NAME- Harshita Totala ROLL NO- SEITA14 DIV- IT A
SUBJECT- Object Oriented Programming

ASSIGNMENT NO. 8

Aim: File Handling

Problem Statement

Implement a program for maintaining a student records database using File Handling. Student has Student_id, name, Roll_no, Class, marks and address. Display the data for five students.

i) Create Database

ii)Display Database

iii) Clear Records

iv)Modify record

v)Search Record

Objectives: To understand the concept of Java FileWriter and FileReader classes.

Theory:

Importance of file handling?

A Stream represents flow of data from one place to another place Input Streams reads or accepts data Output Streams sends or writes data to some other place All streams are represented as classes in javaio package The main advantage of using stream concept is to achieve hardware independence This is because we need not change the stream in our program even though we change the hardware Streams are of two types in Java:

Byte Streams: Handle data in the form of bits and bytes Byte streams are used to handle any

characters (text), images, audio and video files For example, to store an image file (gif or jpg), we should go for a byte stream To handle data in the form of 'bytes' the abstract classes: InputStream and OutputStream are used The important classes of byte streams are:

FileWriter is a class which is in java.io package that is use to create a file by directly writing

characters. Java FileWriter and FileReader classes are used to write and read data from text files (they are Character Stream classes). Reading and writing take place character by character, which increases the number of I/O operations and effects performance of the system. BufferedWriter can be used along with FileWriter to improve speed of execution.

FileWriter

FileWriter is useful to create a file writing characters into it.

• This class inherits from the OutputStream class.

• The constructors of this class assume that the default character encoding and the default

byte-buffer size are acceptable. To specify these values yourself, construct an OutputStreamWriter on a FileOutputStream.

• FileWriter is meant for writing streams of characters. For writing streams of raw bytes,

consider using a FileOutputStream.

• FileWriter creates the output file, if it is not present already.

Constructors of FileWriter

1. FileWriter(String filepath)

2. FileWriter(String filepath, boolean append)

3. FileWriter(File fileobj)

Methods of FileWriter

Method Name	Description
public void write(String text)	Use to write String into file.
public void write(char c)	se to write char into file.
public void close()	Use to to close the file object.
public void flush()	Use to flush the FileWriter contents.

If you will not close the file object then your file data may be lost so don’t forget to close file

object using close() method.

Buffer in java

File Manipulation and operations

Tokenizing the Input Using the Scanner Class

Example

```
import java.io.*;
class FileWriterTest{
  public static void main(String[] args)throws IOException {
           FileWriterfw=new FileWriter("myfile.txt");
           BufferedReaderbr=new BufferedReader(new InputStreamReader(System.in));
           char ch;
           System.out.println("Enter Char to Exit '@'");
           while( (ch=(char) br.read()) !='@' ){
           fw.write(ch);
 }
  fw.close();
}
}
```

In the above step a new file will be created every time and previous data will be lost.

FileWriterfw = new FileWriter("myfile.txt",true);

In this case file will not be create every time, If file already exist in given location then it will

append contents to existing file because mode true is added at the time creating FileWriter object.

FileReader

FileReader class is use to read text from the file.

Constructors of FileReader

1. FileReader(String filepath)

2. FileReader(File fileobj)

Methods of FileReader

1. int read() : Use to return integer value of next character.

2. int read(char buff[]) : Use to up to buffer length.

3. abstract void close() : Use to close the input source

Example

```
import java.io.*;
class FileWriterTest{
	public static void main(String[] args)throws IOException {
		FileReaderfr = new FileReader("myfile.txt");
		intch;
		System.out.println("File contents are:");
		while((ch=fr.read())!=-1){
			System.out.print((char)ch);
		}
		fr.close();
	}
}
```

Buffer in java

A buffer is a memory block that is used to store data. Buffer improved the speed of execution while reading and writing data. We can improve the speed by execution using the following Buffered class.

Buffer Classes: There are four types of buffer classes which work with Stream classes.

