# Assignment. 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

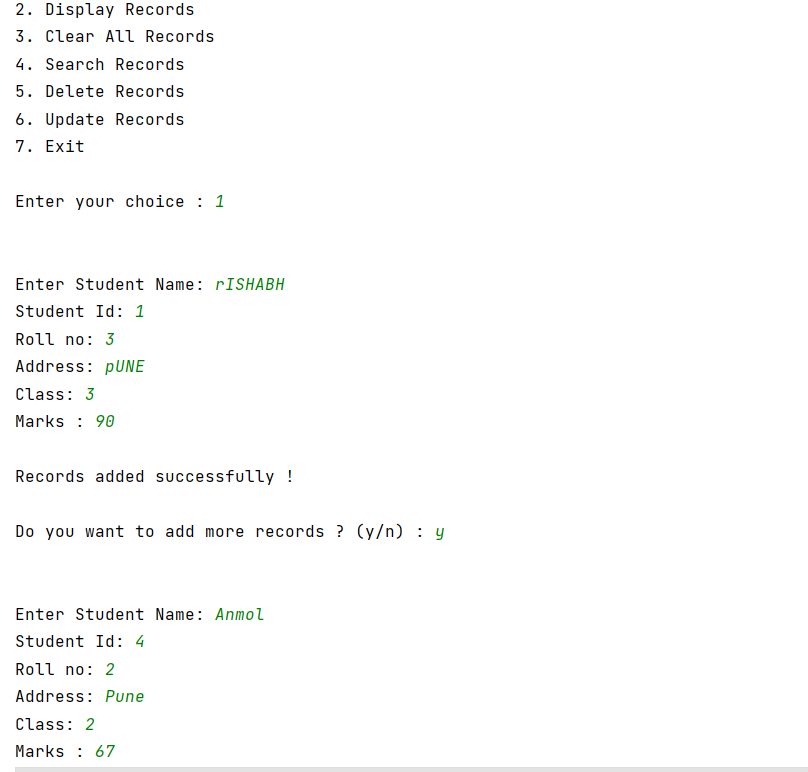
## Frequently asked Question

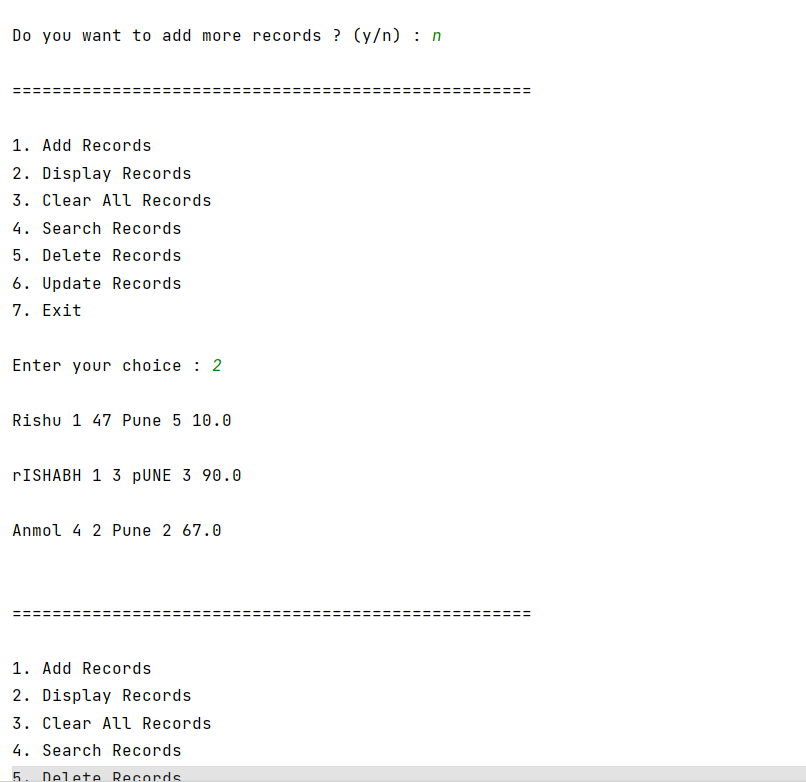
* 1. Give the basic methods in File class?
  2. Why to use FileWriter class and give its advantages .
  3. Write a File-Copy program which copies the content of one file to another. Take both the file names from the user
  4. How to read data from a file ,using FileReader class?
  5. Give usage of BufferedWriter and BufferedReader classes in Java with example
  6. Write a code to generate database for Criket player ising file handling operations

