Experiment – 4

Case Study 1: Student Records Management System Scenario: A college wants to maintain digital records of students. Each student has a name, roll number, department, and CGPA. The system should allow adding new records, retrieving all records, and searching for a student by roll number. All data should be stored and retrieved from a file. Question: Design and implement a Java application that handles student records using file handling. Use character streams (like FileWriter, BufferedWriter, FileReader, BufferedReader) to: • Write student details into a file. • Read and display all student records. • Search for a student by roll number. Discuss how you handle file exceptions and what would happen if the file doesn't exist.

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

class Student {
    String name;
    String rollNo;
    String department;
    double cgpa;

    public Student(String name, String rollNo, String department, double cgpa) {
        this.name = name;
        this.rollNo = rollNo;
        this.department = department;
        this.cgpa = cgpa;
    }

    @Override
    public String toString() {
        return name + "," + rollNo + "," + department + "," + cgpa;
    }

    public static Student fromString(String line) {
        String[] parts = line.split(",");
        if (parts.length == 4) {
            return new Student(parts[0], parts[1], parts[2],
Double.parseDouble(parts[3]));
```

```java
        }
        return null;
    }
}

public class StudentRecordsApp {
    private static final String FILE_NAME = "students.txt";

    // Add a new student to the file
    public static void addStudent(Student student) {
        try (BufferedWriter writer = new BufferedWriter(new
FileWriter(FILE_NAME, true))) {
            writer.write(student.toString());
            writer.newLine();
            System.out.println("Student added successfully.");
        } catch (IOException e) {
            System.out.println("Error writing to file: " + e.getMessage());
        }
    }

    // Read and display all students
    public static void displayAllStudents() {
        try (BufferedReader reader = new BufferedReader(new
FileReader(FILE_NAME))) {
            String line;
            System.out.println("All Students:");
            while ((line = reader.readLine()) != null) {
                Student student = Student.fromString(line);
                if (student != null) {
                    System.out.println("Name: " + student.name + ", Roll No: " +
student.rollNo +
                            ", Dept: " + student.department + ", CGPA: " + student.cgpa);
                }
            }
        } catch (FileNotFoundException e) {
            System.out.println("File not found. No records to display.");
        } catch (IOException e) {
            System.out.println("Error reading file: " + e.getMessage());
        }
    }
    // Search for a student by roll number
    public static void searchByRollNumber(String rollNo) {
        boolean found = false;
        try (BufferedReader reader = new BufferedReader(new
FileReader(FILE_NAME))) {
            String line;
            while ((line = reader.readLine()) != null) {
                Student student = Student.fromString(line);
                if (student != null && student.rollNo.equalsIgnoreCase(rollNo)) {
                    System.out.println("Student Found:");
                    System.out.println("Name: " + student.name + ", Roll No: " +
student.rollNo +
                            ", Dept: " + student.department + ", CGPA: " + student.cgpa);
                    found = true;
                    break;
                }
            }
            if (!found) {
                System.out.println("Student with roll number " + rollNo + " not found.");
            }
        } catch (FileNotFoundException e) {
            System.out.println("File not found. No records exist.");
        } catch (IOException e) {
            System.out.println("Error reading file: " + e.getMessage());
        }
    }

    // Menu-driven interface
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int choice;
        do {
            System.out.println("\nStudent Records Management System");
            System.out.println("1. Add Student");
            System.out.println("2. Display All Students");
            System.out.println("3. Search by Roll Number");
            System.out.println("4. Exit");
            System.out.print("Enter choice: ");
            choice = scanner.nextInt();
            scanner.nextLine(); // consume newline

            switch (choice) {
                case 1:
                    System.out.print("Enter Name: ");
                    String name = scanner.nextLine();
                    System.out.print("Enter Roll Number: ");
                    String rollNo = scanner.nextLine();
                    System.out.print("Enter Department: ");
                    String dept = scanner.nextLine();
                    System.out.print("Enter CGPA: ");
                    double cgpa = scanner.nextDouble();
                    scanner.nextLine();
                    addStudent(new Student(name, rollNo, dept, cgpa));
                    break;

                case 2:
                    displayAllStudents();
                    break;

                case 3:
                    System.out.print("Enter Roll Number to Search: ");
                    String searchRollNo = scanner.nextLine();
                    searchByRollNumber(searchRollNo);
                    break;

                case 4:
                    System.out.println("Exiting Program...");
                    break;

                default:
                    System.out.println("Invalid choice!");
            }
        } while (choice != 4);
        scanner.close();
    }
}
```

Output – javac StudentRecordsApp.java

```
   java StudentRecordsApp
Student Records Management System
1. Add Student
2. Display All Students
3. Search by Roll Number
4. Exit
Enter choice: 1
Enter Name: ansh
Enter Roll Number: 36
Enter Department: it
Enter CGPA: 9
Student added successfully.
Student Records Management System
1. Add Student
2. Display All Students
3. Search by Roll Number
4. Exit
Enter choice: 2
All Students:
Name: ansh, Roll No: 36, Dept: it, CGPA: 9.0
Student Records Management System
1. Add Student
```