BufferedReader

BufferedWriter

BufferedInputStream

BufferedOutputStream

Implementation

Algorithm for Adding Records:-

1. Start
2. Open the database file.
3. Read data from the user.
4. Print the data to file.
5. Close the file.

6. End

Algorithm for Displaying Records:-

1. Start
2. Open the database file.
3. Read data from the filr.
4. Print the data on screen.
5. Close the file.
6. End

Algorithm for Clearing All Records:-

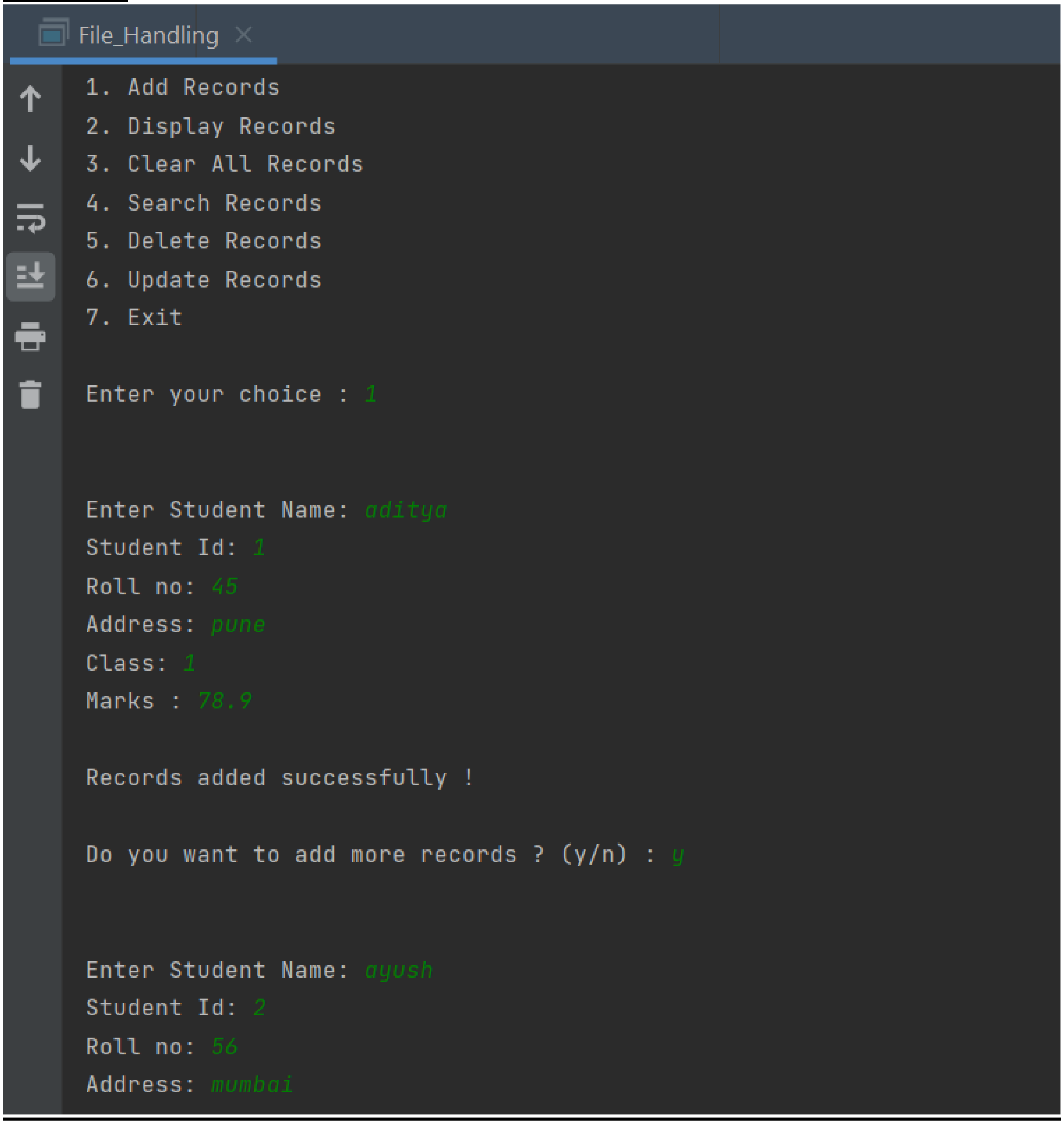
1. Start
2. Overwrite the database file with a blank file.
3. Close the file.
4. End