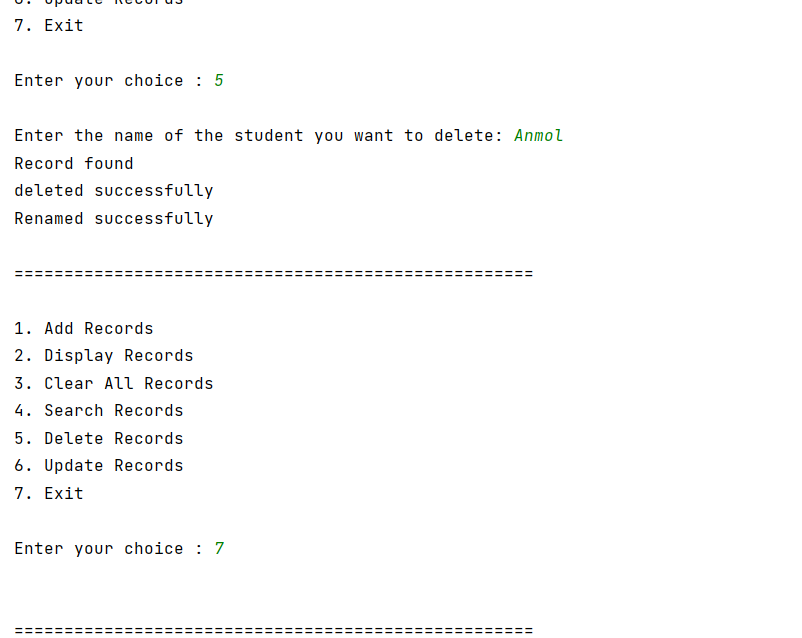
CODE:

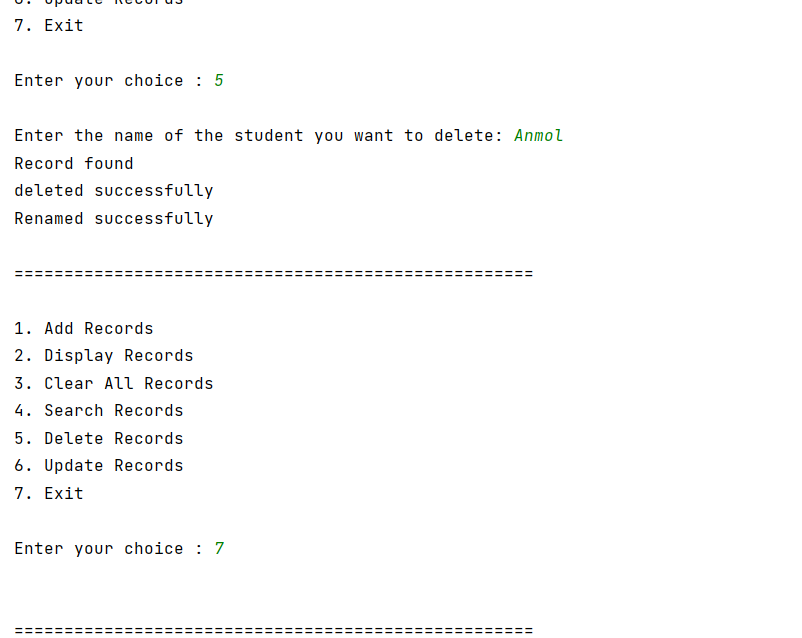
package com.company;  
  
*//Problem Statement::  
/\*  
Implement a program for maintaining a student records database using File Handling.  
Student has Student\_id, name, Roll\_no, Class, marks and address. Display the data  
for five students.  
\*/  
  
  
  
//package assignment;*import java.io.\*;  
import java.util.\*;  
  
  
  
  
*// ========================= FILE1 CLASS =========================//*class Database {  
 static BufferedReader *br* = new BufferedReader(new InputStreamReader(System.*in*));  
*//creating bufferredReder class object  
  
