<>(Arrays.asList("a", "an", "the", "is",
"was", "and", "or", "in", "of", "will"));

        // Input file containing movie dialogs
        String inputFileName = "movie_dialogs.txt";
        // Output file to store cleaned text
        String outputFileName = "cleaned_dialogs.txt";

        try (BufferedReader reader = new BufferedReader(new
FileReader(inputFileName));
            BufferedWriter writer = new BufferedWriter(new
FileWriter(outputFileName)))
        {
            String line;
            while ((line = reader.readLine()) != null) {
                // Split the line into words
                String[] words = line.split(" ");
                for (String word : words) {
                    // Check if the word is a stop word
                    if (!stopWords.contains(word.toLowerCase())) {
                        writer.write(word + " ");
                    }
                }
                writer.newLine();
            }
            System.out.println("cleaned_dialogs text file created successfully.");
        }
    }
}
```

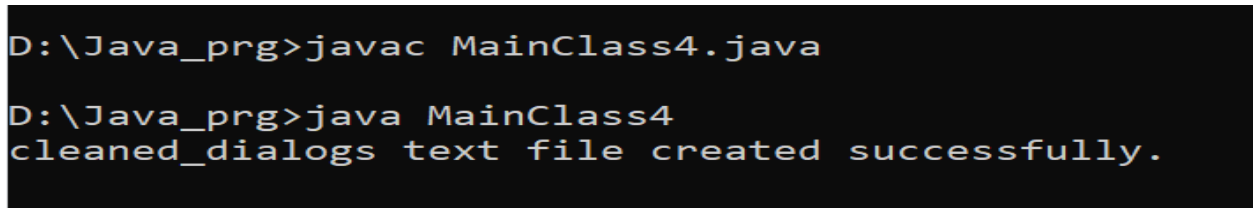
```
    }  
    catch (IOException e) {  
        e.printStackTrace();  
    }  
}  
}
```

Save the file as: **movie\_dialogs.txt**

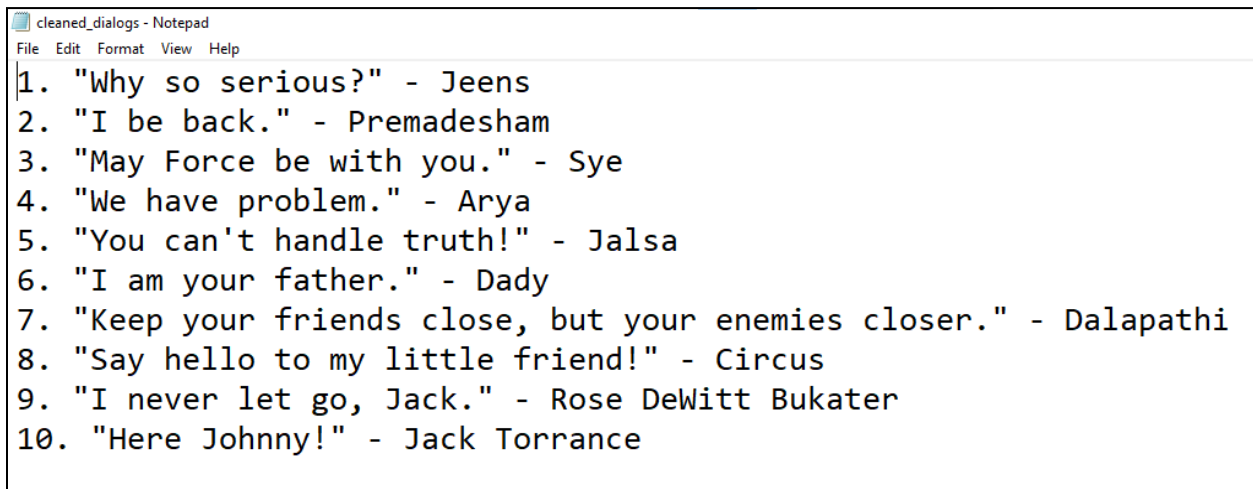


```
File Edit Format View Help  
1. "Why so serious?" - Jeens  
2. "I will be back." - Premadesham  
3. "May the Force be with you." - Sye  
4. "We have a problem." - Arya  
5. "You can't handle the truth!" - Jalsa  
6. "I am your father." - Dady  
7. "Keep your friends close, but your enemies closer." - Dalapathi  
8. "Say hello to my little friend!" - Circus  
9. "I will never let go, Jack." - Rose DeWitt Bukater  
10. "Here is Johnny!" - Jack Torrance
```

Output:



```
D:\Java_prg>javac MainClass4.java  
  
D:\Java_prg>java MainClass4  
cleaned_dialogs text file created successfully.
```



```
File Edit Format View Help  
1. "Why so serious?" - Jeens  
2. "I be back." - Premadesham  
3. "May Force be with you." - Sye  
4. "We have problem." - Arya  
5. "You can't handle truth!" - Jalsa  
6. "I am your father." - Dady  
7. "Keep your friends close, but your enemies closer." - Dalapathi  
8. "Say hello to my little friend!" - Circus  
9. "I never let go, Jack." - Rose DeWitt Bukater  
10. "Here Johnny!" - Jack Torrance
```

5. Write a java program to create Hashtable to act as a dictionary for the word collection. The dictionary meaning of the words, including synonyms, etc., has to be displayed.

**Code:**

```
import java.util.Hashtable;
```

```
public class MainClass5 {
```

```
    public static void main(String[] args) {
```

```
        // Create a Hashtable to store the words and their meanings
```

```
        Hashtable<String, String> dictionary = new Hashtable<>();
```

```
        // Add some words and their meanings to the dictionary
```

```
        dictionary.put("apple", "a round fruit with red or green skin and a white  
inside");
```

```
        dictionary.put("cloud", " a term used to describe a global network of servers,  
each with a unique function");
```

```
        dictionary.put("car", "a vehicle with four wheels that runs on roads");
```

```
        dictionary.put("computer", "an electronic device used for processing and  
storing data");
```

```
        // Add some synonyms for the words in the dictionary
```

```
        dictionary.put("apple", dictionary.get("apple") + ", also known as a fruit");
```

```
        dictionary.put("cloud", dictionary.get("cloud") + ", also called a virtual  
store");
```

```
        dictionary.put("car", dictionary.get("car") + ", also known as an automobile  
or vehicle");
```

```
        dictionary.put("computer", dictionary.get("computer") + ", also called a PC or  
a Mac");
```

```
        // Display the contents of the dictionary
```

```
        System.out.println("Dictionary:");
```

```
        for (String word : dictionary.keySet()) {
```

```
            System.out.println(word + ": " + dictionary.get(word));
```

```
        }
```

```
    }
```

```
}
```

### Output:

```
D:\Java_prg>javac MainClass5.java
```

```
D:\Java_prg>java MainClass5
```

Dictionary:

apple: a round fruit with red or green skin and a white inside, also known as a fruit

car: a vehicle with four wheels that runs on roads, also known as an automobile or vehicle

computer: an electronic device used for processing and storing data, also called a PC or a Mac

cloud: a term used to describe a global network of servers, each with a unique function, also called a virtual store

6. Write a JAVA program, (i) where the Teacher will save the marks of all the students in a file.  
(ii) Once the marks are entered, the student can view the marks

**Code:**

```
import java.io.*;
import java.util.Scanner;

class MainClass6 {
    public static void main(String[] args) {
        try{
            File f1=new File("marks.txt");
            Scanner scanner = new Scanner(f1);
        }
        catch (IOException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
        }

        try {
            // Create a FileReader object to read from the file
            FileReader reader = new FileReader("marks.txt");

            // Ask the student to view their marks
            System.out.print("Enter your roll number to view the marks: ");
            Scanner scanner = new Scanner(System.in);

            int rollNo = scanner.nextInt();
            boolean found = false;
            BufferedReader bufferedReader = new BufferedReader(reader);
            String line;
            while ((line = bufferedReader.readLine()) != null)
            {
                String[] tokens = line.split(" ");
                int studentRollNo = Integer.parseInt(tokens[0]);
                int studentMarks = Integer.parseInt(tokens[1]);
                if (studentRollNo == rollNo)
                {
                    System.out.println("Marks of roll number " + rollNo + ": " + studentMarks);
                    found = true;
                    break;
                }
            }
        }
        if (!found) {
```

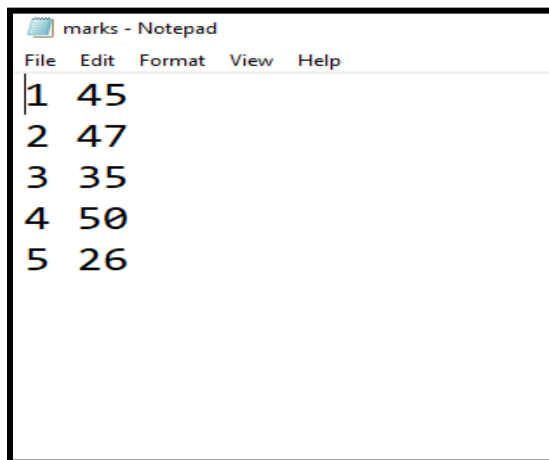
```

        System.out.println("Student with roll number " + rollNo + " not found.");
    }

    // Close the FileReader object
    reader.close();
}
catch (IOException e) {
    System.out.println("An error occurred while reading from the file.");
    e.printStackTrace();
}
}
}

```

Save marks file as: **marks.txt**



**Output:**

```

D:\Java_prg>javac MainClass6.java

D:\Java_prg>java MainClass6
Enter your roll number to view the marks: 2
Marks of roll number 2: 47

D:\Java_prg>java MainClass6
Enter your roll number to view the marks: 10
Student with roll number 10 not found.

```

**7.**

a) Create GUI for the above program to upload the dialog FILE, clean the FILE.

b) Create GUI it should take input from the user for invoking the dictionary for displaying dictionary meaning.

a) code:

```
import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

public class DialogFileProcessorGUI extends JFrame {

    private static final Set<String> STOP_WORDS = new HashSet<>();

    static {
        // Populate stop words set
        STOP_WORDS.add("the");
        STOP_WORDS.add("is");
        STOP_WORDS.add("was");
        STOP_WORDS.add("will");
        STOP_WORDS.add("a");
        STOP_WORDS.add("an");
        STOP_WORDS.add("or");
        // Add more stop words as needed
    }

    private JButton uploadButton;

    public DialogFileProcessorGUI()
    {
        setTitle("Dialog File Processor");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300, 100);
        setLocationRelativeTo(null);
    }
}
```



```

uploadButton = new JButton("Upload File");
uploadButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        JFileChooser fileChooser = new JFileChooser();
        int returnValue = fileChooser.showOpenDialog(null);
        if (returnValue == JFileChooser.APPROVE_OPTION) {
            File selectedFile = fileChooser.getSelectedFile();
            processFile(selectedFile);
        }
    }
});

getContentPane().add(uploadButton);
}

private void processFile(File file) {
    try {
        List<String> lines = Files.readAllLines(file.toPath(), StandardCharsets.UTF_8);
        StringBuilder cleanedText = new StringBuilder();

        for (String line : lines) {
            String[] words = line.split("\\s+");
            for (String word : words) {
                if (!STOP_WORDS.contains(word.toLowerCase())) {
                    cleanedText.append(word).append(" ");
                }
            }
            cleanedText.append("\n");
        }

        JFileChooser saveFileChooser = new JFileChooser();
        int saveReturnValue = saveFileChooser.showSaveDialog(null);
        if (saveReturnValue == JFileChooser.APPROVE_OPTION) {
            File saveFile = saveFileChooser.getSelectedFile();
            Path saveFilePath = saveFile.toPath();
            Files.write(saveFilePath,
cleanedText.toString().trim().getBytes(StandardCharsets.UTF_8));
            JOptionPane.showMessageDialog(null, "File processed successfully!");
        }
    }
}

```

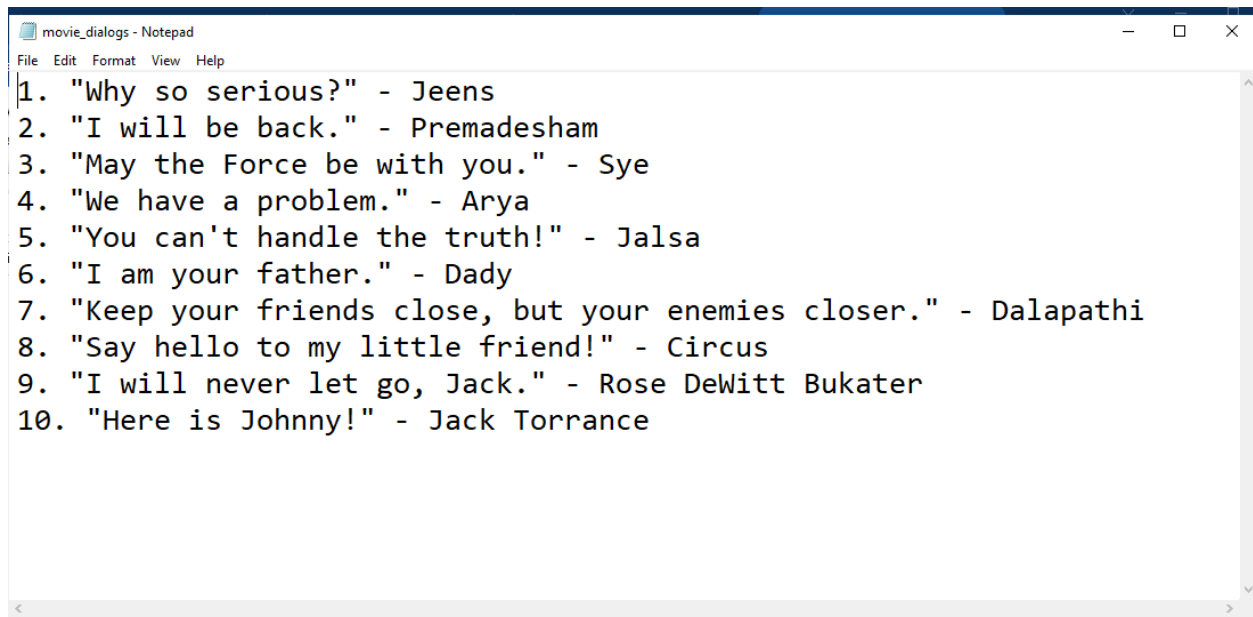
```