Program:

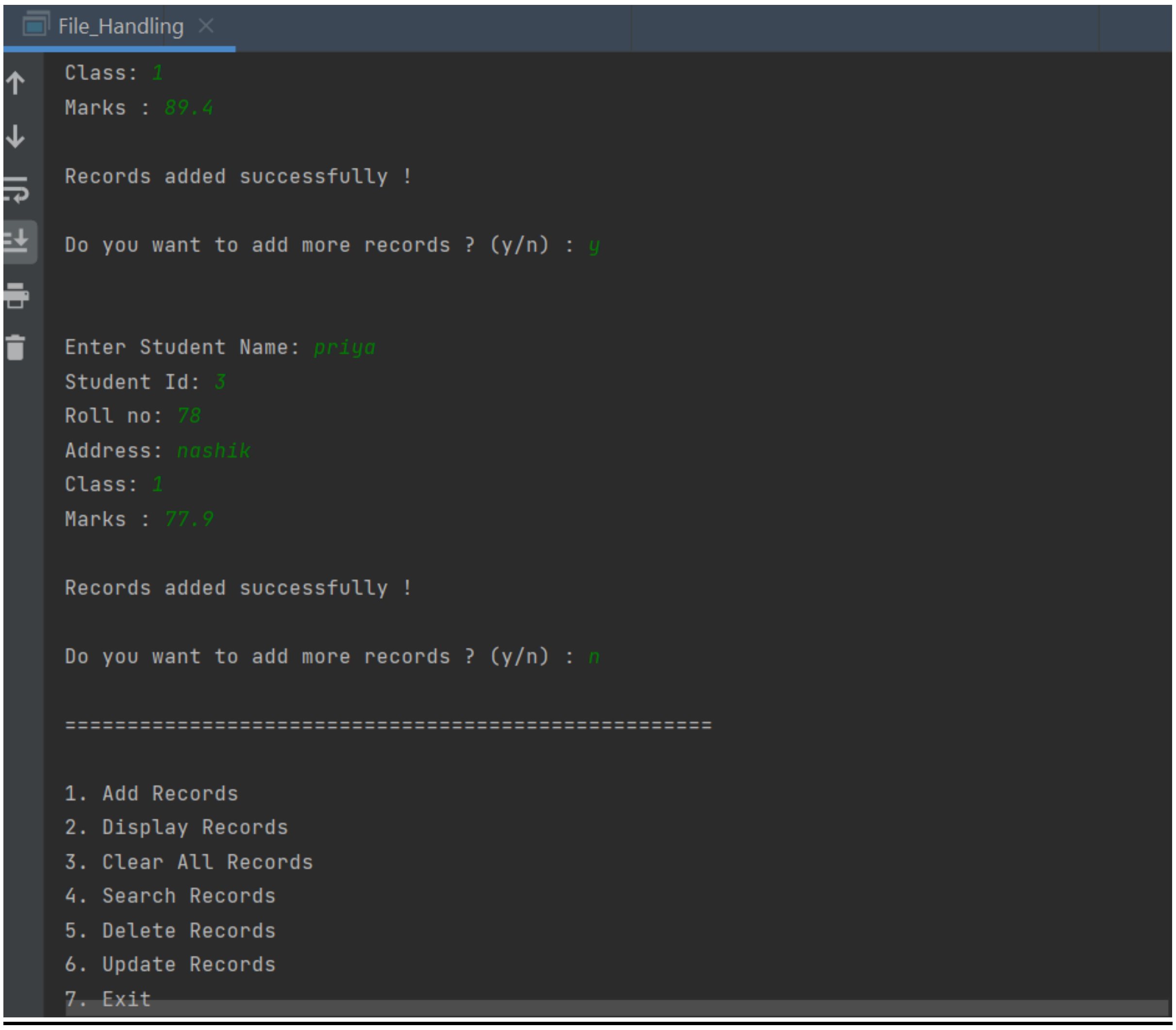
```java
package com.company.assignment;
import java.io.*;
import java.util.*;


class Database {
    static BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));

    public void addRecords() throws IOException {

        PrintWriter pw = new PrintWriter(new BufferedWriter(new
FileWriter("sample.txt",true)));

        String studentname, address,s;
        int studentid, rollno, Class;
        float marks;

        boolean addMore = false;
        do {
            System.out.print("\nEnter Student Name: ");
            studentname = br.readLine();
            System.out.print("Student Id: ");
            studentid = Integer.parseInt(br.readLine());
            System.out.print("Roll no: ");
            rollno = Integer.parseInt(br.readLine());
            System.out.print("Address: ");
            address = br.readLine();
            System.out.print("Class: ");
            Class = Integer.parseInt(br.readLine());
            System.out.print("Marks : ");
            marks = Float.parseFloat(br.readLine());
            pw.println(studentname+" "+studentid+" "+rollno+" "+address+"
"+Class+" "+marks);

            System.out.print("\nRecords added successfully !\n\nDo you want
to add more records ? (y/n) : ");
            s = br.readLine();
            if(s.equalsIgnoreCase("y")){
                addMore = true;
                System.out.println();
            }
            else
                addMore = false;
        }
        while(addMore);
        pw.close();
    }


    public void readRecords() throws IOException {
        try {

            BufferedReader file = new BufferedReader(new
FileReader("sample.txt"));
            String name;
            int i=1;


            while((name = file.readLine()) != null) {
                System.out.println(name);
                System.out.println("");
            } file.close();
        }
        catch(FileNotFoundException e){ //Exception handling
            System.out.println("\nERROR : File not Found !!!");
        }
    }


    public void searchRecords() throws IOException {
        try {
            BufferedReader file = new BufferedReader(new
FileReader("sample.txt"));
            String name;
            int flag=0;
            Scanner sc=new Scanner(System.in);
            System.out.print("Enter an id of the student you want to
search: ");

            String searchname=sc.next(); //taking input from user

            while((name = file.readLine()) != null) {
                String[] line = name.split(" ");

                if(searchname.equalsIgnoreCase(line[1])){
                    System.out.println("Record found");
                    System.out.println(name);
                    System.out.println("");
                    flag=1;
                    break;
                }
            }