 // ---------------------- addRecords method ---------------------- //* public void addRecords() throws IOException {  
*// Create or Modify a file for Database* PrintWriter pw = new PrintWriter(new BufferedWriter(new FileWriter("sample.txt",true)));  
*//creating file with name sapmle.txt* String studentname, address,s;*//declaration of studentname , address ,s* int studentid, rollno, Class;*//declaration of studentid , rollno, Class* float marks;*//declaration of marks* boolean addMore = false; *//declaration of addmore* do {  
 System.*out*.print("\nEnter Student Name: "); *//printing on console* studentname = *br*.readLine(); *//taking input from user* System.*out*.print("Student Id: "); *//printing on console* studentid = Integer.*parseInt*(*br*.readLine()); *//taking input from user* System.*out*.print("Roll no: ");*//printing on console* rollno = Integer.*parseInt*(*br*.readLine()); *//taking input from user* System.*out*.print("Address: "); *//printing on console* address = *br*.readLine(); *//taking input from user* System.*out*.print("Class: ");*//printing on console* Class = Integer.*parseInt*(*br*.readLine()); *//taking input from user* System.*out*.print("Marks : "); *//printing on console* marks = Float.*parseFloat*(*br*.readLine()); *//taking input from user* pw.println(studentname+" "+studentid+" "+rollno+" "+address+" "+Class+" "+marks);  
*//appending data into to file* System.*out*.print("\nRecords added successfully !\n\nDo you want to add more records ? (y/n) : ");  
 s = *br*.readLine();*//take input from user* if(s.equalsIgnoreCase("y")){  
 addMore = true;*//modify addmore* System.*out*.println();  
 }  
 else  
 addMore = false; *//modify addmore* }  
 while(addMore);  
 pw.close();  
 }  
  
 *// ---------------------- addRecords method ---------------------- //* public void readRecords() throws IOException {  
 try {  
*// Open the file* BufferedReader file = new BufferedReader(new FileReader("sample.txt"));  
 String name; *//declaration of string name* int i=1; *//intizing value of i=1  
  
// Read records from the file* while((name = file.readLine()) != null) {  
 System.*out*.println(name); *//printing on console* System.*out*.println("");  
 } file.close();  
 }  
 catch(FileNotFoundException e){ *//Exception handling* System.*out*.println("\nERROR : File not Found !!!"); *//printing on console* }  
 }  
  
 *// ---------------------- addRecords method ---------------------- //* public void searchRecords() throws IOException {  
 try { *// Open the file* BufferedReader file = new BufferedReader(new FileReader("sample.txt"));  
 String name;*//declaration of string name* int flag=0; *//intizing value of flag=0* Scanner sc=new Scanner(System.*in*); *//creating obj of scanner class* System.*out*.print("Enter an id of the student you want to search: ");  
*//printing on console* String searchname=sc.next(); *//taking input from user  
// Read records from the file* while((name = file.readLine()) != null) {  
 String[] line = name.split(" ");  
  
 if(searchname.equalsIgnoreCase(line[1])){  
 System.*out*.println("Record found"); *//printing on console* System.*out*.println(name); *//printing record on console* System.*out*.println("");  
 flag=1; *//modify value* break;  
 }  
 }  
 if(flag==0) *//check condition* System.*out*.println("Record not found"); *//printing on console* file.close(); *//closing file* }  
 catch(FileNotFoundException e) {*//Exception handling* System.*out*.println("\nERROR : File not Found !!!");*//printing on console* }  
 }  
  
 *// ---------------------- addRecords method ---------------------- //* public void deleteRecords() throws IOException {  
 try { *// Open the file* BufferedReader file1 = new BufferedReader(new FileReader("sample.txt"));  
 PrintWriter pw = new PrintWriter(new BufferedWriter(new FileWriter("new.txt",true)));  
 String name; *//declaration of string name* int flag=0; *//intizing value of flag=0* Scanner sc=new Scanner(System.*in*); *//creating obj of scanner class* System.*out*.print("Enter the name of the student you want to delete: ");  
 String searchname=sc.next(); *// Read records from the file* while((name = file1.readLine()) != null) {  
 String[] line = name.split(" ");  
 if(!searchname.equalsIgnoreCase(line[0])){  
 pw.println(name);  
 flag=0; *//modify value* }  
 else{  
 System.*out*.println("Record found"); *//printing on console* flag=1;*//modify value* }  
 } file1.close();*//closing file* pw.close();  
  
 File delName = new File("sample.txt");*//creating obj of sample.txt* File oldName = new File("new.txt"); *//creating obj of new.txt* File newName = new File("sample.txt"); *//creating obj of sample.txt* if(delName.delete())  
 System.*out*.println("deleted successfully"); *//printing on console* else  
 System.*out*.println("Error");*//printing on console* if (oldName.renameTo(newName))  
 System.*out*.println("Renamed successfully"); *//printing on console* else  
 System.*out*.println("Error"); *//printing on console* }  
 catch(FileNotFoundException e) {*//Exception handling* System.*out*.println("\nERROR : File not Found !!!");  
 }  
 }  
  