    } catch (IOException ex) {
        JOptionPane.showMessageDialog(null, "Error reading or writing the file.");
    }
}

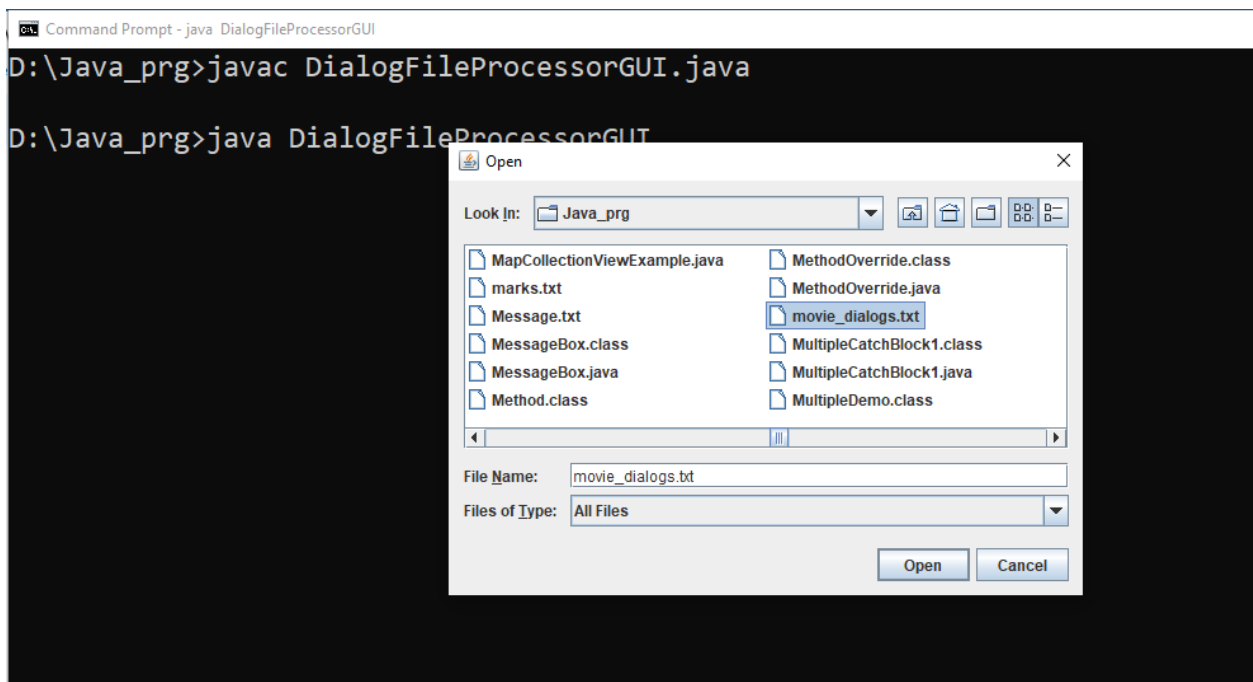
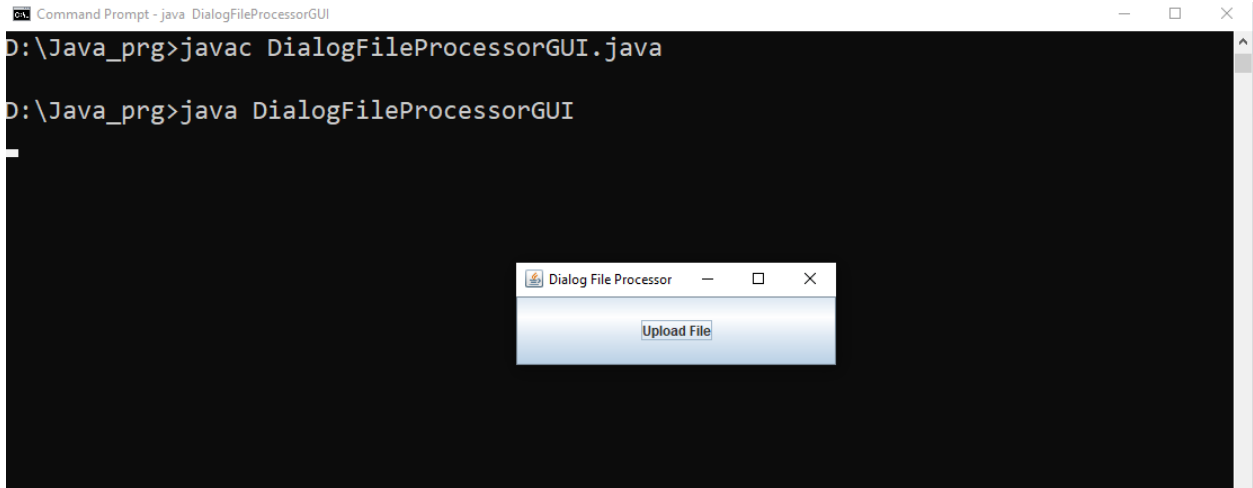
public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            new DialogFileProcessorGUI().setVisible(true);
        }
    });
}
}

```

Movie dialogs file: save as movie\_dialogs



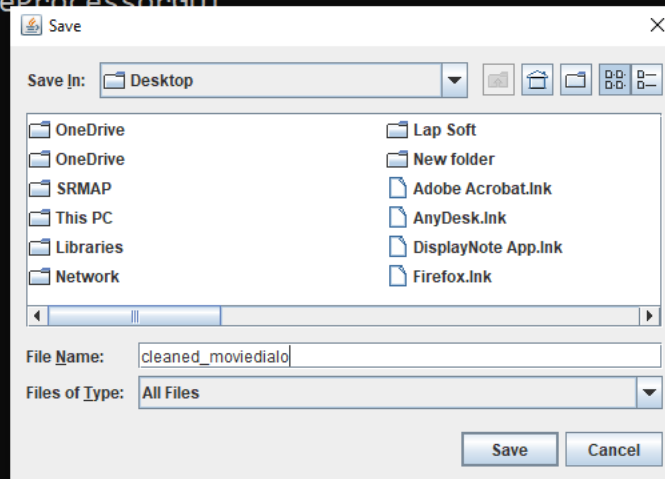
**Output:**



Command Prompt - java DialogFileProcessorGUI

```
D:\Java_prg>javac DialogFileProcessorGUI.java
```

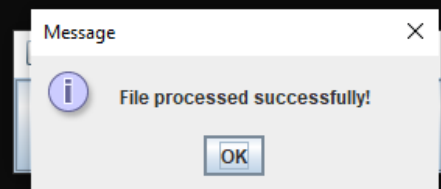
```
D:\Java_prg>java DialogFileProcessorGUI
```



Command Prompt - java DialogFileProcessorGUI

```
D:\Java_prg>javac DialogFileProcessorGUI.java
```

```
D:\Java_prg>java DialogFileProcessorGUI
```



1. "Why so serious?" - Jeens
2. "I be back." - Premadeshm
3. "May Force be with you." - Sye
4. "We have problem." - Arya
5. "You can't handle truth!" - Jalsa
6. "I am your father." - Dady
7. "Keep your friends close, but your enemies closer." - Dalapathi
8. "Say hello to my little friend!" - Circus
9. "I never let go, Jack." - Rose DeWitt Bukater
10. "Here Johnny!" - Jack Torrance

**b) Code:**

```
import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.HashMap;
import java.util.Map;

public class DictionaryGUI extends JFrame {

    private JTextArea dictionaryTextArea;
    private JTextField wordTextField;
    private JTextField meaningTextField;
    private JButton addButton;
    private JButton showDictionaryButton;

    private Map<String, String> dictionary;

    public DictionaryGUI() {
        setTitle("Dictionary");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(500, 400);
        setLocationRelativeTo(null);

        dictionary = new HashMap<>();

        JPanel panel = new JPanel();
        getContentPane().add(panel);

        dictionaryTextArea = new JTextArea(1,1);
        dictionaryTextArea.setEditable(false);
        panel.add(dictionaryTextArea);

        JLabel wordLabel = new JLabel("Word:");
        panel.add(wordLabel);

        wordTextField = new JTextField(15);
        panel.add(wordTextField);

        JLabel meaningLabel = new JLabel("Meaning:");
```

```
panel.add(meaningLabel);
```

```
meaningTextField = new JTextField(15);
```

```
panel.add(meaningTextField);
```

```
addButton = new JButton("Add to Dictionary");
```

```
addButton.addActionListener(new ActionListener() {
```

```
    @Override
```

```
    public void actionPerformed(ActionEvent e) {
```

```
        String word = wordTextField.getText().trim();
```

```
        String meaning = meaningTextField.getText().trim();
```

```
        if (!word.isEmpty() && !meaning.isEmpty()) {
```

```
            dictionary.put(word, meaning);
```

```
            wordTextField.setText("");
```

```
            meaningTextField.setText("");
```

```
            JOptionPane.showMessageDialog(null, "Word added to dictionary.");
```

```
        }
```

```
    }
```

```
});
```

```
panel.add(addButton);
```

```
showDictionaryButton = new JButton("Show Dictionary");
```

```
showDictionaryButton.addActionListener(new ActionListener() {
```

```
    @Override
```

```
    public void actionPerformed(ActionEvent e) {
```

```
        StringBuilder dictionaryContent = new StringBuilder();
```

```
        if (!dictionary.isEmpty()) {
```

```
            for (Map.Entry<String, String> entry : dictionary.entrySet()) {
```

```
                dictionaryContent.append(entry.getKey()).append(":
```

```
            ").append(entry.getValue()).append("\n");
```

```
            }
```

```
            dictionaryTextArea.setText(dictionaryContent.toString());
```

```
        } else {
```

```
            dictionaryTextArea.setText("Dictionary is empty.");
```

```
        }
```

```
    }
```

```
});
```

```
panel.add(showDictionaryButton);
```

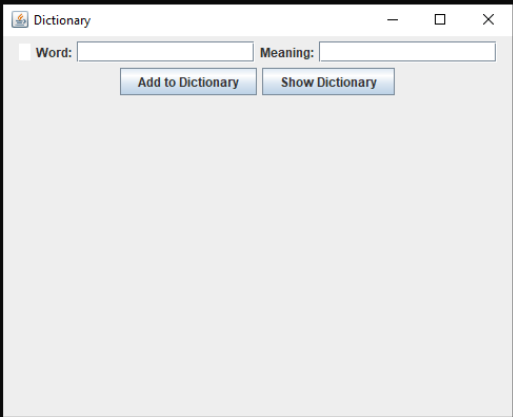
```
}
```

```
public static void main(String[] args) {  
    SwingUtilities.invokeLater(new Runnable() {  
        public void run() {  
            new DictionaryGUI().setVisible(true);  
        }  
    });  
}
```

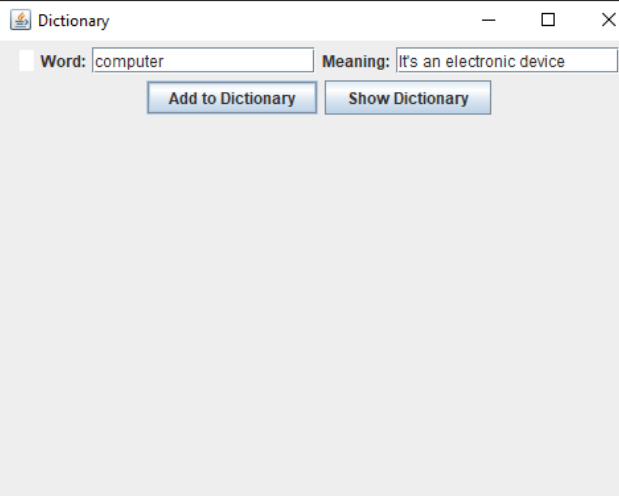


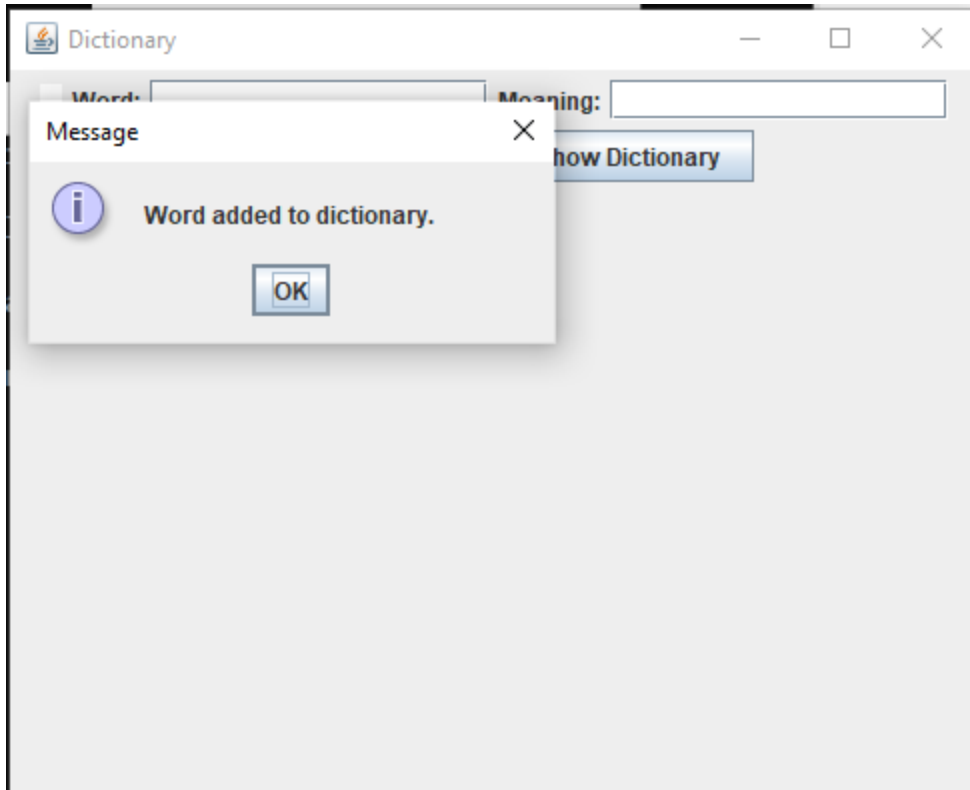
Output:

```
Command Prompt - java DictionaryGUI
D:\Java_prg>javac DialogFileProcessorGUI.java
D:\Java_prg>java DialogFileProcessorGUI
D:\Java_prg>javac DictionaryGUI.java
D:\Java_prg>java DictionaryGUI
```



```
Command Prompt - java DictionaryGUI
D:\Java_prg>javac DialogFileProcessorGUI.java
D:\Java_prg>java DialogFileProcessorGUI
D:\Java_prg>javac DictionaryGUI.java
D:\Java_prg>java DictionaryGUI
```





8. Declare a class Named Teacher. The class will have all the data members as per your convenient. The class will have constructors. Develop a GUI to read the values of the class variables from the keyboard. Use text field to read the values. Use button to store it in a file one by one. The values will be stored in a structured format of your own choice.

**Teacher1.java**

```
public class Teacher1 {  
    private String rollNumber;  
    private String name;  
    private String subject;  
  
    public Teacher1(String rollNumber, String name, String subject) {  
        this.rollNumber = rollNumber;  
        this.name = name;  
        this.subject = subject;  
    }  
  
    public String getRollNumber() {  
        return rollNumber;  
    }  
  
    public String toString() {  
        return "Roll Number: " + rollNumber + "\n"  
            + "Name: " + name + "\n"  
            + "Subject: " + subject;  
    }  
}
```

## TeacherGUI.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;

public class TeacherGUI extends JFrame {

    private JTextField rollNumberField;
    private JTextField nameField;
    private JTextField subjectField;
    private JTextArea outputArea;

    private ArrayList<Teacher1> teachers;

    public TeacherGUI() {
        teachers = new ArrayList<>();

        setTitle("Teacher Information");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(400, 300);
        setLocationRelativeTo(null);

        JPanel inputPanel = new JPanel(new GridLayout(4, 2));

        JLabel rollNumberLabel = new JLabel("Roll Number:");
        rollNumberField = new JTextField();
        JLabel nameLabel = new JLabel("Name:");
        nameField = new JTextField();
        JLabel subjectLabel = new JLabel("Subject:");
        subjectField = new JTextField();

        inputPanel.add(rollNumberLabel);
        inputPanel.add(rollNumberField);
        inputPanel.add(nameLabel);
        inputPanel.add(nameField);
```

```
inputPanel.add(subjectLabel);
inputPanel.add(subjectField);
```

```
JButton saveButton = new JButton("Save");
saveButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        saveTeacher();
    }
});
```

```
JPanel buttonPanel = new JPanel();
buttonPanel.add(saveButton);
```

```
outputArea = new JTextArea();
outputArea.setEditable(false);
```

```
getContentPane().setLayout(new BorderLayout());
getContentPane().add(inputPanel, BorderLayout.NORTH);
getContentPane().add(buttonPanel, BorderLayout.CENTER);
getContentPane().add(new JScrollPane(outputArea), BorderLayout.SOUTH);
}
```

```
private void saveTeacher() {
    String rollNumber = rollNumberField.getText();
    String name = nameField.getText();
    String subject = subjectField.getText();

    Teacher1 teacher = new Teacher1(rollNumber, name, subject);
    teachers.add(teacher);

    try (FileWriter writer = new FileWriter("teacher_data.txt", true)) {
        writer.write(teacher.toString() + "\n");
        writer.close();
    } catch (IOException e) {
        e.printStackTrace();
    }

    rollNumberField.setText("");
    nameField.setText("");
    subjectField.setText("");
}
```

```

    }

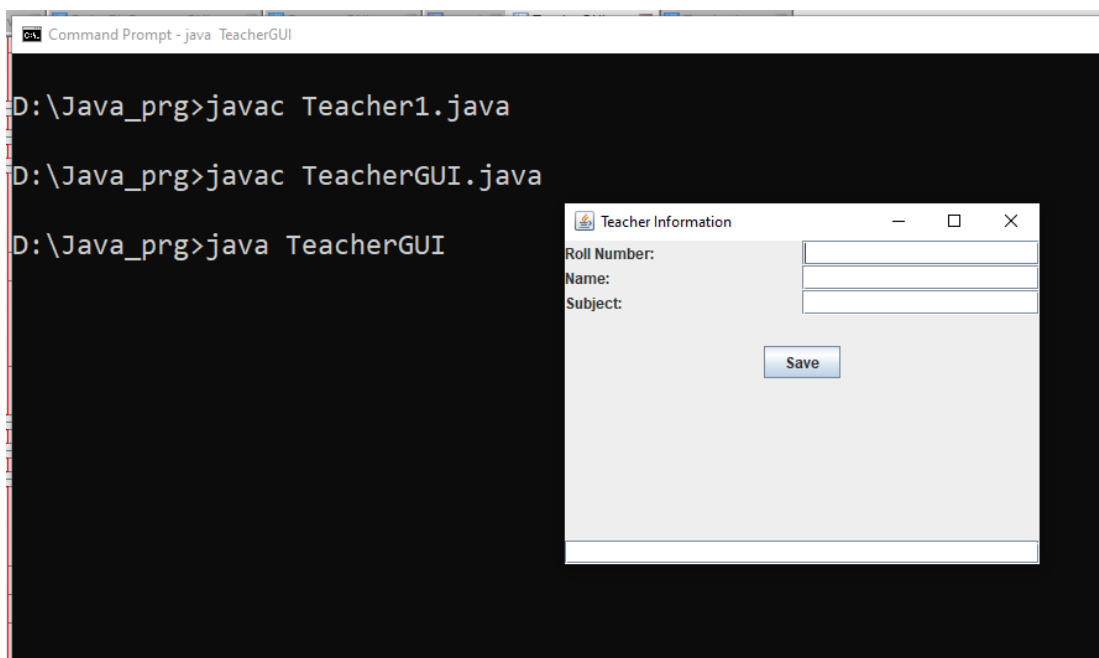
    private void searchTeacher(String rollNumber) {
        for (Teacher1 teacher : teachers) {
            if (teacher.getRollNumber().equals(rollNumber)) {
                outputArea.setText(teacher.toString());
                return;
            }
        }

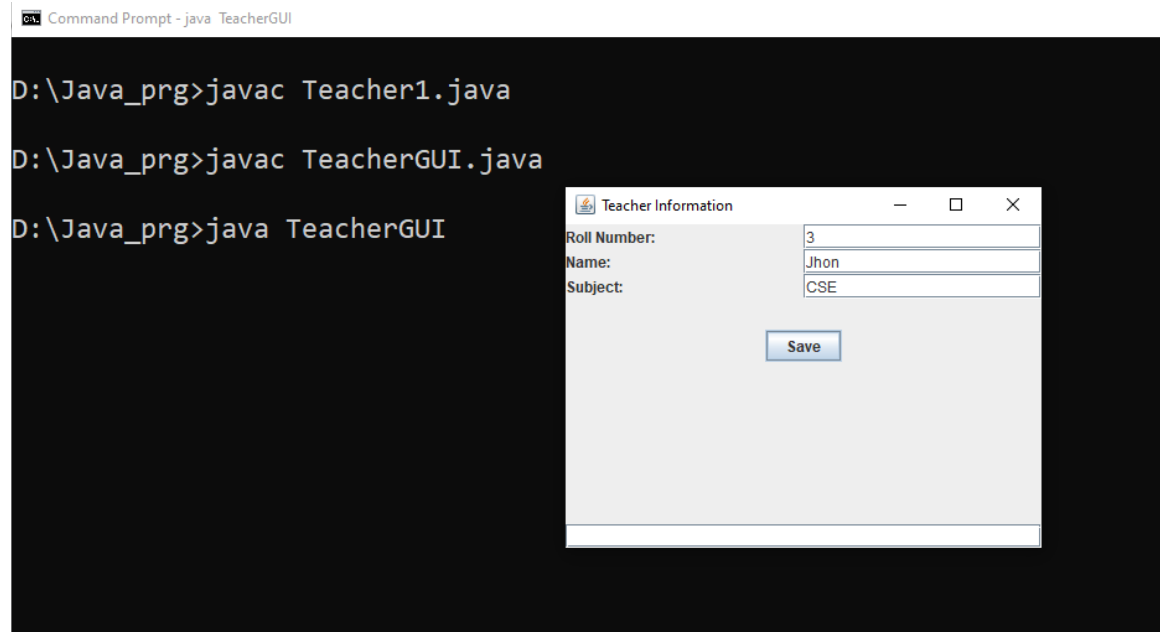
        outputArea.setText("Teacher not found for roll number: " + rollNumber);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                TeacherGUI teacherGUI = new TeacherGUI();
                teacherGUI.setVisible(true);
            }
        });
    }
}

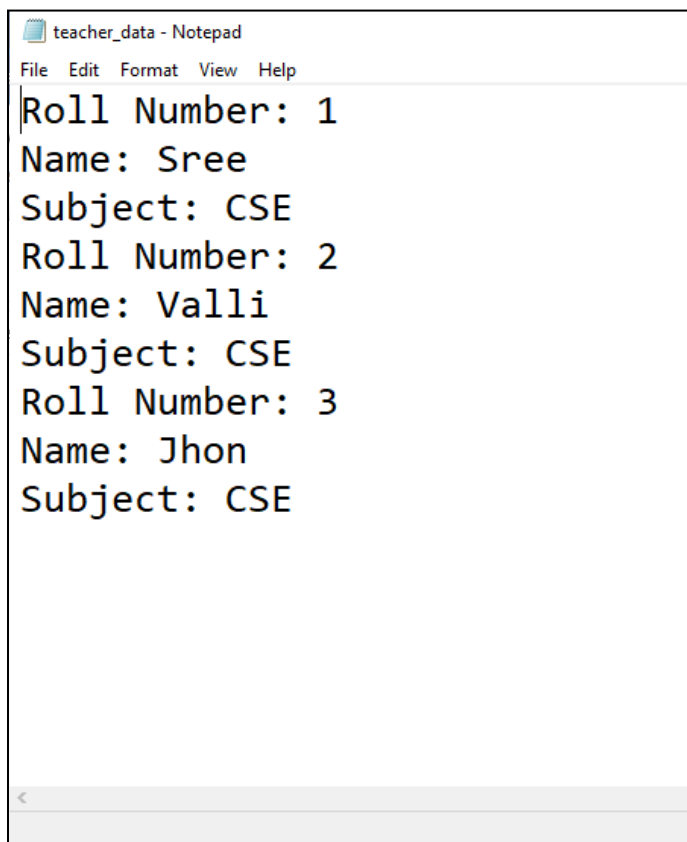
```

Output:





File: teacher\_data.txt



9. Create two classes Named Student and Teacher with required data members. Read the information about the student and teacher using text fields. Use the checkbox to choose the option to feed either teacher information or student information. Store the information about the Student and Teacher in a text file. Read n and m number of Student and Teacher information from the File. Show in the GUI about a Teacher who taught two subjects to a section. Develop at least one of the applications (AWT problem) using the swing package.

#### Teacher1.java

```
public class Teacher1 {
    private String rollNumber;
    private String name;
    private String subject;

    public Teacher1(String rollNumber, String name, String subject) {
        this.rollNumber = rollNumber;
        this.name = name;
        this.subject = subject;
    }

    public String getRollNumber() {
        return rollNumber;
    }

    public String toString() {
        return "Roll Number: " + rollNumber + "\n"
            + "Name: " + name + "\n"
            + "Subject: " + subject;
    }
}
```



## Student1.java

```
public class Student1 {  
    private String rollNumber;  
    private String name;  
    private String subject;  
  
    public Student1(String rollNumber, String name, String subject) {  
        this.rollNumber = rollNumber;  
        this.name = name;  
        this.subject = subject;  
    }  
  
    public String getRollNumber() {  
        return rollNumber;  
    }  
  
    public String toString() {  
        return "Roll Number: " + rollNumber + "\n"  
            + "Name: " + name + "\n"  
            + "Subject: " + subject;  
    }  
}
```

## InformationGUI.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.FileWriter;
import java.io.IOException;

public class InformationGUI extends JFrame {

    private JCheckBox teacherCheckbox;
    private JTextField rollNumberField;
    private JTextField nameField;
    private JTextField subjectField;
    private JButton saveButton;

    public InformationGUI() {
        setTitle("Student and Teacher Information");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(400, 300);
        setLocationRelativeTo(null);

        JPanel inputPanel = new JPanel(new GridLayout(4, 2));

        JLabel rollNumberLabel = new JLabel("Roll Number:");
        rollNumberField = new JTextField();
        JLabel nameLabel = new JLabel("Name:");
        nameField = new JTextField();
        JLabel subjectLabel = new JLabel("Subject:");
        subjectField = new JTextField();

        inputPanel.add(rollNumberLabel);
        inputPanel.add(rollNumberField);
        inputPanel.add(nameLabel);
        inputPanel.add(nameField);
        inputPanel.add(subjectLabel);
        inputPanel.add(subjectField);

        teacherCheckbox = new JCheckBox("Teacher Information");
```

```

saveButton = new JButton("Save");
saveButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        saveInformation();
    }
});

JPanel checkboxPanel = new JPanel();
checkboxPanel.add(teacherCheckbox);

