**ĐỌC GHI MẢNG ĐỐI TƯỢNG VÀ TỆP TIN**

**(Lưu trữ dài lâu)**

Đây là một ví dụ "trọn gói" (self-contained) minh họa toàn bộ quá trình:

1. Định nghĩa một lớp Student (có implements Serializable).
2. Tạo một ArrayList chứa các đối tượng Student (dữ liệu mẫu).
3. **Ghi (Write):** Lưu *toàn bộ* ArrayList này xuống file students.dat.
4. **Đọc (Read):** Tải ArrayList từ file students.dat lên một danh sách mới.
5. In danh sách mới ra để kiểm chứng.

Bạn chỉ cần tạo 2 file Student.java và MainApp.java là có thể chạy ngay.

**File 1: Lớp dữ liệu (Phải Serializable)**

Đây là "khuôn mẫu" cho đối tượng mà chúng ta muốn lưu.

**Student.java**

Java

import java.io.Serializable;

// BƯỚC 1: Phải "đánh dấu" lớp này là có thể Tuần tự hóa

public class Student implements Serializable {

    // BƯỚC 2: (Good practice) Thêm serialVersionUID để đảm bảo

    // phiên bản lớp khi Ghi và Đọc là nhất quán.

    private static final long serialVersionUID = 1L;

    private String id;

    private String name;

    private double gpa;

    public Student(String id, String name, double gpa) {

        this.id = id;

        this.name = name;

        this.gpa = gpa;

    }

    // BƯỚC 3: Thêm hàm toString() để có thể in ra kiểm tra

    @Override

    public String toString() {

        return "Student{" +

                "id='" + id + '\'' +

                ", name='" + name + '\'' +

                ", gpa=" + gpa +

                '}';

    }

}

**File 2: Chương trình Ghi và Đọc**

Đây là file main thực hiện việc lưu và tải.

**MainApp.java**

Java

import java.io.\*;

import java.util.ArrayList;

public class MainApp {

    private static final String FILENAME = "students.dat";

    public static void main(String[] args) {

        // 1. Tạo dữ liệu mẫu (Một danh sách các đối tượng)

        ArrayList<Student> listToWrite = new ArrayList<>();

        listToWrite.add(new Student("B21DCCN001", "An Nguyen", 3.2));

        listToWrite.add(new Student("B21DCCN002", "Binh Le", 2.8));

        listToWrite.add(new Student("B21DCCN003", "Chi Pham", 3.5));

        // 2. GỌI HÀM GHI FILE

        writeObjectsToFile(listToWrite);

        // 3. GỌI HÀM ĐỌC FILE

        ArrayList<Student> listFromRead = readObjectsFromFile();

        // 4. KIỂM CHỨNG KẾT QUẢ

        System.out.println("\n--- Dữ liệu đã đọc từ file " + FILENAME + ": ---");

        if (listFromRead != null) {

            for (Student s : listFromRead) {

                System.out.println(s);

            }

        }

    }

    /\*\*

     \* Ghi một ArrayList các đối tượng Student xuống file.

     \* @param studentList Danh sách cần ghi

     \*/

    public static void writeObjectsToFile(ArrayList<Student> studentList) {

        System.out.println("--- Bắt đầu GHI file: " + FILENAME + " ---");

        // Sử dụng try-with-resources để tự động đóng stream

        try (FileOutputStream fos = new FileOutputStream(FILENAME);

             ObjectOutputStream oos = new ObjectOutputStream(fos)) {

            // GHI TOÀN BỘ ArrayList xuống file CHỈ BẰNG 1 LỆNH

            oos.writeObject(studentList);

            System.out.println("Ghi file thành công!");

        } catch (IOException e) {

            System.out.println("Lỗi khi ghi file: " + e.getMessage());

            e.printStackTrace();

        }

    }

    /\*\*

     \* Đọc một ArrayList các đối tượng Student từ file.

     \* @return Danh sách đã đọc, hoặc null nếu có lỗi.

     \*/

    public static ArrayList<Student> readObjectsFromFile() {

        System.out.println("\n--- Bắt đầu ĐỌC file: " + FILENAME + " ---");

        ArrayList<Student> studentList = null;

        try (FileInputStream fis = new FileInputStream(FILENAME);

             ObjectInputStream ois = new ObjectInputStream(fis)) {

            // ĐỌC TOÀN BỘ ArrayList lên CHỈ BẰNG 1 LỆNH

            // Cần ép kiểu (cast) về đúng kiểu dữ liệu ban đầu

            studentList = (ArrayList<Student>) ois.readObject();

            System.out.println("Đọc file thành công!");

        } catch (FileNotFoundException e) {

            System.out.println("Lỗi: Không tìm thấy file " + FILENAME);

        } catch (IOException | ClassNotFoundException e) {

            // ClassNotFoundException xảy ra nếu lớp Student không được tìm thấy

            System.out.println("Lỗi khi đọc file: " + e.getMessage());

            e.printStackTrace();

        }

        return studentList;

    }

}

**Kết quả (Console Output)**

Khi bạn chạy file MainApp.java, bạn sẽ thấy kết quả như sau:

Plaintext

--- Bắt đầu GHI file: students.dat ---

Ghi file thành công!

--- Bắt đầu ĐỌC file: students.dat ---

Đọc file thành công!

--- Dữ liệu đã đọc từ file students.dat: ---

Student{id='B21DCCN001', name='An Nguyen', gpa=3.2}

Student{id='B21DCCN002', name='Binh Le', gpa=2.8}

Student{id='B21DCCN003', name='Chi Pham', gpa=3.5}

**Giải thích:** Như bạn thấy, chúng ta đã "đóng băng" toàn bộ ArrayList (chứa 3 đối tượng Student) và lưu nó xuống file students.dat. Sau đó, ở một hàm khác (giả lập như một lần chạy chương trình mới), chúng ta đã "rã đông" file đó và tải lại toàn bộ ArrayList vào bộ nhớ chỉ bằng hai lệnh writeObject và readObject.

* Lớp dữ liệu: Bắt buộc phải implements Serializable. Việc thêm serialVersionUID là một thói quen lập trình tốt để quản lý phiên bản của lớp khi Ghi/Đọc.
* Quá trình Ghi: Sử dụng ObjectOutputStream để ghi toàn bộ ArrayList xuống file chỉ bằng một lệnh duy nhất là writeObject(listToWrite).
* Quá trình Đọc: Sử dụng ObjectInputStream để đọc lại toàn bộ ArrayList từ file cũng bằng một lệnh readObject(), sau đó thực hiện ép kiểu (cast) về (ArrayList<Student>).
* Try-with-resources: Ví dụ đã sử dụng cú pháp này (với try (...)), đây là cách làm hiện đại và an toàn để đảm bảo các luồng (streams) như FileOutputStream hay ObjectInputStream được tự động đóng lại sau khi sử dụng xong, tránh rò rỉ tài nguyên.