Problem Statement: This program builds a file-based Student Record System using file handling, collections, and core OOP concepts. It loads and saves student records from a text file using BufferedReader and BufferedWriter, ensuring permanent storage. Student data is stored in an ArrayList, enabling adding, viewing, searching, deleting, and sorting records using a Comparator. It also includes random access reading, file property display, and a clean menu-driven interface. The program demonstrates practical use of file I/O, collections, encapsulation, and user interaction in Java.

1. Purpose of the Program

The objective of this program is to demonstrate how file handling, collections, random access, and object-oriented concepts can be combined to create a small student record management system.

It shows how data can be stored permanently using files and how it can be loaded, sorted, deleted, or searched through Java methods.

2. Student Class

A separate Student class is used to represent each student's data.

It contains variables for roll number, name, course, email, and marks. The class also provides a clean method to display the student's details on the screen.

This part represents encapsulation, where data and methods related to a student are grouped together.

3. File Handling (Reading and Writing)

Two major operations are performed on the file students.txt: a.

Loading Data from File

- When the program starts, it reads data from the file using BufferedReader.
- Each line represents one student.
- Data is split and converted into a Student object.
- All records are stored inside an ArrayList.

b. Saving Data Back to File

- When the program exits, all student records are written back to the file using BufferedWriter.
- This ensures that data is not lost when the program closes.
- This demonstrates persistent storage using text files.

4. ArrayList for Storing Multiple Students

An ArrayList is used to store all Student objects dynamically.
It allows:

- Adding more records
- Deleting any record
- Searching through all students
- Sorting based on marks
  ArrayList makes the program flexible and scalable.

5. StudentManager Class

A separate class acts as the controller or manager for all student operations.
It contains functions like:

- Add Student
- View Students
- Search Student
- Delete Student
- Sort Students

This class separates the program's logic from data representation.

6. Sorting Using Comparator

The program sorts students on the basis of marks using a built-in Comparator.
This shows how Java's collection utilities can easily sort objects based on custom values.

7. Random Access File Reading

A RandomAccessFile is used to read specific characters from the file (e.g., first 20 characters).

This shows how Java allows reading files from any position without reading the entire file.

It demonstrates:

- Jumping to specific positions
- Reading selected bytes or characters

This is useful for indexing and partial file reading.

8. Displaying File Properties File properties such as:

- File name
- Path
- Size
- Read/Write permissions

are displayed using the File class.

This helps understand how Java interacts with the operating system's file system.

9. Menu-Based Interaction

The program uses a loop to continuously show a menu:

- Add Student
- View
- Search
- Delete
- Sort
- File Properties
- Random Access
- Exit

This makes user interaction simple and organized.

Each option calls the respective method in StudentManager.

10. Scanner Handling and Closing

A single Scanner object is used for all user inputs.

It is closed safely only at the end so that the program runs without warnings or input errors.

Overall Concepts Demonstrated

Concept	Description
Classes & Objects	Student, StudentManager, FileUtil
Encapsulation	Student details grouped inside one class
Collections	ArrayList to store multiple students
File Handling	BufferedReader, BufferedWriter
Random Access File	Reading specific file positions
Comparator	Sorting students by marks
Exception Handling	try–catch while reading/writing files
User Input Handling	Scanner for menu and student info
Menu-Driven Program	Clear and interactive user experience

Input

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

class Student {
    int roll;
    String name;
    String course;
    String email;
    double marks;

    Student(int roll, String name, String course, String email, double marks) {
        this.roll = roll;
        this.name = name;
        this.course = course;
        this.email = email;
        this.marks = marks;
    }

    @Override
    public String toString() {
        return roll + "," + name + "," + course + "," + email + "," + marks;
    }

    void display() {
        System.out.println("Roll: " + roll);
        System.out.println("Name: " + name);
        System.out.println("Course: " + course);
        System.out.println("Email: " + email);
        System.out.println("Marks: " + marks);
        System.out.println(x: "----------------------------");
    }
```