JPanel buttonPanel = new JPanel();
buttonPanel.add(saveButton);

getContentPane().setLayout(new BorderLayout());
getContentPane().add(inputPanel, BorderLayout.NORTH);
getContentPane().add(checkboxPanel, BorderLayout.CENTER);
getContentPane().add(buttonPanel, BorderLayout.SOUTH);
}

private void saveInformation() {
    String rollNumber = rollNumberField.getText();
    String name = nameField.getText();
    String subject = subjectField.getText();

    if (teacherCheckbox.isSelected()) {
        Teacher1 teacher = new Teacher1(rollNumber, name, subject);
        writeToFile(teacher.toString());
    } else {
        Student1 student = new Student1(rollNumber, name, subject);
        writeToFile(student.toString());
    }

    rollNumberField.setText("");
    nameField.setText("");
    subjectField.setText("");
}

private void writeToFile(String content) {
    try (FileWriter writer = new FileWriter("information.txt", true)) {

```

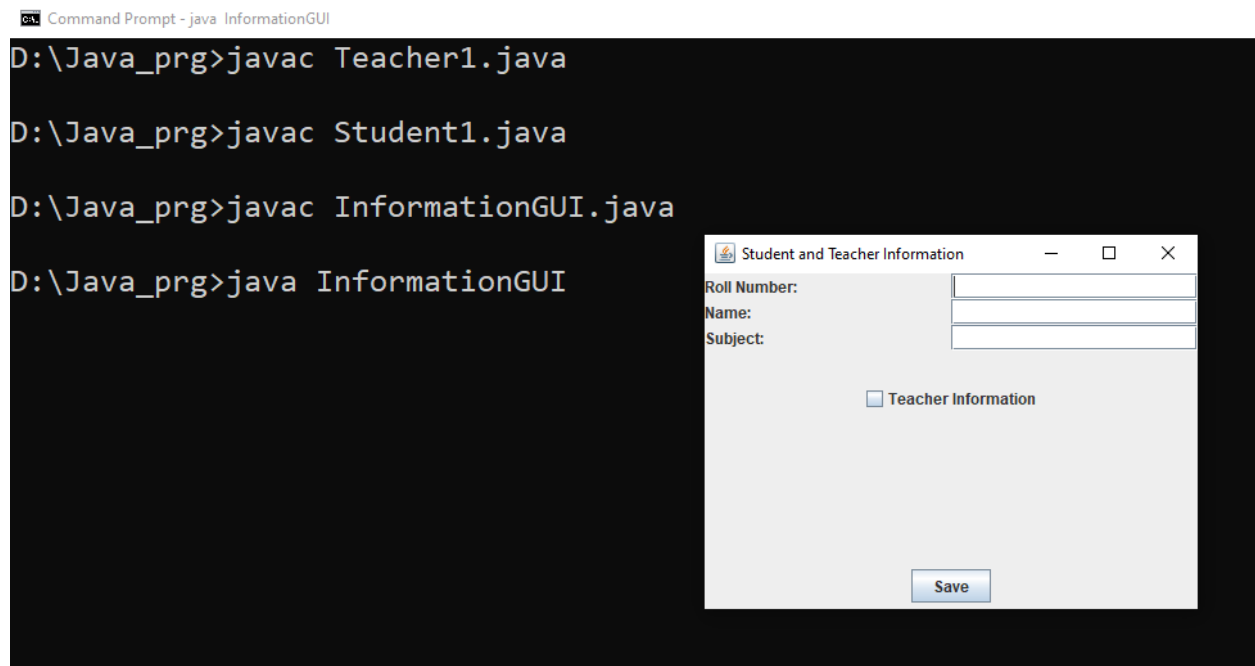
```

        writer.write(content + "\n");
    } catch (IOException e) {
        e.printStackTrace();
    }
}

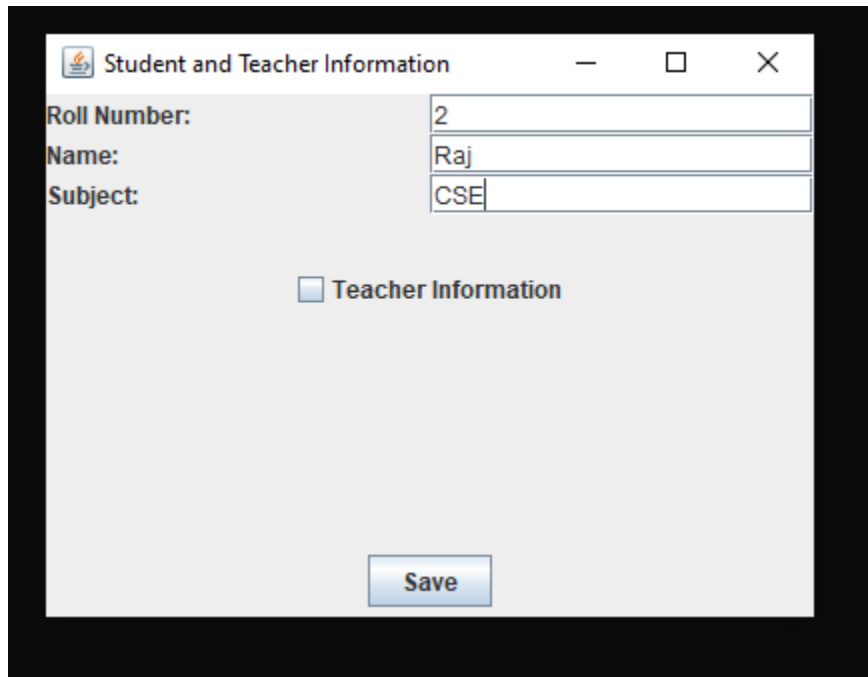
public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            InformationGUI informationGUI = new InformationGUI();
            informationGUI.setVisible(true);
        }
    });
}
}

```

## Output:



## Adding student information



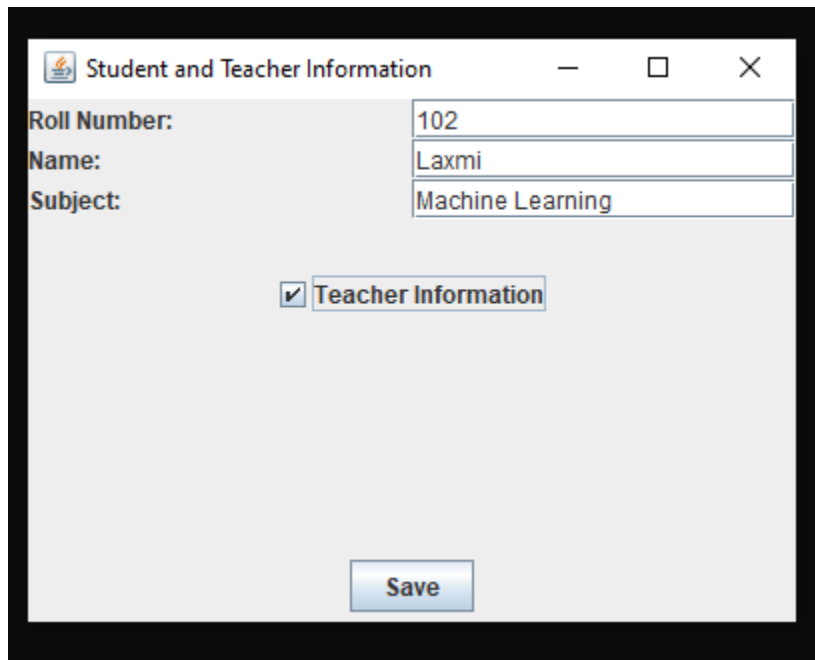
A Java Swing window titled "Student and Teacher Information" with standard window controls (minimize, maximize, close). The window contains three text input fields for student information: "Roll Number:" with the value "2", "Name:" with the value "Raj", and "Subject:" with the value "CSE". Below these fields is a checkbox labeled "Teacher Information" which is currently unchecked. At the bottom center of the window is a "Save" button.

Roll Number:	2
Name:	Raj
Subject:	CSE

☐ Teacher Information

Save

## Adding Teacher information



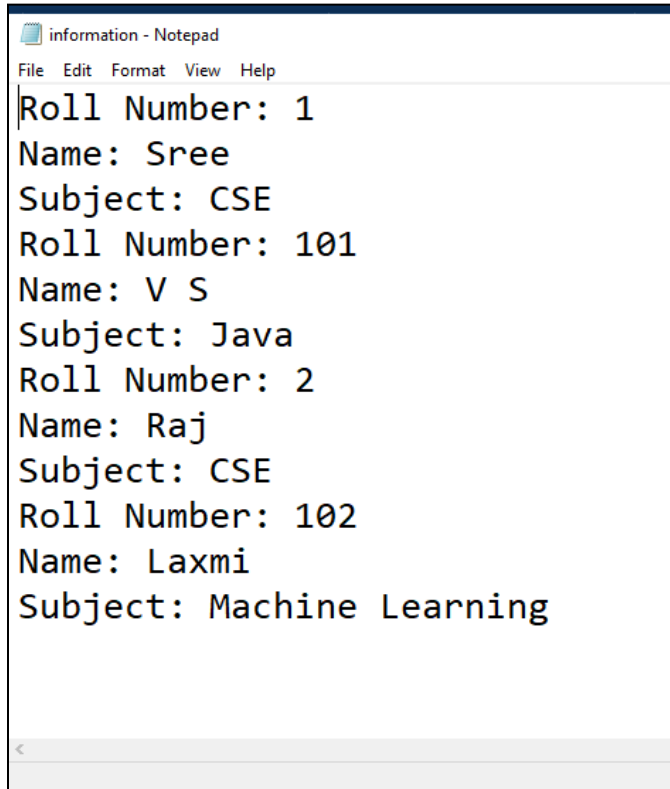
A Java Swing window titled "Student and Teacher Information" with standard window controls (minimize, maximize, close). The window contains three text input fields for teacher information: "Roll Number:" with the value "102", "Name:" with the value "Laxmi", and "Subject:" with the value "Machine Learning". Below these fields is a checkbox labeled "Teacher Information" which is currently checked. At the bottom center of the window is a "Save" button.

Roll Number:	102
Name:	Laxmi
Subject:	Machine Learning

☒ Teacher Information

Save

**File: information.txt**



The image shows a screenshot of a Notepad application window. The title bar reads "information - Notepad". The menu bar includes "File", "Edit", "Format", "View", and "Help". The text area contains the following content:

```
Roll Number: 1  
Name: Sree  
Subject: CSE  
Roll Number: 101  
Name: V S  
Subject: Java  
Roll Number: 2  
Name: Raj  
Subject: CSE  
Roll Number: 102  
Name: Laxmi  
Subject: Machine Learning
```

A horizontal scrollbar is visible at the bottom of the text area.

**10. Create a Window based application using various controls to handle subject registration for exams. Have a List Box to display the subject of semesters. Have one more List box having subject codes. Have a combo box to select the Semester, which will change the list of course and code in the list boxes. Display the subject registered for the examination on the right side of the window.**

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

public class SubjectRegistrationApp extends JFrame {
    private String[] semesters = {"Semester 5", "Semester 6", "Semester 7"};
    private String[][] subjects = {
        {"ML", "Cloud", "Network Security"},
        {"AI", "Big Data", "Android Dev"},
        {"CNN", "Mobile Networking", "IoT"}
    };

    private JList<String> subjectListBox;
    private JComboBox<String> semesterComboBox;
    private JButton submitButton;
    private JTextArea registrationTextArea;

    public SubjectRegistrationApp() {
        // Set up the main frame
        setTitle("Subject Registration");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());
        setSize(400, 300);

        // Create the subject list box
        subjectListBox = new JList<>();

        // Create the semester combo box
        semesterComboBox = new JComboBox<>(semesters);
        semesterComboBox.addActionListener(e -> updateSubjectList());

        // Create the submit button
        submitButton = new JButton("Submit");
```

```

submitButton.addActionListener(e -> registerSubjects());

// Create the registration text area
registrationTextArea = new JTextArea();

// Add the controls to the frame
JPanel leftPanel = new JPanel(new BorderLayout());
leftPanel.add(new JScrollPane(subjectListBox), BorderLayout.CENTER);

JPanel rightPanel = new JPanel(new BorderLayout());
rightPanel.add(semesterComboBox, BorderLayout.NORTH);
rightPanel.add(submitButton, BorderLayout.SOUTH);
rightPanel.add(new JScrollPane(registrationTextArea), BorderLayout.CENTER);

add(leftPanel, BorderLayout.CENTER);
add(rightPanel, BorderLayout.EAST);
}

private void updateSubjectList() {
    int selectedSemesterIndex = semesterComboBox.getSelectedIndex();
    subjectListBox.setListData(subjects[selectedSemesterIndex]);
}

private void registerSubjects() {
    String[] selectedSubjects = subjectListBox.getSelectedValuesList().toArray(new
String[0]);
    registrationTextArea.setText("Registered subjects:\n" + String.join(", ",
selectedSubjects));
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> {
        SubjectRegistrationApp app = new SubjectRegistrationApp();
        app.setVisible(true);
    });
}
}

```

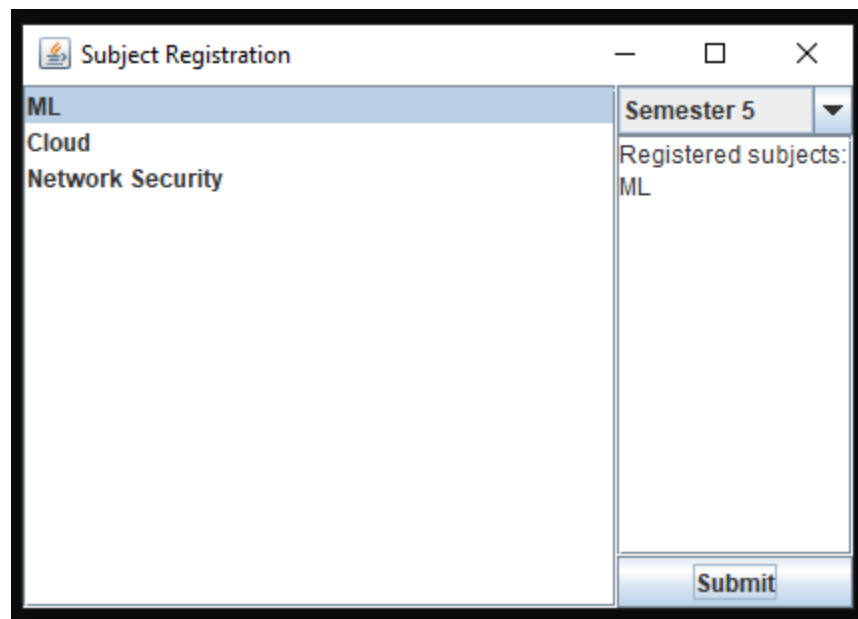
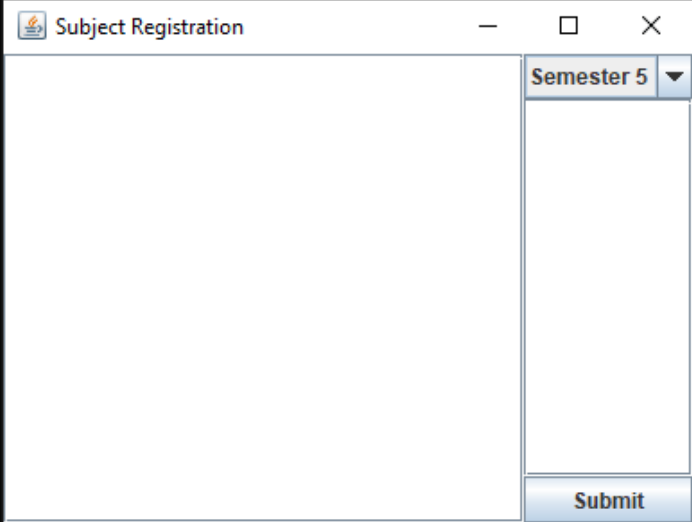



Output:

```
Command Prompt - java SubjectRegistrationApp

D:\Java_prg>javac SubjectRegistrationApp.java

D:\Java_prg>java SubjectRegistrationApp
```



 Subject Registration

AI

Big Data

Android Dev

Semester 6

Registered subjects:  
AI, Big Data

Submit

**11.** Declare a class Named Teacher. The class will have all the data members as per your convenience. The class will have constructors. Develop a GUI to read the values of the class variables from the keyboard. Use a text field to read the values. Use the button to store it in a file one by one. The values will be stored in a structured format of your own choice. Have an option in the GUI to search the Name of the students by roll number and display the content in the text field. Develop at least one of the applications (AWT problem) using the swing package.