            if(flag==0)
                System.out.println("Record not found");
            file.close();
        }
        catch(FileNotFoundException e) {
            System.out.println("\nERROR : File not Found !!!");
        }
    }


    public void deleteRecords() throws IOException {
        try {
            BufferedReader file1 = new BufferedReader(new
FileReader("sample.txt"));
            PrintWriter pw = new PrintWriter(new BufferedWriter(new
FileWriter("new.txt",true)));
            String name;
            int flag=0;
            Scanner sc=new Scanner(System.in);
            System.out.print("Enter the name of the student you want to
delete: ");
            String searchname=sc.next();
            while((name = file1.readLine()) != null) {
                String[] line = name.split(" ");
                if(!searchname.equalsIgnoreCase(line[0])){
                    pw.println(name);
                    flag=0;
                }
                else{
                    System.out.println("Record found");
                    flag=1;
                }
            } file1.close();
            pw.close();

            File delName = new File("sample.txt");
            File oldName = new File("new.txt");
            File newName = new File("sample.txt");

            if(delName.delete())
                System.out.println("deleted successfully");
            else
                System.out.println("Error");

            if (oldName.renameTo(newName))
                System.out.println("Renamed successfully");
            else
                System.out.println("Error");

        }
        catch(FileNotFoundException e) {
            System.out.println("\nERROR : File not Found !!!");
        }
    }
```

```java
    public void updateRecords() throws IOException {
        try {

            BufferedReader file1 = new BufferedReader(new
FileReader("sample.txt"));
            PrintWriter pw = new PrintWriter(new BufferedWriter(new
FileWriter("new.txt",true)));
            String name;
            int flag=0;
            Scanner sc=new Scanner(System.in);
            System.out.print("Enter the name of the student you want to
update: ");
            String searchname=sc.next();

            while((name = file1.readLine()) != null) {
                String[] line = name.split(" ");

                if(!searchname.equalsIgnoreCase(line[0])){
                    pw.println(name);
                    flag=0;
                }
                else
                {
                    System.out.println("Record found");
                    System.out.print("Enter updated marks: ");
                    String up_mark=sc.next();
                    pw.println(line[0]+" "+line[1]+" "+line[2]+"
"+line[3]+" "+line[4]+" "+up_mark);
                    flag=1;
                }
            }
            file1.close();