```
}

class FileUtil {

    static ArrayList<Student> loadStudents() {
        ArrayList<Student> list = new ArrayList<>();

        try (BufferedReader br = new BufferedReader(new FileReader(fileName: "students.txt"))) {

            String line;
            while ((line = br.readLine()) != null) {
                String data[] = line.split(regex: ",");
                list.add(new Student(
                        Integer.parseInt(data[0]),
                        data[1], data[2], data[3],
                        Double.parseDouble(data[4])));
            }

        } catch (FileNotFoundException e) {
            return list; // no file yet, return empty list
        } catch (Exception e) {
            System.out.println(x: "Error loading file.");
        }
        return list;
    }

    static void saveStudents(ArrayList<Student> list) {
        try (BufferedWriter bw = new BufferedWriter(new FileWriter(fileName: "students.txt"))) {

            for (Student s : list) {
                bw.write(s.toString());
```

```
            bw.newLine();
        }

    } catch (Exception e) {
        System.out.println(x: "Error saving file.");
    }
}

static void showFileProperties() {
    File f = new File(pathname: "students.txt");
    System.out.println("File Name: " + f.getName());
    System.out.println("Path: " + f.getAbsolutePath());
    System.out.println("Size: " + f.length() + " bytes");
    System.out.println("Readable: " + f.canRead());
    System.out.println("Writable: " + f.canWrite());
}

static void randomAccessRead() {
    try (RandomAccessFile raf = new RandomAccessFile(pathname: "students.txt", mode: "r")) {
        System.out.println(x: "Reading first 20 characters:");
        for (int i = 0; i < 20 && raf.getFilePointer() < raf.length(); i++) {
            System.out.print((char) raf.read());
        }
        System.out.println(x: "\n----------------------------");
    } catch (Exception e) {
        System.out.println(x: "Random access read error.");
    }
}
}

```

```
93  class StudentManager {
94
95      ArrayList<Student> list;
96
97      StudentManager() {
98          list = FileUtil.loadStudents();
99      }
100
101     void addStudent(Scanner sc) {
102         System.out.print(s: "Roll: ");
103         int r = sc.nextInt();
104         sc.nextLine();
105
106         System.out.print(s: "Name: ");
107         String n = sc.nextLine();
108
109         System.out.print(s: "Course: ");
110         String c = sc.nextLine();
111
112         System.out.print(s: "Email: ");
113         String e = sc.nextLine();
114
115         System.out.print(s: "Marks: ");
116         double m = sc.nextDouble();
117
118         list.add(new Student(r, n, c, e, m));
119         System.out.println(x: "Student Added Successfully!");
120     }
121
122     void viewStudents() {
123         if (list.isEmpty()) {
124             System.out.println(x: "No records available.");
```

```
125             return;
126         }
127         for (Student s : list) s.display();
128     }
129
130     void searchStudent(Scanner sc) {
131         System.out.print(s: "Enter Roll to Search: ");
132         int r = sc.nextInt();
133
134         for (Student s : list) {
135             if (s.roll == r) {
136                 s.display();
137                 return;
138             }
139         }
140         System.out.println(x: "Student not found.");
141     }
142
143     void deleteStudent(Scanner sc) {
144         System.out.print(s: "Enter Roll to Delete: ");
145         int r = sc.nextInt();
146
147         Iterator<Student> it = list.iterator();
148         while (it.hasNext()) {
149             if (it.next().roll == r) {
150                 it.remove();
151                 System.out.println(x: "Record Deleted!");
152                 return;
153             }
```