```
2. Display All Students

3. Search by Roll Number

4. Exit

Enter choice: 4

Exiting Program...
```

Case Study 2: Library Book Issue Tracker Scenario: A library maintains a file-based record of books issued to members. Each entry contains the book ID, book name, member ID, issue date, and return date. Question: Develop a Java program using byte streams (like FileOutputStream, FileInputStream) to: • Add book issue records. • Display all issue records. • Update the return date for a specific record. Explain your approach to reading/writing binary data and how you ensure data consistency in case of interrupted file operations.

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

class BookIssue implements Serializable {
    private static final long serialVersionUID = 1L;

    String bookId;
    String bookName;
    String memberId;
    String issueDate;
    String returnDate;

    public BookIssue(String bookId, String bookName, String memberId, String
issueDate, String returnDate) {
        this.bookId = bookId;
        this.bookName = bookName;
        this.memberId = memberId;
        this.issueDate = issueDate;
        this.returnDate = returnDate;
    }

    @Override
    public String toString() {
        return "Book ID: " + bookId + ", Book Name: " + bookName +
                ", Member ID: " + memberId + ", Issue Date: " + issueDate +
                ", Return Date: " + returnDate;
    }
}

public class LibraryTrackerApp {
    private static final String FILE_NAME = "book_issues.dat";

    // Add a new book issue record
    public static void addBookIssue(BookIssue issue) {
        List<BookIssue> records = readAllIssues();
        records.add(issue);
        writeAllIssues(records);
        System.out.println("Book issue record added.");
    }

    // Read all issue records
    public static List<BookIssue> readAllIssues() {
        List<BookIssue> issues = new ArrayList<>();
        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(FILE_NAME))) {
            issues = (List<BookIssue>) ois.readObject();
        } catch (FileNotFoundException e) {
            // File might not exist initially — no action needed
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Error reading file: " + e.getMessage());
        }
        return issues;
    }

    // Write all issue records (used to update file)
    public static void writeAllIssues(List<BookIssue> issues) {
        // Write to a temp file first to ensure data consistency
        File tempFile = new File("temp_" + FILE_NAME);
        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(tempFile))) {
            oos.writeObject(issues);
            oos.flush();
            // Replace original file with temp file
            File originalFile = new File(FILE_NAME);
            if (originalFile.exists()) {
                originalFile.delete();
            }
            tempFile.renameTo(originalFile);

        } catch (IOException e) {
            System.out.println("Error writing to file: " + e.getMessage());
        }
    }

    // Display all book issue records
    public static void displayAllIssues() {
        List<BookIssue> records = readAllIssues();
        if (records.isEmpty()) {
            System.out.println("No records found.");
        } else {
            for (BookIssue record : records) {
                System.out.println(record);
            }
        }
    }

    // Update return date for a specific book ID and member ID
    public static void updateReturnDate(String bookId, String memberId, String
newReturnDate) {
        List<BookIssue> records = readAllIssues();
        boolean updated = false;
        for (BookIssue issue : records) {
            if (issue.bookId.equals(bookId) && issue.memberId.equals(memberId)) {
                issue.returnDate = newReturnDate;
                updated = true;
                break;
            }
        }
        if (updated) {
            writeAllIssues(records);
            System.out.println("Return date updated.");
        } else {
            System.out.println("Record not found.");
        }
    }

    // Menu
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int choice;

        do {
            System.out.println("\nLibrary Book Issue Tracker");
            System.out.println("1. Add Book Issue Record");
            System.out.println("2. Display All Records");
            System.out.println("3. Update Return Date");
            System.out.println("4. Exit");
            System.out.print("Enter choice: ");
            choice = scanner.nextInt(); scanner.nextLine();

            switch (choice) {
                case 1:
                    System.out.print("Enter Book ID: ");
                    String bookId = scanner.nextLine();
                    System.out.print("Enter Book Name: ");
                    String bookName = scanner.nextLine();
                    System.out.print("Enter Member ID: ");
                    String memberId = scanner.nextLine();
                    System.out.print("Enter Issue Date (dd-mm-yyyy): ");
                    String issueDate = scanner.nextLine();
                    System.out.print("Enter Return Date (dd-mm-yyyy): ");
                    String returnDate = scanner.nextLine();
                    addBookIssue(new BookIssue(bookId, bookName, memberId,
issueDate, returnDate));
                    break;

                case 2:
                    displayAllIssues();
                    break;

                case 3:
```

```java
                    System.out.print("Enter Book ID: ");
                    String bId = scanner.nextLine();
                    System.out.print("Enter Member ID: ");
                    String mId = scanner.nextLine();
                    System.out.print("Enter New Return Date (dd-mm-yyyy): ");
                    String newReturn = scanner.nextLine();
                    updateReturnDate(bId, mId, newReturn);
                    break;

                case 4:
                    System.out.println("Exiting program.");
                    break;

                default:
                    System.out.println("Invalid choice.");
            }

        } while (choice != 4);

        scanner.close();
    }
}
```