            pw.close();
            File delName = new File("sample.txt");
            File oldName = new File("new.txt");
            File newName = new File("sample.txt");

            if(delName.delete())
                System.out.println("record updated successfully");
            else
                System.out.println("Error");

            if (oldName.renameTo(newName))
                System.out.println("Renamed successfully");
            else
                System.out.println("Error");

        }
        catch(FileNotFoundException e) {
            System.out.println("\nERROR : File not Found !!!");
        }
    }


    public void clear(String filename) throws IOException {

        PrintWriter pw = new PrintWriter(new BufferedWriter(new
FileWriter(filename)));
        pw.close();
        System.out.println("\nAll Records cleared successfully !");
    }



}




public class File_Handling{
    public static void main(String args[]) throws IOException {
        Database f = new Database();
        Scanner sc =new Scanner(System.in);
        System.out.println("");
        while(true) {

            System.out.print("1. Add Records\n2. Display Records\n3. Clear 
All Records\n4. Search Records"
                    + "\n5. Delete Records\n6. Update Records \n7. 
Exit\n\nEnter your choice : ");
            int choice = sc.nextInt();
            System.out.println("");


            switch(choice) {
                case 1:
                    f.addRecords();

System.out.println("\n====================================================\
n");
                    break;

                case 2:
                    f.readRecords();

System.out.println("\n====================================================\
n");
                    break;

                case 3:
                    f.clear("sample.txt");

System.out.println("\n====================================================\
n");
                    break;

                case 4:
                    f.searchRecords();

System.out.println("\n====================================================\
n");
                    break;

                case 5:
                    f.deleteRecords();

System.out.println("\n====================================================\
n");
                    break;
                case 6:
                    f.updateRecords();

System.out.println("\n====================================================\
n");
                    break;

                case 7:

System.out.println("\n====================================================\
n");
                    System.exit(0);
                    break;

                default:
                    System.out.println("\nInvalid Choice !");

System.out.println("\n====================================================\
n");
                    break;
            }
        }

    }



}
```