Code:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

class Teacher {
    private int rollNumber;
    private String name;
    private int age;

    public Teacher(int rollNumber, String name, int age) {
        this.rollNumber = rollNumber;
        this.name = name;
        this.age = age;
    }

    public int getRollNumber() {
        return rollNumber;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }
}
```

```

    }

    public String toString() {
        return "Roll Number: " + rollNumber + "\nName: " + name + "\nAge: " + age + "\n";
    }
}

public class TeacherGUI1 extends JFrame {
    private JTextField rollNumberField;
    private JTextField nameField;
    private JTextField ageField;
    private JTextArea outputTextArea;

    private List<Teacher> teachers;

    public TeacherGUI1() {
        teachers = new ArrayList<>();

        setTitle("Teacher Information");
        setSize(400, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        JPanel inputPanel = new JPanel();
        inputPanel.setLayout(new GridLayout(4, 2));
        inputPanel.add(new JLabel("Roll Number:"));
        rollNumberField = new JTextField();
        inputPanel.add(rollNumberField);
        inputPanel.add(new JLabel("Name:"));
        nameField = new JTextField();
        inputPanel.add(nameField);
        inputPanel.add(new JLabel("Age:"));
        ageField = new JTextField();
        inputPanel.add(ageField);
        JButton addButton = new JButton("Add Student");
        inputPanel.add(addButton);

        outputTextArea = new JTextArea();
        outputTextArea.setEditable(false);
        JScrollPane scrollPane = new JScrollPane(outputTextArea);

```

```

JPanel searchPanel = new JPanel();
searchPanel.setLayout(new FlowLayout());
searchPanel.add(new JLabel("Search Roll Number:"));
JTextField searchRollNumberField = new JTextField();
searchRollNumberField.setPreferredSize(new Dimension(100, 25));
searchPanel.add(searchRollNumberField);
JButton searchButton = new JButton("Search");
searchPanel.add(searchButton);

add(inputPanel, BorderLayout.NORTH);
add(scrollPane, BorderLayout.CENTER);
add(searchPanel, BorderLayout.SOUTH);

addButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        addTeacher();
    }
});

searchButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        searchTeacherByRollNumber(searchRollNumberField.getText());
    }
});
}

private void addTeacher() {
    int rollNumber = Integer.parseInt(rollNumberField.getText());
    String name = nameField.getText();
    int age = Integer.parseInt(ageField.getText());

    Teacher teacher = new Teacher(rollNumber, name, age);
    teachers.add(teacher);
    outputTextArea.append(teacher.toString());
    clearInputFields();
    writeToFile(teacher);
}

```

```

private void searchTeacherByRollNumber(String rollNumber) {
    try {
        int searchRollNumber = Integer.parseInt(rollNumber);
        for (Teacher teacher : teachers) {
            if (teacher.getRollNumber() == searchRollNumber) {
                outputTextArea.setText(teacher.toString());
                return;
            }
        }
        outputTextArea.setText("No student found with Roll Number: " +
searchRollNumber);
    } catch (NumberFormatException e) {
        outputTextArea.setText("Invalid Roll Number!");
    }
}

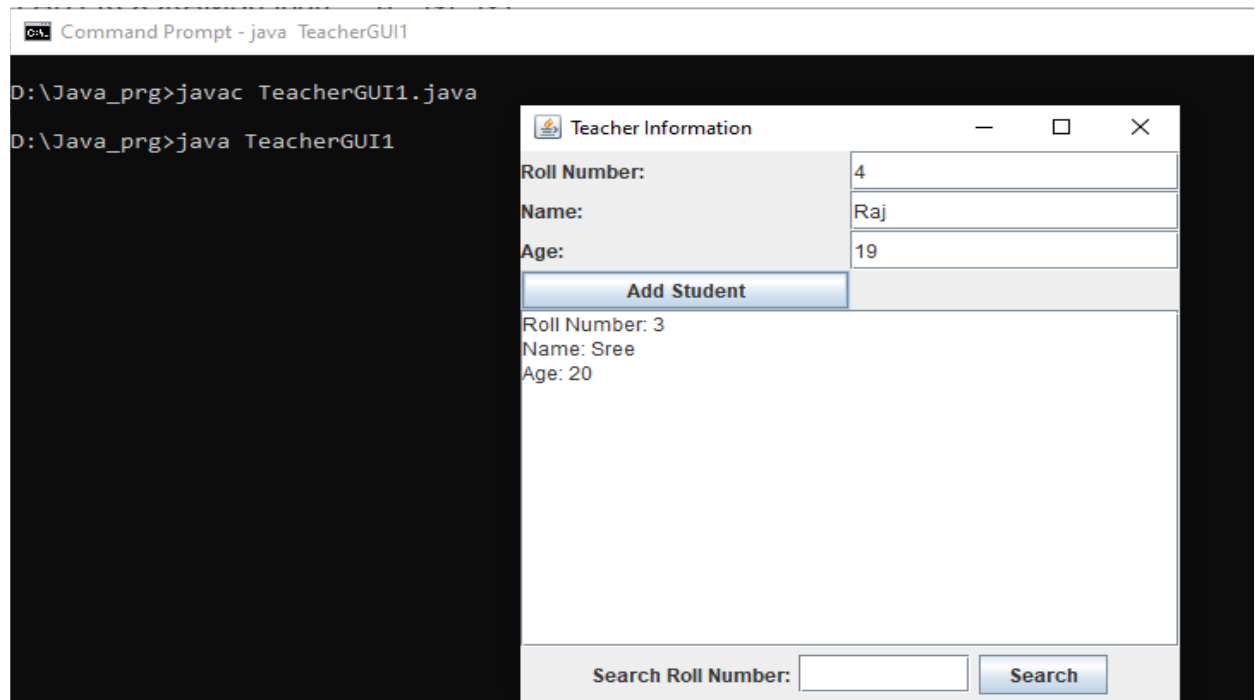
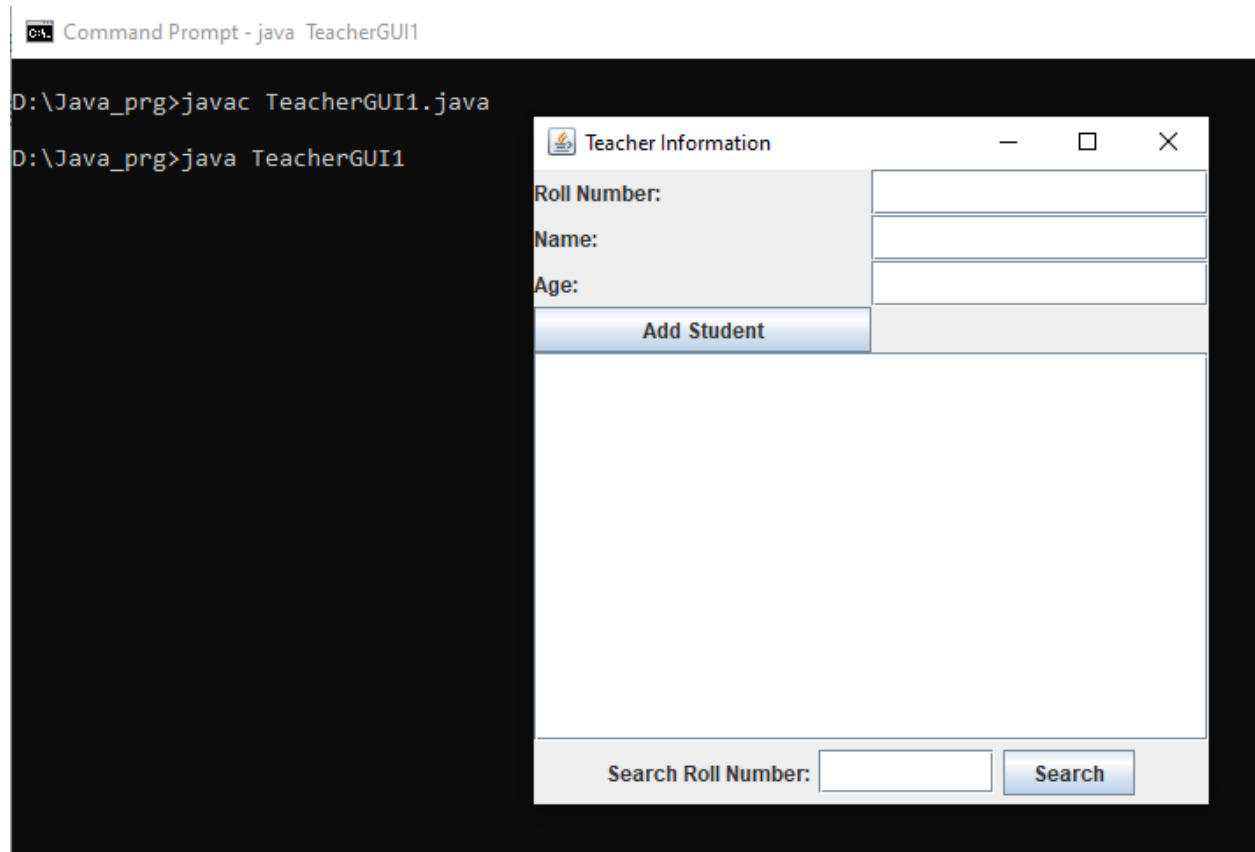
private void clearInputFields() {
    rollNumberField.setText("");
    nameField.setText("");
    ageField.setText("");
}

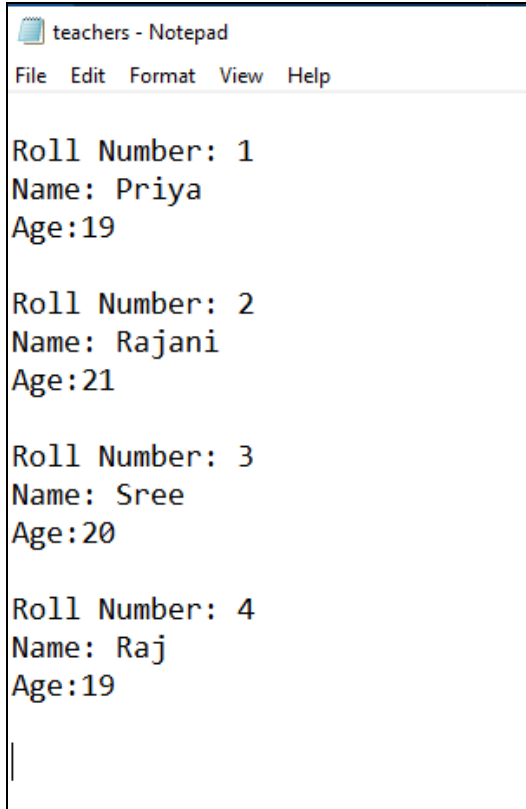
private void writeToFile(Teacher teacher) {
    try (BufferedWriter writer = new BufferedWriter(new FileWriter("teachers.txt", true)))
{
        writer.write(teacher.toString());
        writer.newLine();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            TeacherGUI1 teacherGUI = new TeacherGUI1();
            teacherGUI.setVisible(true);
        }
    });
}
}

```

Output:







12. Create a Window based application for displaying your photo album. Create a Frame and Canvas. Change the border, foreground and background colors of canvas and other controls. Have buttons to start the image show, pause the image show and end the image show. Explore the options to play background music.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;
import javax.sound.sampled.*;

public class PhotoAlbumApp {
    private JFrame frame;
    private JPanel canvas;
    private JButton startButton;
    private JButton pauseButton;
    private JButton endButton;
    private Clip musicClip;
    private JLabel imageLabel;
    private Timer slideshowTimer;
    private int currentIndex;
    private File[] imageFiles;

    public PhotoAlbumApp() {
        frame = new JFrame("Photo Album");
        frame.setSize(500,400);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(new BorderLayout());

        canvas = new JPanel();
        canvas.setBackground(Color.WHITE);
        canvas.setBorder(BorderFactory.createLineBorder(Color.BLACK, 2));
        frame.add(canvas, BorderLayout.CENTER);

        startButton = new JButton("Start");
        pauseButton = new JButton("Pause");
```

```

endButton = new JButton("End");

JPanel buttonPanel = new JPanel();
buttonPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
buttonPanel.add(startButton);
buttonPanel.add(pauseButton);
buttonPanel.add(endButton);
frame.add(buttonPanel, BorderLayout.SOUTH);

// Set foreground and background colors of controls
startButton.setForeground(Color.BLUE);
startButton.setBackground(Color.WHITE);
pauseButton.setForeground(Color.RED);
pauseButton.setBackground(Color.WHITE);
endButton.setForeground(Color.GREEN);
endButton.setBackground(Color.WHITE);

// Set border color of canvas
canvas.setBorder(BorderFactory.createLineBorder(Color.BLUE, 2));

// Button actions
startButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        startSlideshow();
    }
});

pauseButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        pauseSlideshow();
    }
});

endButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        endSlideshow();
    }
});