```
154             }
155             System.out.println(x: "Record not found.");
156         }
157
158         void sortByMarks() {
159             list.sort(Comparator.comparingDouble(s -> s.marks));
160             System.out.println(x: "Sorted by marks (ascending).");
161         }
162     }
163
164     public class assignment4 {
            Run | Debug
165         public static void main(String[] args) {
166
167             Scanner sc = new Scanner(System.in);
168             StudentManager sm = new StudentManager();
169
170             boolean run = true;
171
172             while (run) {
173                 System.out.println(x: "\n----- ASSIGNMENT 4 MENU -----");
174                 System.out.println(x: "1. Add Student");
175                 System.out.println(x: "2. View Students");
176                 System.out.println(x: "3. Search Student");
177                 System.out.println(x: "4. Delete Student");
178                 System.out.println(x: "5. Sort by Marks");
179                 System.out.println(x: "6. File Properties");
180                 System.out.println(x: "7. Random Access Read");
181                 System.out.println(x: "8. Exit");
182                 System.out.print(s: "Enter choice: ");
```

```
183
184             int ch = sc.nextInt();
185
186             switch (ch) {
187                 case 1 -> sm.addStudent(sc);
188                 case 2 -> sm.viewStudents();
189                 case 3 -> sm.searchStudent(sc);
190                 case 4 -> sm.deleteStudent(sc);
191                 case 5 -> sm.sortByMarks();
192                 case 6 -> FileUtil.showFileProperties();
193                 case 7 -> FileUtil.randomAccessRead();
194                 case 8 -> {
195                     FileUtil.saveStudents(sm.list);
196                     System.out.println(x: "Data saved. Exiting…");
197                     run = false;
198                 }
199                 default -> System.out.println(x: "Invalid choice!");
200             }
201         }
202
203         sc.close(); //
204     }
205 }
```

Output

```
----- ASSIGNMENT 4 MENU -----
1. Add Student
2. View Students
3. Search Student
4. Delete Student
5. Sort by Marks
6. File Properties
7. Random Access Read
8. Exit
Enter choice: 1
Roll: 102
Name: khushi
Course: bca
Email: khushi@gmail.com
Marks: 99
Student Added Successfully!

----- ASSIGNMENT 4 MENU -----
1. Add Student
2. View Students
3. Search Student
4. Delete Student
5. Sort by Marks
6. File Properties
7. Random Access Read
8. Exit
Enter choice: 1
Roll: 101
Name: vishal
Course: bca
```

```
Email: vishal@gmail.com
Marks: 98
Student Added Successfully!

----- ASSIGNMENT 4 MENU -----
1. Add Student
2. View Students
3. Search Student
4. Delete Student
5. Sort by Marks
6. File Properties
7. Random Access Read
8. Exit
Enter choice: 2
Roll: 102
Name: khushi
Course: bca
Email: khushi@gmail.com
Marks: 99.0
----------------------------
Roll: 101
Name: vishal
Course: bca
Email: vishal@gmail.com
Marks: 98.0
----------------------------

----- ASSIGNMENT 4 MENU -----
1. Add Student
2. View Students
3. Search Student
4. Delete Student
5. Sort by Marks
6. File Properties
7. Random Access Read
8. Exit
```

```
8. Exit
Enter choice: 102
Invalid choice!

----- ASSIGNMENT 4 MENU -----
1. Add Student
2. View Students
3. Search Student
4. Delete Student
5. Sort by Marks
6. File Properties
7. Random Access Read
8. Exit
Enter choice: 5
Sorted by marks (ascending).

----- ASSIGNMENT 4 MENU -----
1. Add Student
2. View Students
3. Search Student
4. Delete Student
5. Sort by Marks
6. File Properties
7. Random Access Read
8. Exit
Enter choice: 6
File Name: students.txt
Path: C:\Users\Hp\JAVA4\students.txt
Size: 0 bytes
Readable: false
Writable: false

----- ASSIGNMENT 4 MENU -----
1. Add Student
2. View Students
3. Search Student
```

```
3. Search Student
4. Delete Student
5. Sort by Marks
6. File Properties
7. Random Access Read
8. Exit
Enter choice: 7
Random access read error.

----- ASSIGNMENT 4 MENU -----
1. Add Student
2. View Students
3. Search Student
4. Delete Student
5. Sort by Marks
6. File Properties
7. Random Access Read
8. Exit
Enter choice: 8
Data saved. Exiting?
PS C:\Users\Hp\JAVA4>
```