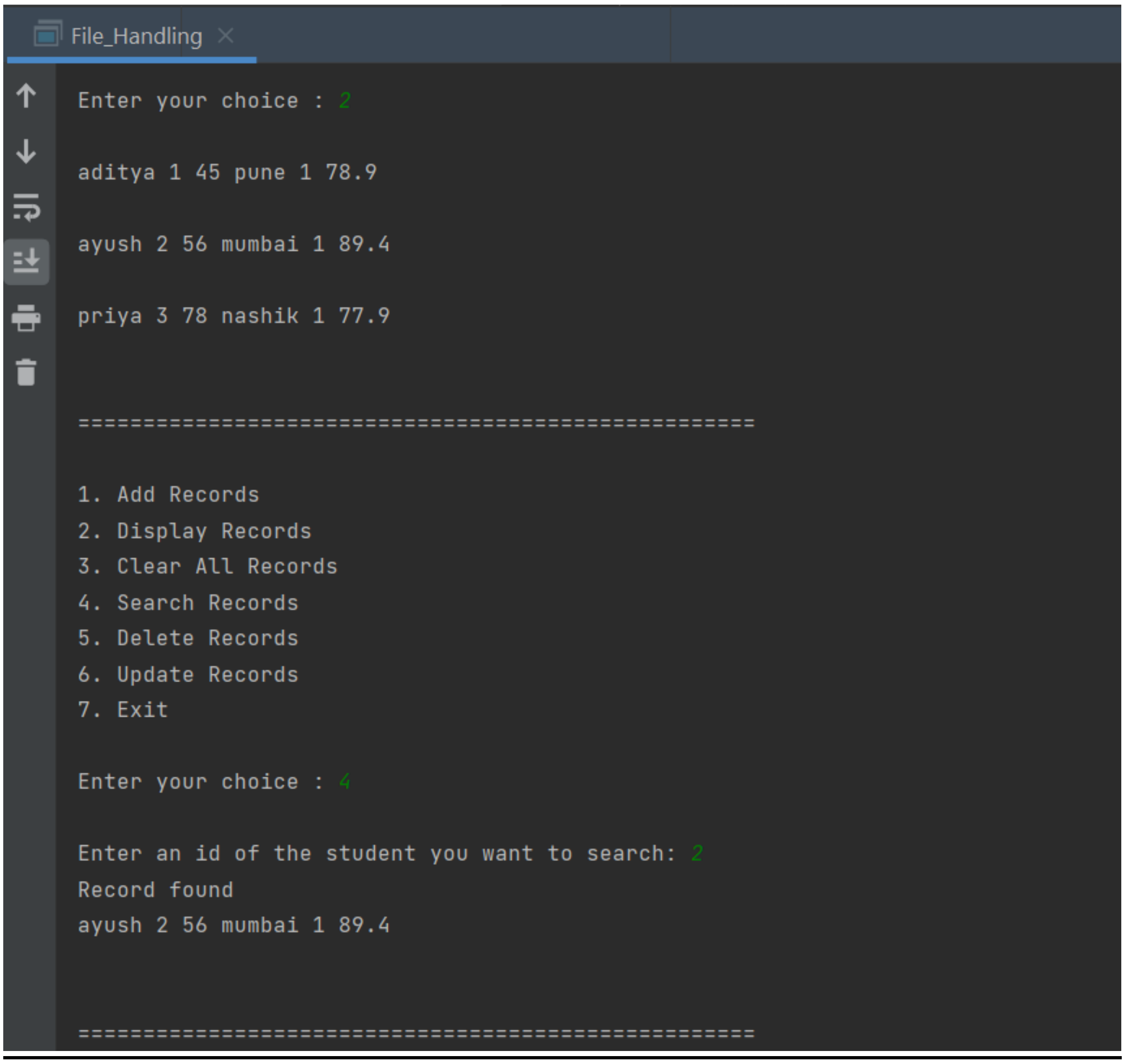
OUTPUT:



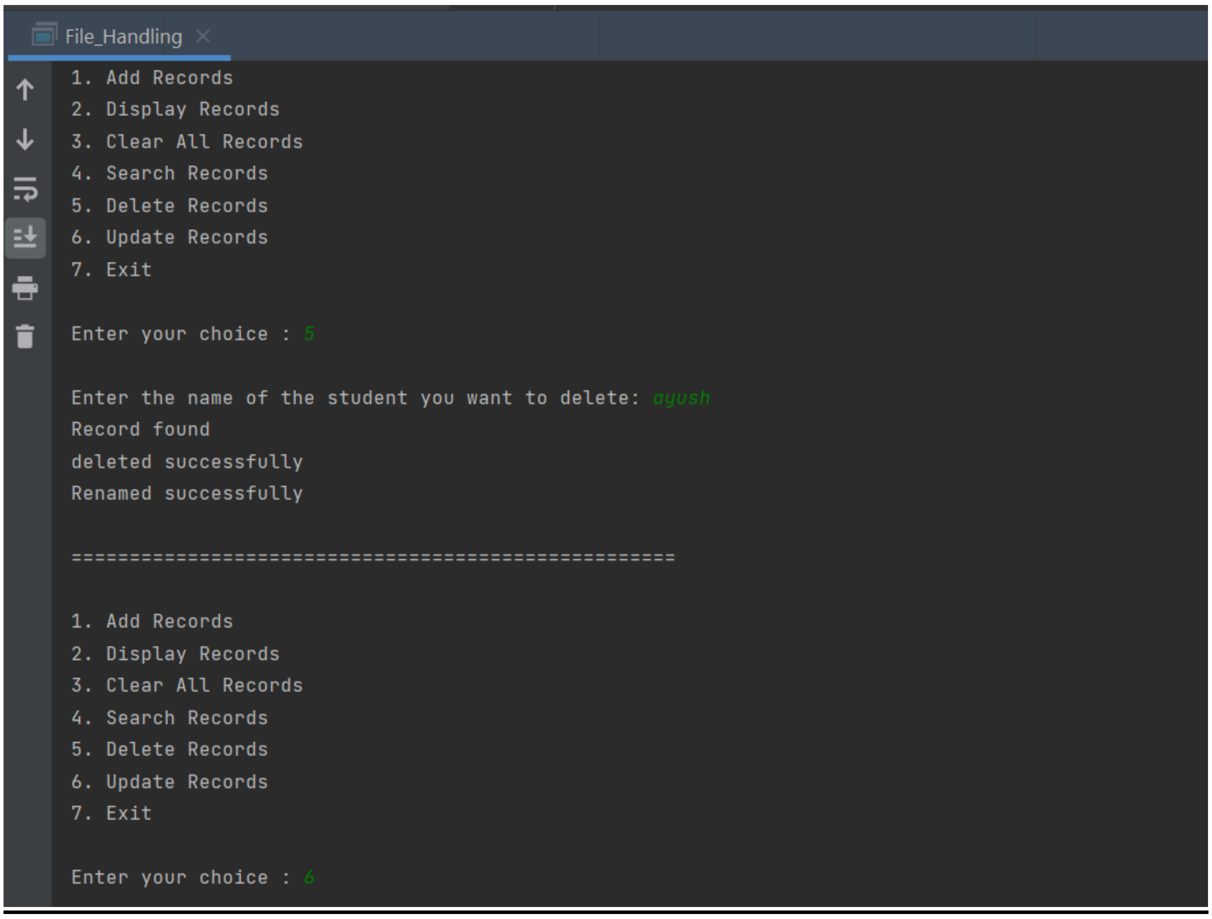
```
1. Add Records
2. Display Records
3. Clear All Records
4. Search Records
5. Delete Records
6. Update Records
7. Exit

Enter your choice : 1



Enter Student Name: aditya
Student Id: 1
Roll no: 45
Address: pune
Class: 1
Marks : 78.9

Records added successfully !

Do you want to add more records ? (y/n) : y



Enter Student Name: ayush
Student Id: 2
Roll no: 56
Address: mumbai
```