```

```

});

// Play background music
try {
    File musicFile = new File("background_music.wav");
    AudioInputStream audioStream =
AudioSystem.getAudioInputStream(musicFile);
    AudioFormat format = audioStream.getFormat();
    DataLine.Info info = new DataLine.Info(Clip.class, format);
    musicClip = (Clip) AudioSystem.getLine(info);
    musicClip.open(audioStream);
    musicClip.loop(Clip.LOOP_CONTINUOUSLY);
} catch (UnsupportedAudioFileException | IOException | LineUnavailableException
ex) {
    ex.printStackTrace();
}
}

private void startSlideshow() {
    currentIndex = 0;
    displayImage();
    currentIndex++;

    ActionListener slideshowAction = new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if (currentIndex < imageFiles.length) {
                displayImage();
                currentIndex++;
            } else {
                endSlideshow();
            }
        }
    };

    int delay = 2000; // Delay between image changes in milliseconds
    slideshowTimer = new Timer(delay, slideshowAction);
    slideshowTimer.start();
}

```

```

private void displayImage() {
    try {
        Image image = ImageIO.read(imageFiles[currentIndex]);
        image = image.getScaledInstance(canvas.getWidth(), canvas.getHeight(),
Image.SCALE_SMOOTH);
        imageLabel.setIcon(new ImageIcon(image));
    } catch (IOException ex) {
        ex.printStackTrace();
    }
}

private void pauseSlideshow() {
    if (slideshowTimer != null && slideshowTimer.isRunning()) {
        slideshowTimer.stop();
    }
}

private void endSlideshow() {
    if (slideshowTimer != null && slideshowTimer.isRunning()) {
        slideshowTimer.stop();
    }
    musicClip.stop(); // Stop background music
    frame.dispose(); // Close the application
}

public void display() {
    File imageDirectory = new File("images");
    imageFiles = imageDirectory.listFiles();

    if (imageFiles != null && imageFiles.length > 0) {
        imageLabel = new JLabel();
        canvas.add(imageLabel);
    } else {
        JLabel noImagesLabel = new JLabel("No images found");
        canvas.add(noImagesLabel);
    }

    frame.setVisible(true);
}

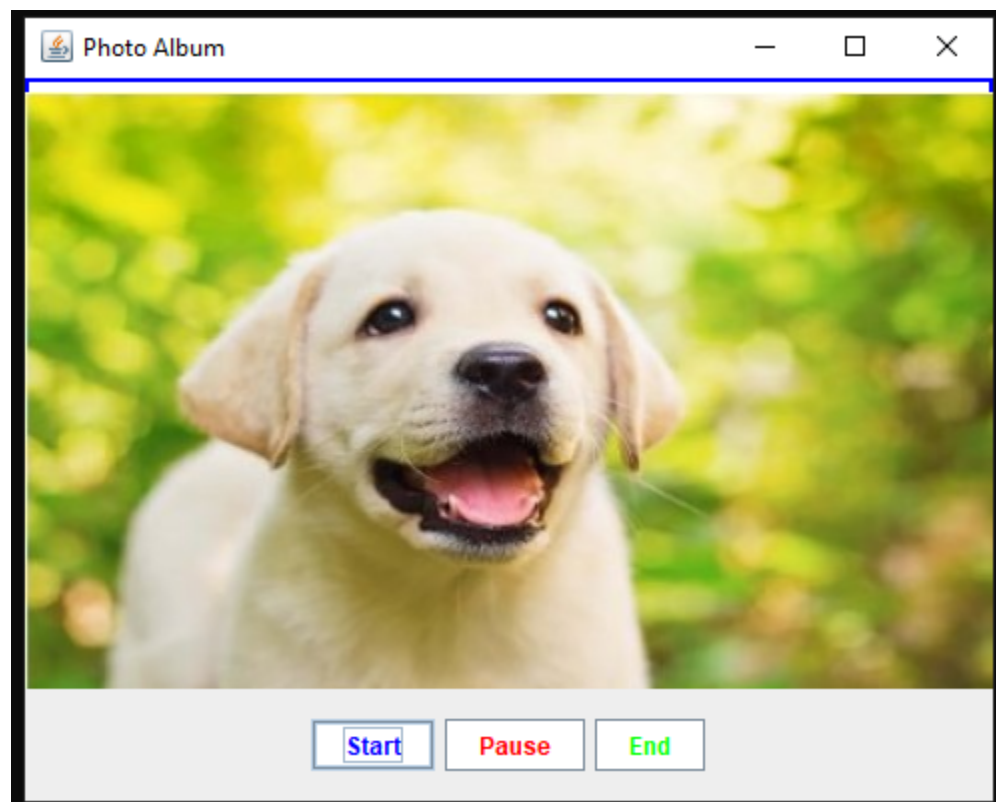
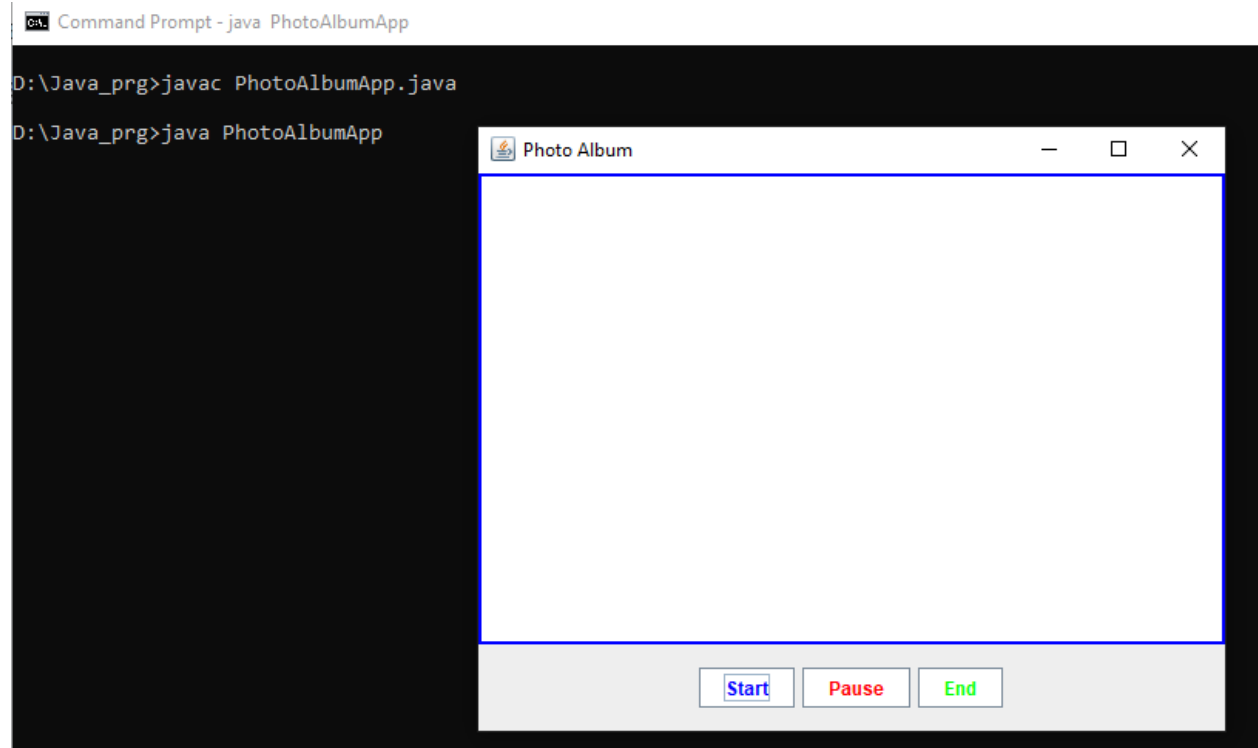
```

```
public static void main(String[] args) {  
    SwingUtilities.invokeLater(new Runnable() {  
        @Override  
        public void run() {  
            PhotoAlbumApp app = new PhotoAlbumApp();  
            app.display();  
        }  
    });  
}
```

Output:

Note:

1. Place the image files inside the "images" folder within the same directory as your Java source file. The code will load the images from that folder and display them in the slideshow. If no images are found, a "No images found" message will be displayed.
2. Make sure to place the **background\_music.wav** file in the same directory as your Java source file.



**13.** Create a Window application with a menu bar and menu. The frame will also have a text area with a scroll bar. In the menu, have File related options. Open a file and its content has to be displayed in the text area.

**Code:**

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;

public class FileViewerApp extends JFrame {
    private JTextArea textArea;

    public FileViewerApp() {
        setTitle("File Viewer");
        setSize(800, 600);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Create menu bar
        JMenuBar menuBar = new JMenuBar();
        setJMenuBar(menuBar);

        // Create File menu
        JMenu fileMenu = new JMenu("File");
        menuBar.add(fileMenu);

        // Create Open option
        JMenuItem openMenuItem = new JMenuItem("Open");
        fileMenu.add(openMenuItem);

        // Create text area with scroll bar
        textArea = new JTextArea();
        JScrollPane scrollPane = new JScrollPane(textArea);
        add(scrollPane, BorderLayout.CENTER);

        // Attach action listener to the Open option
```

```

openMenuItem.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        openFile();
    }
});
}

private void openFile() {
    JFileChooser fileChooser = new JFileChooser();

    // Set initial directory
    fileChooser.setCurrentDirectory(new File("."));

    // Show file chooser dialog
    int result = fileChooser.showOpenDialog(this);

    // Check if file was selected
    if (result == JFileChooser.APPROVE_OPTION) {
        File selectedFile = fileChooser.getSelectedFile();

        // Read file content and display in text area
        try (BufferedReader reader = new BufferedReader(new
FileReader(selectedFile))) {
            StringBuilder content = new StringBuilder();
            String line;
            while ((line = reader.readLine()) != null) {
                content.append(line).append("\n");
            }
            textArea.setText(content.toString());
        } catch (IOException e) {
            JOptionPane.showMessageDialog(this, "Error reading file: " +
e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
}

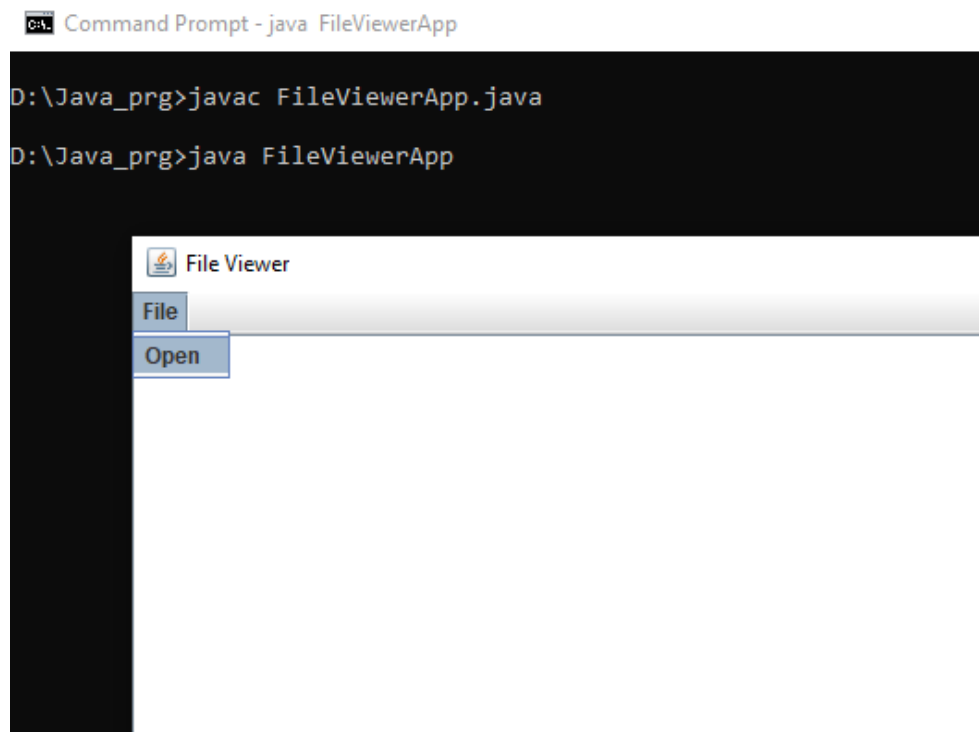
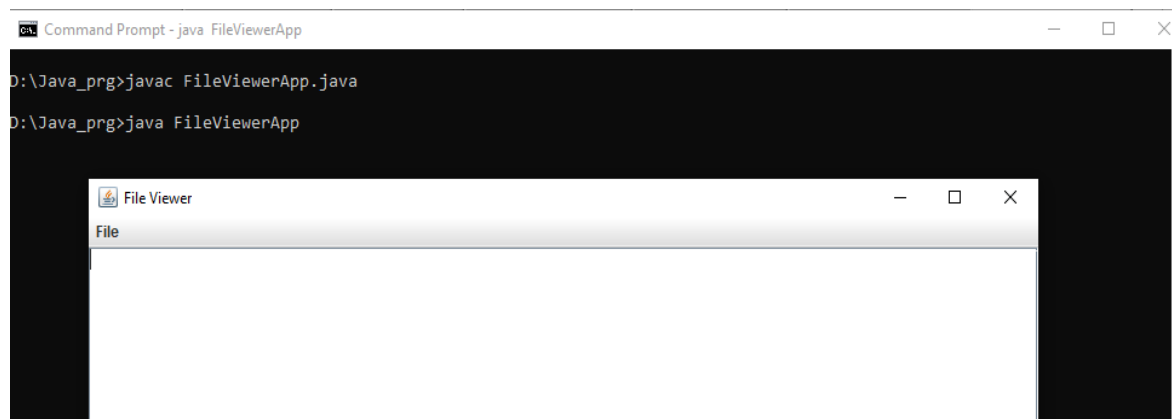
public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {

