TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THÀNH PHỐ HỒ CHÍ MINH	**ĐỀ MINH HỌA THI GIỮA KÌ 01**
----------	**Môn thi:** Lập trình JAVA trong CNTT
KHOA CÔNG NGHỆ THÔNG TIN	**Lớp:** DHTH17
	Ngày thi: 2022
	Thời gian: 70 phút

Họ tên: ..... MSSV……..

Mô tả yêu cầu:

Một trường đại học cần xây dựng một hệ thống quản lý danh sách sinh viên. Thông tin của đối tượng sinh viên bao gồm:

- **Mã sinh viên** (*student_id*)
- **Họ và tên sinh viên** (*student_name*)
- **Điểm trung bình** (*student_result*)
- **Xếp loại học tập** (*student_rank*).

Xếp loại học tập được phân loại như sau:

- Nếu điểm trung bình từ 8.5 trở lên: **Giỏi (Excellent)**
- Từ 6.5 đến dưới 8.5: **Khá (Good)**
- Từ 5.0 đến dưới 6.5: **Trung bình (Average)**
- Dưới 5.0: **Yếu (Poor)**

Sinh viên sử dụng danh sách liên kết (*LinkedList*) để xây dựng chương trình quản lý sinh viên, với menu thực hiện các chức năng sau:

Câu 1: Đọc danh sách sinh viên từ file (*student_id, student_name*).

Câu 2: Lưu danh sách sinh viên ra file.

Câu 3: Xuất thông tin danh sách sinh viên, bao gồm: mã sinh viên, họ tên, điểm trung bình, xếp loại.

Câu 4: Thêm sinh viên vào danh sách, nhập thông tin chi tiết của sinh viên từ bàn phím.

Câu 5: Tìm kiếm sinh viên theo mã sinh viên (*student_id*).

Câu 6: Sắp xếp danh sách sinh viên theo họ tên (*student_name*).

Câu 7: Xóa sinh viên khỏi danh sách theo mã sinh viên (*student_id*).

Câu 8: Cập nhật điểm trung bình của sinh viên, đồng thời cập nhật lại xếp loại học tập dựa trên điểm mới.

Hệ thống cần đảm bảo hoạt động ổn định và hiển thị thông báo rõ ràng sau khi thực hiện mỗi chức năng.

----------Hết----------

1. Cấu trúc lớp Student.java

```
import java.io.Serializable;

public class Student implements Serializable {
    private String student_id;
    private String student_name;
    private double student_result;
    private String student_rank;

    public Student(String student_id, String student_name) {
        this.student_id = student_id;
        this.student_name = student_name;
        this.student_result = 0.0;
        this.student_rank = "Not Graded";
    }

    public String getStudent_id() {
        return student_id;
    }

    public void setStudent_id(String student_id) {
        this.student_id = student_id;
    }

    public String getStudent_name() {
        return student_name;
    }

    public void setStudent_name(String student_name) {
        this.student_name = student_name;
    }

    public double getStudent_result() {
        return student_result;
    }

    public void setStudent_result(double student_result) {
        this.student_result = student_result;
        setStudent_rank(); // Cập nhật xếp loại mỗi khi thay đổi điểm
    }

    public String getStudent_rank() {
        return student_rank;
    }

    public void setStudent_rank() {
        if (student_result >= 8.5) {
            this.student_rank = "Excellent";
```

```
        } else if (student_result >= 6.5) {
            this.student_rank = "Good";
        } else if (student_result >= 5.0) {
            this.student_rank = "Average";
        } else {
            this.student_rank = "Poor";
        }
    }

    @Override
    public String toString() {
        return String.format("%-10s %-20s %-10.2f %-10s", student_id,
student_name, student_result, student_rank);
    }
}
```