 *// ---------------------- addRecords method ---------------------- //* public void updateRecords() throws IOException {  
 try {  
*// Open the file* BufferedReader file1 = new BufferedReader(new FileReader("sample.txt"));  
 PrintWriter pw = new PrintWriter(new BufferedWriter(new FileWriter("new.txt",true)));  
 String name;*//declaration of string name* int flag=0; *//intizing flag to 0* Scanner sc=new Scanner(System.*in*); *//creating obje of scanner class* System.*out*.print("Enter the name of the student you want to update: "); *//printing on console* String searchname=sc.next(); *// Read records from the file* while((name = file1.readLine()) != null) { *//check condition* String[] line = name.split(" ");  
  
 if(!searchname.equalsIgnoreCase(line[0])){ *//check condition* pw.println(name);  
 flag=0; *//modify value of flag* }  
 else  
 {  
 System.*out*.println("Record found"); *//printing on console* System.*out*.print("Enter updated marks: "); *//printing on console* String up\_mark=sc.next(); *//taking input from user* pw.println(line[0]+" "+line[1]+" "+line[2]+" "+line[3]+" "+line[4]+" "+up\_mark);  
 flag=1; *//modify value of flag* }  
 }  
 file1.close(); *//closing file* pw.close();  
 File delName = new File("sample.txt");*//creating obj of sample.txt* File oldName = new File("new.txt"); *//creating obj of new.txt* File newName = new File("sample.txt"); *//creating obj of sample.txt* if(delName.delete()) *//check condition* System.*out*.println("record updated successfully"); *//printing on console* else  
 System.*out*.println("Error"); *//printing on console* if (oldName.renameTo(newName)) *//check condition* System.*out*.println("Renamed successfully"); *//printing on console* else  
 System.*out*.println("Error"); *//printing on console* }  
 catch(FileNotFoundException e) { *//Exception handling* System.*out*.println("\nERROR : File not Found !!!"); *//printing on console* }  
 }  
  
 *// ---------------------- addRecords method ---------------------- //* public void clear(String filename) throws IOException {  
*// Create a blank file* PrintWriter pw = new PrintWriter(new BufferedWriter(new FileWriter(filename)));  
 pw.close(); *//closing PrintWriter object* System.*out*.println("\nAll Records cleared successfully !");  
*//printing on console* }  
  
  
  
}  
  
  
  
*// ========================= MAIN CLASS =========================//*public class AssignmentNo8{  
 public static void main(String args[]) throws IOException {  
 Database f = new Database(); *//creating obj of Database class* Scanner sc =new Scanner(System.*in*);*//creating object of scanner class* System.*out*.println("");  
 while(true) {  
*//menu driven* 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();*//taking input from user* System.*out*.println("");  
  
*//switch Case* switch(choice) {  
 case 1:  
 f.addRecords(); *//calling addRecords method* System.*out*.println("\n====================================================\n");  
 break;  
  
 case 2:  
 f.readRecords(); *//calling readRecords method* System.*out*.println("\n====================================================\n");  
 break;  
  
 case 3:  
 f.clear("sample.txt"); *//calling clear method* System.*out*.println("\n====================================================\n");  
 break;  
  
 case 4:  
 f.searchRecords(); *//calling searchRecords method* System.*out*.println("\n====================================================\n");  
 break;  
  
 case 5:  
 f.deleteRecords();*//calling deleteRecords method* System.*out*.println("\n====================================================\n");  
 break;  
  
 case 6:  
 f.updateRecords(); *//calling updateRecords method* System.*out*.println("\n====================================================\n");  
 break;  
  
 case 7:  
 System.*out*.println("\n====================================================\n");  
 System.*exit*(0);*//stop execution of program* break;  
  
 default:  
 System.*out*.println("\nInvalid Choice !"); *//default case* System.*out*.println("\n====================================================\n");  
 break;  
 }  
 }  
  
 }  
  
  
  
}

OUTPUT:









CONCLUSION

IN THIS ASSIGNMENT WE HAVE SUCCESSFULLY LEARNED THE CONCEPT OF FILE HANDLING IN JAVA.