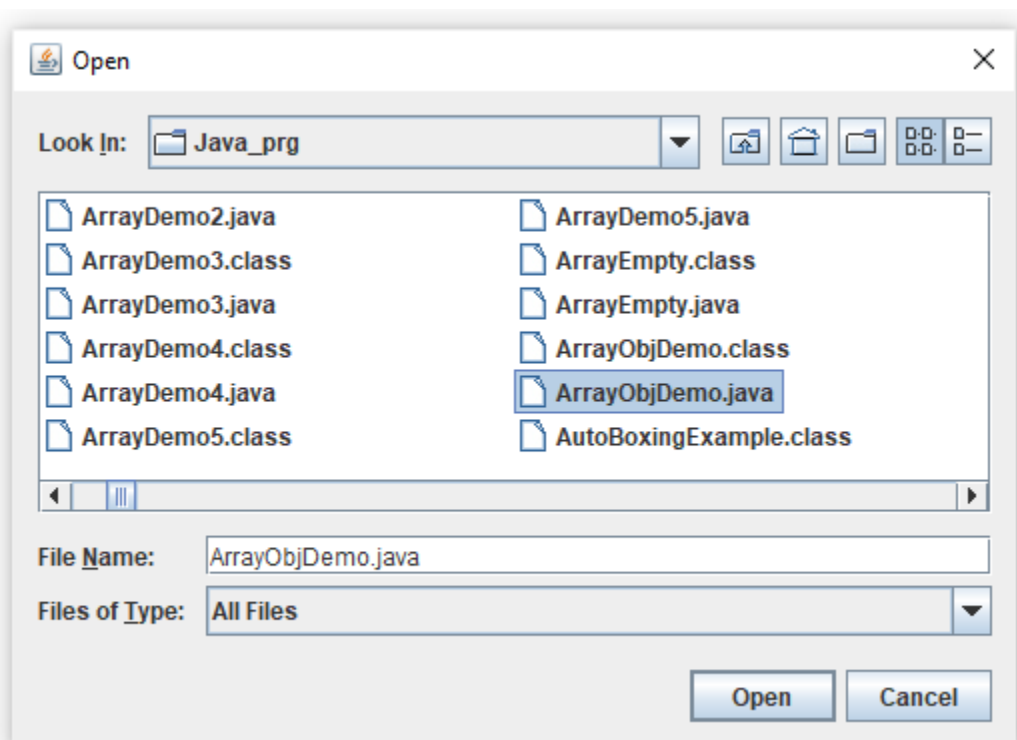
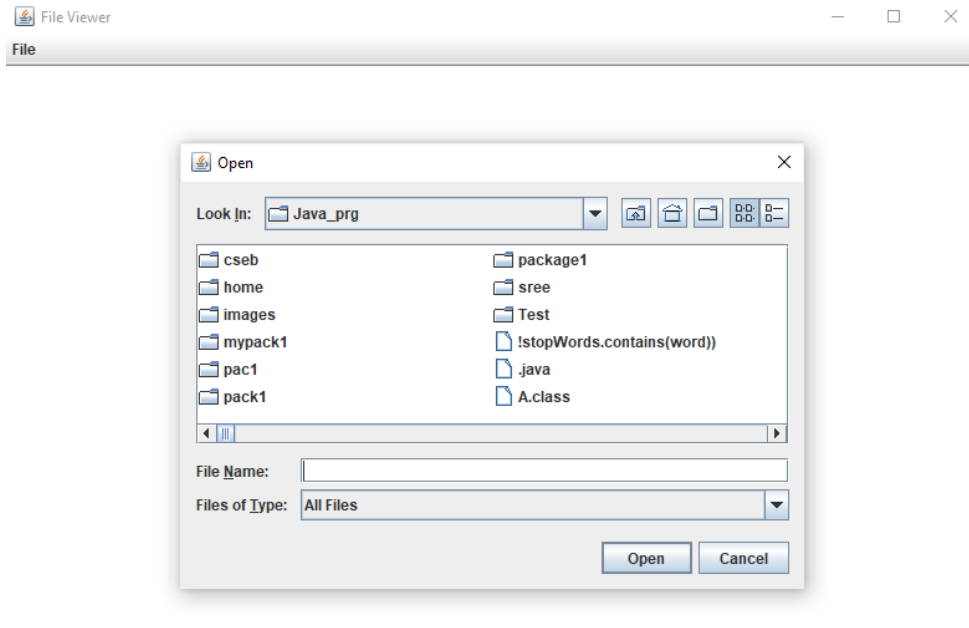
```

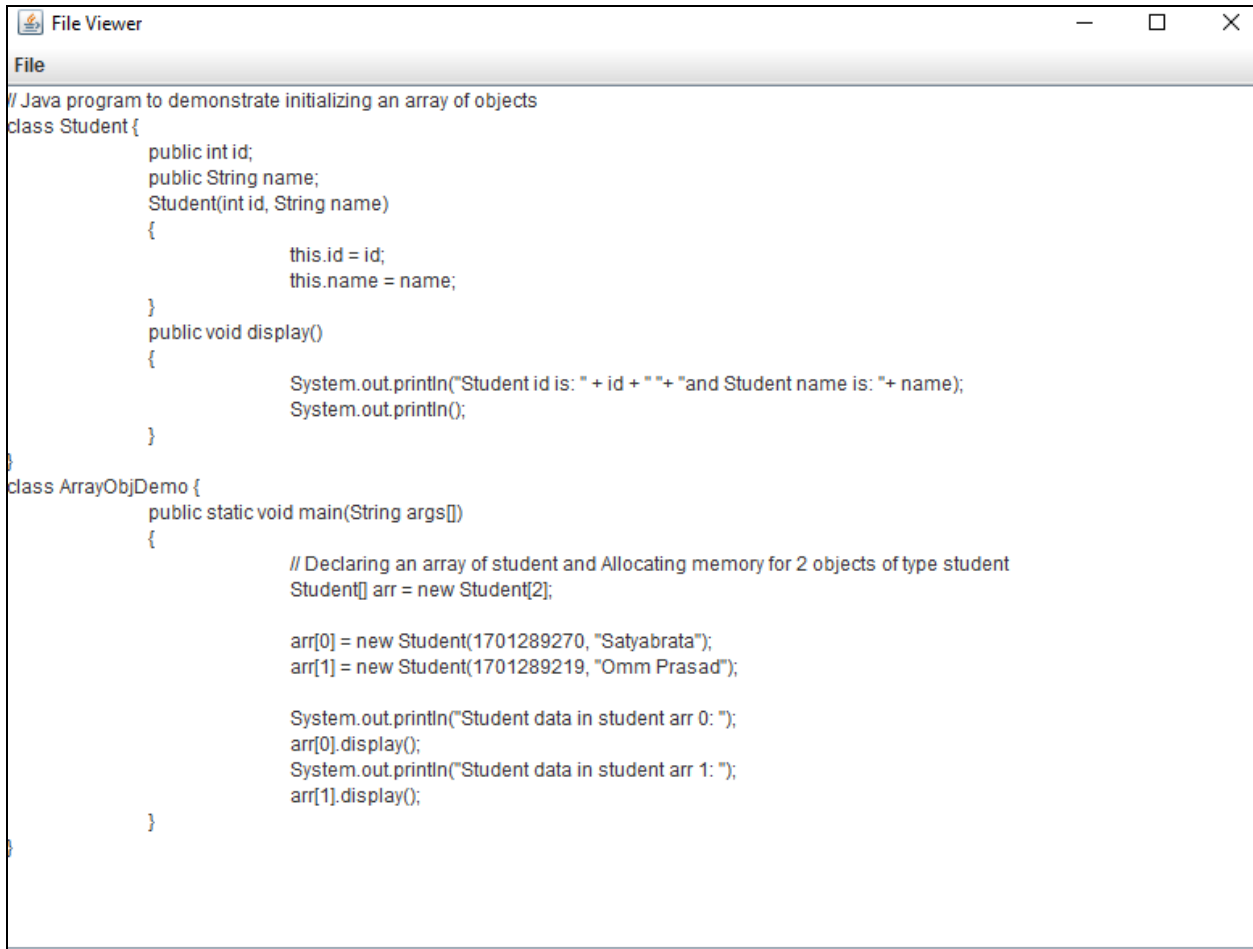


```
FileViewerApp fileViewerApp = new FileViewerApp();
fileViewerApp.setVisible(true);
    }
    });
}
```

## Output:







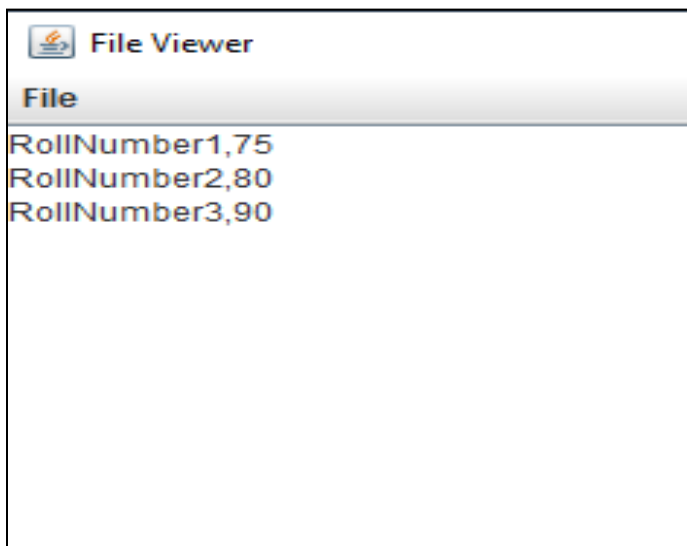
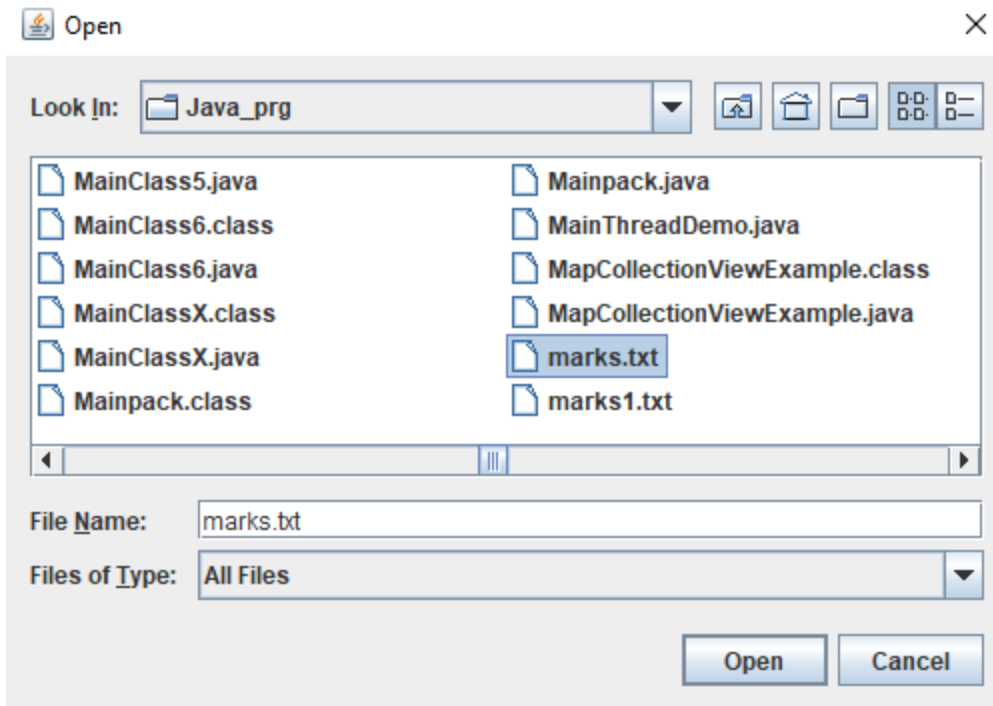
The image shows a window titled "File Viewer" with standard Windows window controls (minimize, maximize, close). The window contains a Java program. The code defines a `Student` class with attributes `id` and `name`, a constructor, and a `display` method. It also defines an `ArrayObjDemo` class with a `main` method that creates an array of `Student` objects and prints their details.