Output – javac LibraryTrackerApp.java

```
      java LibraryTrackerApp

     Library Book Issue Tracker

     1. Add Book Issue Record

     2. Display All Records

     3. Update Return Date

     4. Exit

     Enter choice: 1

     Enter Book ID: 1
```

```
Enter Book Name: java

Enter Member ID: 12

Enter Issue Date (dd-mm-yyyy): 12-3-2024

Enter Return Date (dd-mm-yyyy): 12-3-2025

Book issue record added.

Library Book Issue Tracker

1. Add Book Issue Record

2. Display All Records

3. Update Return Date

4. Exit

Enter choice: 2

No records found.

Library Book Issue Tracker

1. Add Book Issue Record

2. Display All Records

3. Update Return Date

4. Exit

Enter choice: 4

Exiting program.
```

Case Study 3: Daily Sales Logger for a Retail Store Scenario: A retail store logs daily sales transactions into a file. Each transaction includes item name, quantity sold, price per item, and date. Question: Create a Java application that: • Appends new sales transactions to a file daily. • Reads and summarizes total sales for a specific date. • Handles exception like malformed entries in the file. Demonstrate how you use BufferedReader and BufferedWriter with file append mode, and manage file access efficiently.

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

public class DailySalesLoggerApp {
    private static final String FILE_NAME = "sales_log.txt";

    // Add a new sales transaction to the file (append mode)
    public static void appendTransaction(String itemName, int quantity, double
price, String date) {
        try (BufferedWriter writer = new BufferedWriter(new
FileWriter(FILE_NAME, true))) {
            String record = itemName + "," + quantity + "," + price + "," + date;
            writer.write(record);
            writer.newLine();
            System.out.println("Transaction logged successfully.");
        } catch (IOException e) {
            System.out.println("Error writing to file: " + e.getMessage());
        }
    }

    // Summarize total sales for a given date
    public static void summarizeSalesForDate(String targetDate) {
        double totalSales = 0.0;
        int malformedCount = 0;

        try (BufferedReader reader = new BufferedReader(new
FileReader(FILE_NAME))) {
            String line;
            while ((line = reader.readLine()) != null) {
                try {
                    String[] parts = line.split(",");
                    if (parts.length != 4) {
                        throw new IllegalArgumentException("Invalid record format");
                    }

                    String item = parts[0].trim();
                    int quantity = Integer.parseInt(parts[1].trim());
                    double price = Double.parseDouble(parts[2].trim());
                    String date = parts[3].trim();

                    if (date.equals(targetDate)) {
                        totalSales += quantity * price;
                    }

                } catch (Exception e) {
                    malformedCount++;
                    // Continue reading other lines
                }
            }

            System.out.printf("Total sales on %s: $%.2f\n", targetDate, totalSales);
            if (malformedCount > 0) {
                System.out.println("Ignored malformed entries: " + malformedCount);
            }

        } catch (FileNotFoundException e) {
            System.out.println("Sales file not found.");
        } catch (IOException e) {
            System.out.println("Error reading file: " + e.getMessage());
        }
    }

    // Main menu
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int choice;

        do {
            System.out.println("\n--- Daily Sales Logger ---");
            System.out.println("1. Add New Sale");
            System.out.println("2. View Total Sales by Date");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt(); scanner.nextLine(); // consume newline

            switch (choice) {
                case 1:
                    System.out.print("Enter item name: ");
                    String itemName = scanner.nextLine();
                    System.out.print("Enter quantity sold: ");
                    int quantity = scanner.nextInt();
                    System.out.print("Enter price per item: ");
                    double price = scanner.nextDouble(); scanner.nextLine();
                    System.out.print("Enter date (YYYY-MM-DD): ");
                    String date = scanner.nextLine();

                    appendTransaction(itemName, quantity, price, date);
                    break;

                case 2:
                    System.out.print("Enter date to summarize (YYYY-MM-DD): ");
                    String summaryDate = scanner.nextLine();
                    summarizeSalesForDate(summaryDate);
                    break;

                case 3:
                    System.out.println("Exiting. Goodbye!");
                    break;

                default:
                    System.out.println("Invalid choice.");
            }
        } while (choice != 3);

        scanner.close();
    }
}
```

Output – javac DailySalesLoggerApp.java

java DailySalesLoggerApp

```
--- Daily Sales Logger ---

1. Add New Sale
```

```
2. View Total Sales by Date

3. Exit

Enter your choice: 1

Enter item name: adarsh

Enter quantity sold: 1

Enter price per item: 0.01

Enter date (YYYY-MM-DD): 25-05-2025

Transaction logged successfully.

--- Daily Sales Logger ---

1. Add New Sale

2. View Total Sales by Date

3. Exit

Enter your choice: 2

Enter date to summarize (YYYY-MM-DD): 25-05-2025

Total sales on 25-05-2025: $0.01

--- Daily Sales Logger ---

1. Add New Sale

2. View Total Sales by Date

3. Exit

Enter your choice: 4

Invalid choice.

--- Daily Sales Logger ---

1. Add New Sale
```

```
2. View Total Sales by Date

3. Exit

Enter your choice: 3

Exiting. Goodbye!
```

ANSH PANDEY

2300290130036

IT- (A)-36