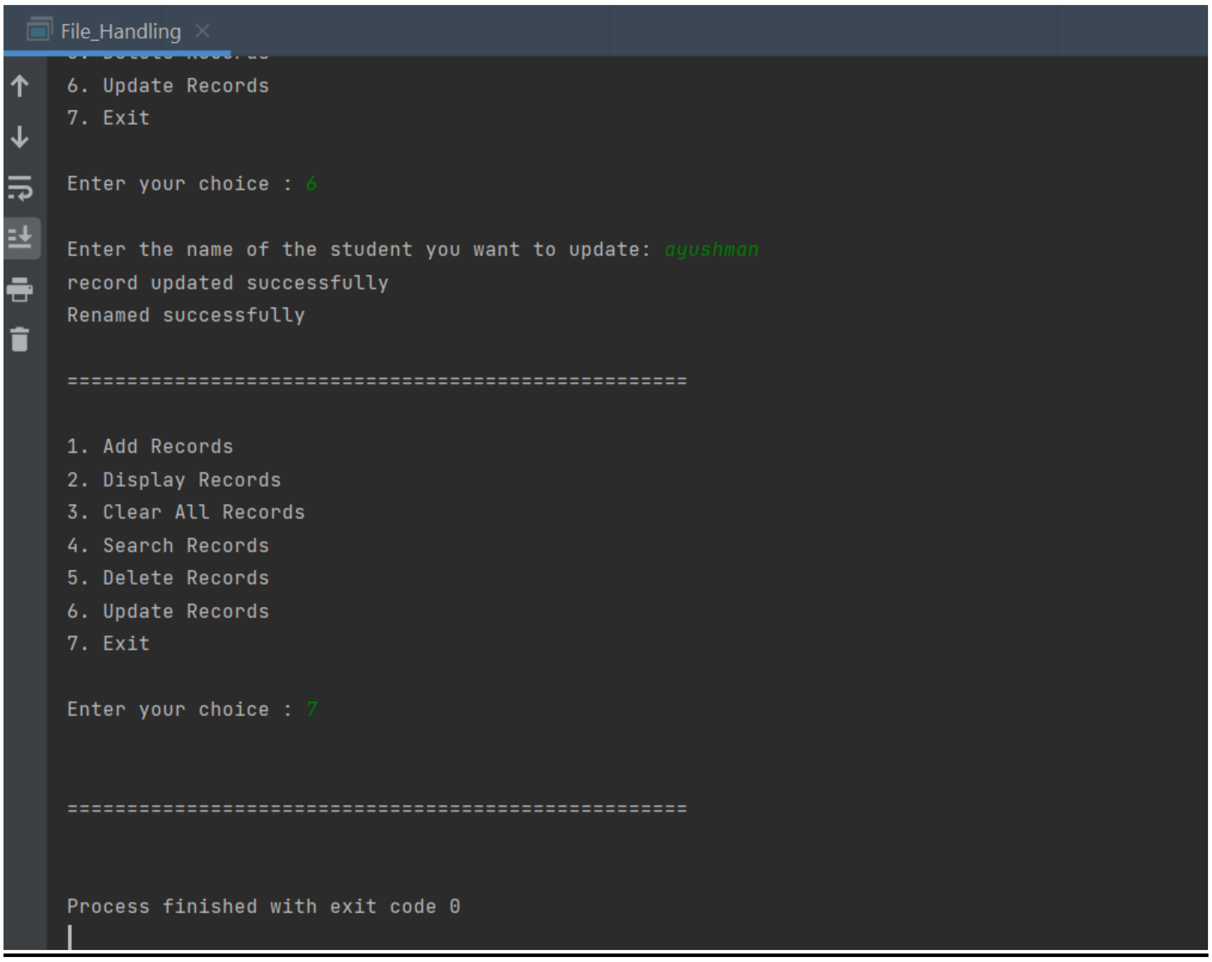
```
Class: 1
Marks : 89.4

Records added successfully !

Do you want to add more records ? (y/n) : y


Enter Student Name: priya
Student Id: 3
Roll no: 78
Address: nashik
Class: 1
Marks : 77.9

Records added successfully !

Do you want to add more records ? (y/n) : n

====================================================

1. Add Records
2. Display Records
3. Clear All Records
4. Search Records
5. Delete Records
6. Update Records
7. Exit
```



```
Enter your choice : 2

aditya 1 45 pune 1 78.9

ayush 2 56 mumbai 1 89.4

priya 3 78 nashik 1 77.9


=====================================================


1. Add Records
2. Display Records
3. Clear All Records
4. Search Records
5. Delete Records
6. Update Records
7. Exit

Enter your choice : 4

Enter an id of the student you want to search: 2
Record found
ayush 2 56 mumbai 1 89.4


=====================================================
```



```
1. Add Records
2. Display Records
3. Clear All Records
4. Search Records
5. Delete Records
6. Update Records
7. Exit

Enter your choice : 5

Enter the name of the student you want to delete: ayush
Record found
deleted successfully
Renamed successfully

====================================================

1. Add Records
2. Display Records
3. Clear All Records
4. Search Records
5. Delete Records
6. Update Records
7. Exit

Enter your choice : 6
```



File_Handling

```
6. Update Records
7. Exit

Enter your choice : 6

Enter the name of the student you want to update: ayushman
record updated successfully
Renamed successfully

====================================================

1. Add Records
2. Display Records
3. Clear All Records
4. Search Records
5. Delete Records
6. Update Records
7. Exit

Enter your choice : 7


====================================================


Process finished with exit code 0
```

Conclusion: Here, we have learned and implemented file handling concepts successfully.