```
// Java program to demonstrate initializing an array of objects
class Student {
    public int id;
    public String name;
    Student(int id, String name)
    {
        this.id = id;
        this.name = name;
    }
    public void display()
    {
        System.out.println("Student id is: " + id + " " + "and Student name is: " + name);
        System.out.println();
    }
}

class ArrayObjDemo {
    public static void main(String args[])
    {
        // Declaring an array of student and Allocating memory for 2 objects of type student
        Student[] arr = new Student[2];

        arr[0] = new Student(1701289270, "Satyabrata");
        arr[1] = new Student(1701289219, "Omm Prasad");

        System.out.println("Student data in student arr 0: ");
        arr[0].display();
        System.out.println("Student data in student arr 1: ");
        arr[1].display();
    }
}
```



**14. Create a GUI using various controls: (i) to upload the marks of all the students presented in a marks.csv or marks.txt file into the database. (ii) to show the marks of the respective student after uploading the marks into the database. Note: Handle the exception, if the file is not present (or) if the marks are not uploaded in the database.**

Code:

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

public class MarksUploadApp extends JFrame {
    private JTextField fileTextField;
    private JTextArea resultTextArea;

    private Connection connection;

    public MarksUploadApp() {
        setTitle("Marks Upload App");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        // Create panel for file selection
        JPanel filePanel = new JPanel();
        filePanel.setLayout(new FlowLayout());

        JLabel fileLabel = new JLabel("File:");
        fileTextField = new JTextField(20);
        JButton browseButton = new JButton("Browse");

        filePanel.add(fileLabel);
        filePanel.add(fileTextField);
        filePanel.add(browseButton);

        // Create panel for buttons
        JPanel buttonPanel = new JPanel();
```

```
buttonPanel.setLayout(new FlowLayout());

JButton uploadButton = new JButton("Upload");
JButton viewButton = new JButton("View Marks");

buttonPanel.add(uploadButton);
buttonPanel.add(viewButton);

// Create text area for displaying result
resultTextArea = new JTextArea();
resultTextArea.setEditable(false);
JScrollPane scrollPane = new JScrollPane(resultTextArea);

// Add components to the frame
add(filePanel, BorderLayout.NORTH);
add(buttonPanel, BorderLayout.SOUTH);
add(scrollPane, BorderLayout.CENTER);

// Attach action listener to the Browse button
browseButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        browseFile();
    }
});

// Attach action listener to the Upload button
uploadButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        uploadMarks();
    }
});

// Attach action listener to the View Marks button
viewButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        viewMarks();
    }
});
```

```
});  
}
```

```
private void browseFile() {  
    JFileChooser fileChooser = new JFileChooser();  
    fileChooser.setCurrentDirectory(new File("."));  
  
    int result = fileChooser.showOpenDialog(this);  
    if (result == JFileChooser.APPROVE_OPTION) {  
        File selectedFile = fileChooser.getSelectedFile();  
        fileTextField.setText(selectedFile.getAbsolutePath());  
    }  
}
```

```
private void uploadMarks() {  
    String filePath = fileTextField.getText();  
  
    try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {  
        // Establish database connection  
        establishConnection();  
  
        // Create table if it doesn't exist  
        createMarksTable();  
  
        String line;  
        int count = 0;  
  
        while ((line = reader.readLine()) != null) {  
            String[] data = line.split(",");  
  
            if (data.length == 2) {  
                String rollNumber = data[0].trim();  
                int marks = Integer.parseInt(data[1].trim());  
  
                // Insert marks into the database  
                insertMarks(rollNumber, marks);  
  
                count++;  
            }  
        }  
    }  
}
```



```

        resultTextArea.setText("Marks uploaded for " + count + " students.");
    } catch (IOException e) {
        resultTextArea.setText("Error reading file: " + e.getMessage());
    } catch (SQLException e) {
        resultTextArea.setText("Error connecting to database: " + e.getMessage());
    }
}

```

```

private void viewMarks() {
    String rollNumber = JOptionPane.showInputDialog(this, "Enter Roll Number:");

```

```

    try {
        // Establish database connection
        establishConnection();

        // Retrieve marks from the database
        int marks = retrieveMarks(rollNumber);

        resultTextArea.setText("Marks for Roll Number " + rollNumber + ": " + marks);
    } catch (SQLException e) {
        resultTextArea.setText("Error connecting to database: " + e.getMessage());
    }
}

```

```

private void establishConnection() throws SQLException {
    // Replace "jdbc:mysql://localhost:3306/database_name" with your database
    //connection URL
    String url = "jdbc:mysql://localhost:3306/students_database";
    String username = "root";
    String password = "srm@123";

    connection = DriverManager.getConnection(url, username, password);
}

```

```

private void createMarksTable() throws SQLException {
    String createTableQuery = "CREATE TABLE IF NOT EXISTS stu_marks (" +
        "rollNumber VARCHAR(20) NOT NULL," +
        "marks INT NOT NULL," +
        "PRIMARY KEY (rollNumber)" +

```

```

        ");

        Statement statement = connection.createStatement();
        statement.executeUpdate(createTableQuery);
    }

    private void insertMarks(String rollNumber, int marks) throws SQLException {
        String insertQuery = "INSERT INTO stu_marks (rollNumber, marks) VALUES (?,
?)"

        PreparedStatement preparedStatement =
connection.prepareStatement(insertQuery);
        preparedStatement.setString(1, rollNumber);
        preparedStatement.setInt(2, marks);

        preparedStatement.executeUpdate();
    }

    private int retrieveMarks(String rollNumber) throws SQLException {
        String selectQuery = "SELECT marks FROM stu_marks WHERE rollNumber = ?";

        PreparedStatement preparedStatement =
connection.prepareStatement(selectQuery);
        preparedStatement.setString(1, rollNumber);

        ResultSet resultSet = preparedStatement.executeQuery();

        if (resultSet.next()) {
            return resultSet.getInt("marks");
        }

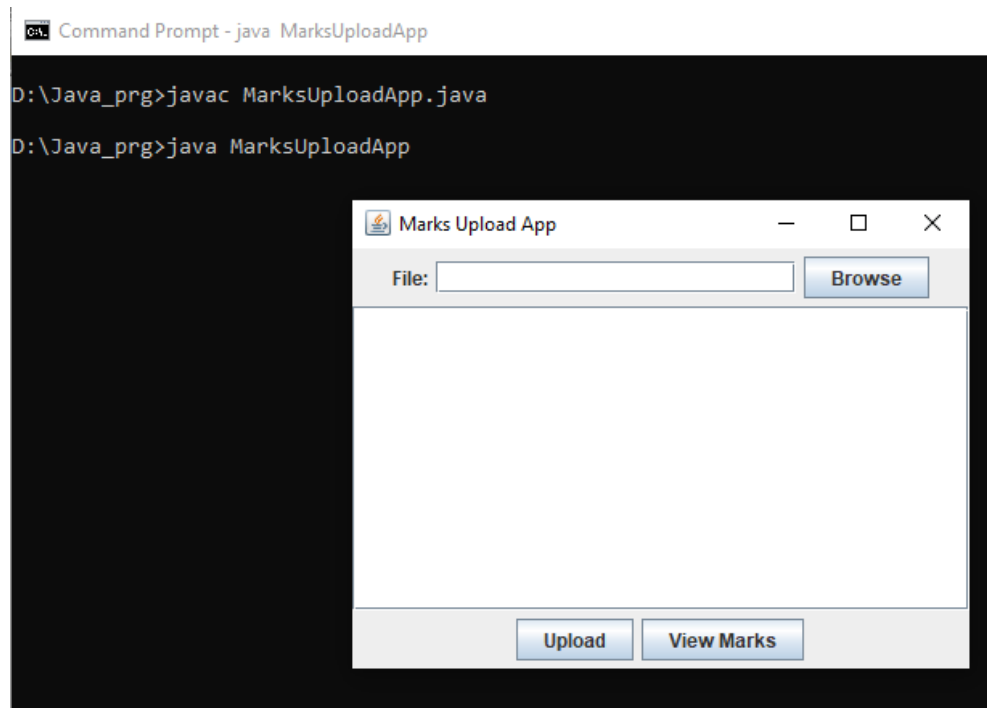
        return 0;
    }

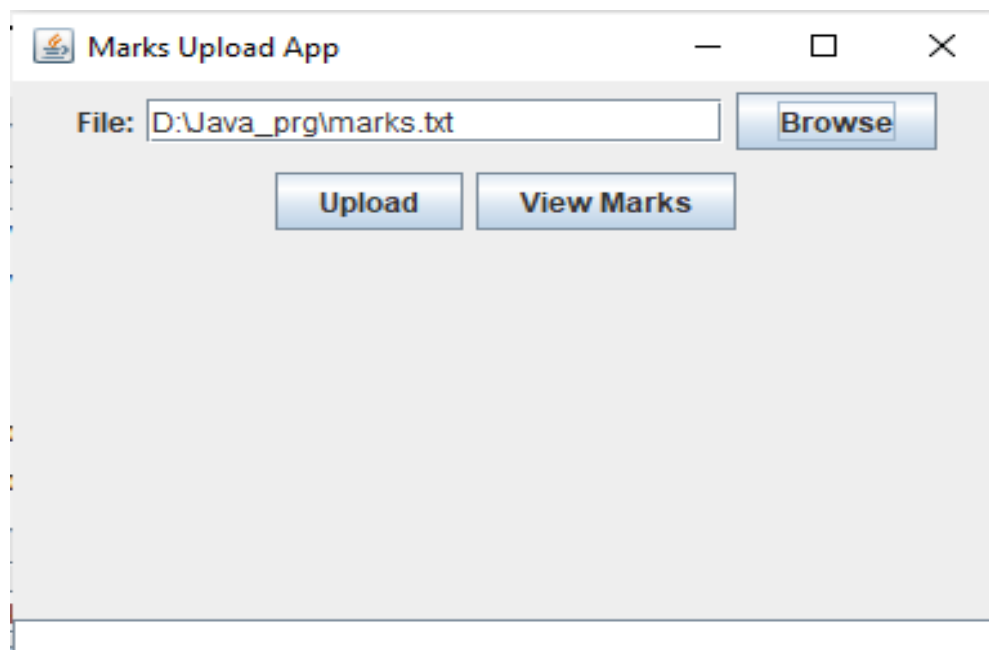
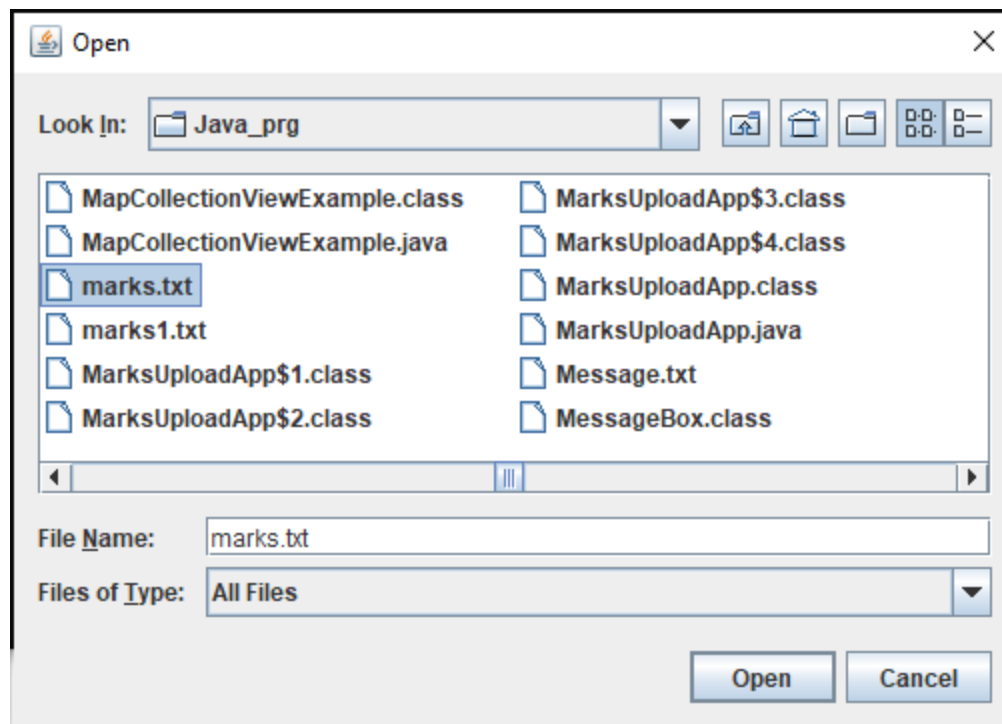
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                MarksUploadApp marksUploadApp = new MarksUploadApp();
                marksUploadApp.setVisible(true);
            }
        });
    }

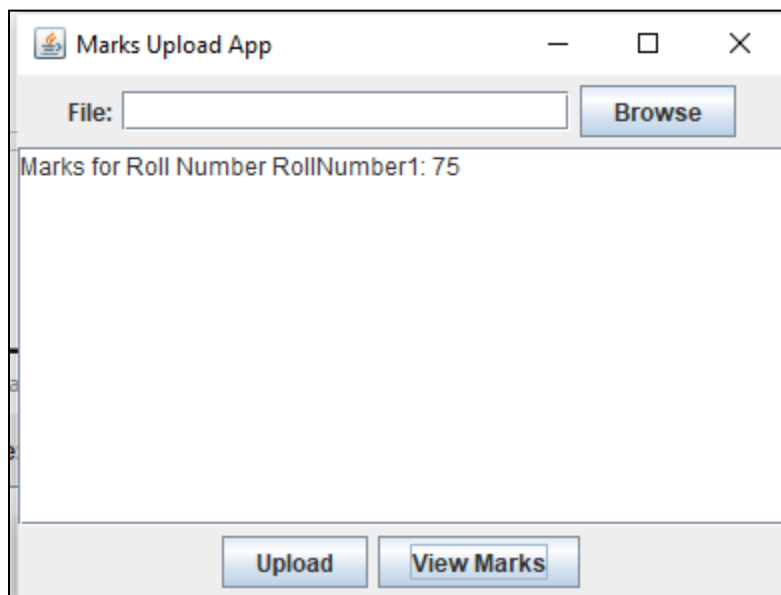
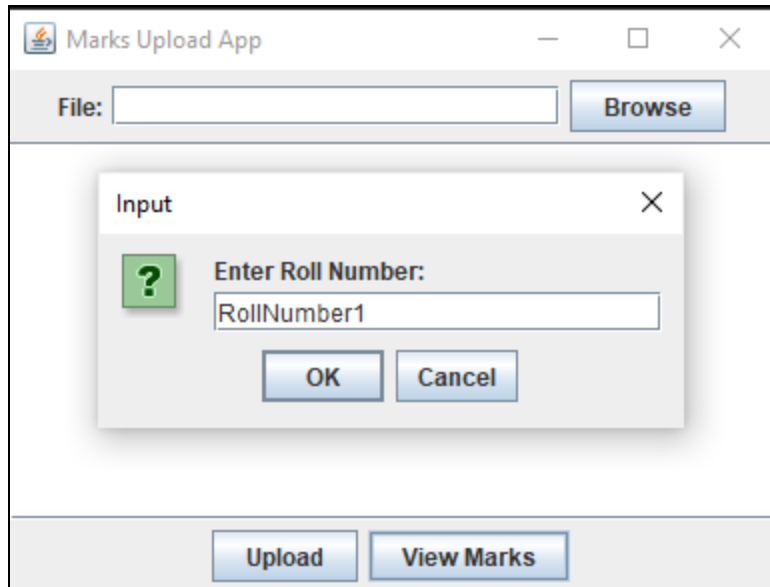
```

```
    });  
  }  
}
```

Output:







In Database: stu\_marks table values

```
mysql> select * from stu_marks;
+-----+-----+
| rollNumber | marks |
+-----+-----+
| RollNumber1 | 75 |
| RollNumber2 | 80 |
| RollNumber3 | 90 |
+-----+-----+
3 rows in set (0.00 sec)
```

-----END-----