2. Lớp Student_Manage (Quản lý sinh viên)

Import các thư viện cần thiết

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

Khai báo lớp Student_Manage

```
public class Student_Manage {
    static Scanner scan = new Scanner(System.in);
    static LinkedList<Student> studentList = new LinkedList<>();

    // Đọc danh sách sinh viên từ file
    @SuppressWarnings("unchecked")
    public static LinkedList<Student> readFromFile(String path) {
        LinkedList<Student> list = null;
        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(path))) {
            list = (LinkedList<Student>) ois.readObject();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return list != null ? list : new LinkedList<>();
    }
```

```
    // Lưu danh sách sinh viên ra file
    public static boolean saveToFile(LinkedList<Student> list, String path) {
        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(path))) {
            oos.writeObject(list);
            return true;
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }

    // Xuất thông tin danh sách sinh viên
    public static void displayStudentList(LinkedList<Student> list) {
        System.out.printf("%-10s %-20s %-10s %-10s\n", "ID", "Name", "Result",
"Rank");
        for (Student student : list) {
            System.out.println(student);
        }
    }


// Thêm sinh viên mới vào danh sách

    public static void addStudent(LinkedList<Student> list) {
        System.out.println("Enter Student ID: ");
        String id = scan.nextLine();
        System.out.println("Enter Student Name: ");
        String name = scan.nextLine();
        Student student = new Student(id, name);
        list.add(student);
    }

    // Tìm kiếm sinh viên theo mã sinh viên
    public static int findStudentById(LinkedList<Student> list, String id) {
        for (int i = 0; i < list.size(); i++) {
            if (list.get(i).getStudent_id().equalsIgnoreCase(id)) {
                return i;
            }
        }
        return -1;
    }

    // Xóa sinh viên khỏi danh sách
    public static void deleteStudent(LinkedList<Student> list) {
        System.out.println("Enter Student ID to delete: ");
        String id = scan.nextLine();
        int index = findStudentById(list, id);
        if (index != -1) {
```

```
            list.remove(index);
            System.out.println("Student deleted successfully.");
        } else {
            System.out.println("Student not found.");
        }
    }

    // Sắp xếp danh sách sinh viên theo họ tên
    public static void sortStudentsByName(LinkedList<Student> list) {
        list.sort((s1, s2) ->
s1.getStudent_name().compareToIgnoreCase(s2.getStudent_name()));
        System.out.println("List sorted by name.");
    }

    // Cập nhật điểm trung bình của sinh viên
    public static void updateStudentResult(LinkedList<Student> list) {
        System.out.println("Enter Student ID to update result: ");
        String id = scan.nextLine();
        int index = findStudentById(list, id);
        if (index != -1) {
            System.out.println("Enter new result: ");
            double result = scan.nextDouble();
            scan.nextLine(); // Clear buffer
            list.get(index).setStudent_result(result);
            System.out.println("Student result updated.");
        } else {
            System.out.println("Student not found.");
        }
    }
```

```
// Menu chính
    public static void main(String[] args) {
        String filePath = "students.dat";
        int choice;
        do {
            System.out.println("\n--- STUDENT MANAGEMENT SYSTEM ---");
            System.out.println("1. Read Student List from File");
            System.out.println("2. Save Student List to File");
            System.out.println("3. Display Student List");
            System.out.println("4. Add Student");
            System.out.println("5. Find Student by ID");
            System.out.println("6. Delete Student by ID");
            System.out.println("7. Sort Students by Name");
            System.out.println("8. Update Student Result");
            System.out.println("0. Exit");
            System.out.print("Enter your choice: ");
            choice = scan.nextInt();
            scan.nextLine(); // Clear buffer
            switch (choice) {
                case 1 -> studentList = readFromFile(filePath);
                case 2 -> {
                    if (saveToFile(studentList, filePath))
System.out.println("Data saved successfully.");
                    else System.out.println("Error saving data.");
                }
                case 3 -> displayStudentList(studentList);
                case 4 -> addStudent(studentList);
                case 5 -> {
                    System.out.println("Enter Student ID to search: ");
                    String id = scan.nextLine();
                    int index = findStudentById(studentList, id);
                         if (index != -1)
                   System.out.println(studentList.get(index));
                    else System.out.println("Student not found.");
                }
                case 6 -> deleteStudent(studentList);
                case 7 -> sortStudentsByName(studentList);
                case 8 -> updateStudentResult(studentList);
                case 0 -> System.out.println("Exiting program.");
                default -> System.out.println("Invalid choice. Please try
again.");
            }
        } while (choice != 0);
    }
}
```

----------Hết----------