**PHIẾU HỌC TẬP CHỦ ĐỘNG - BÀI 14**

**Chủ đề: Xử lý File I/O - Lưu trữ dữ liệu**

* **Tên Sinh viên:** Đào Văn Dương
* **Mã Lớp:** CNTT 1807

Tham khảo: https://youtu.be/YjX-b84oWvo?si=iPkuIV\_8Paj75ZcS

**Phần 1: Kiến thức cốt lõi (Khái niệm)**

**1. Điền vào chỗ trống:**

* **Luồng (Stream):** Là một dòng **byte** (dữ liệu) liên tục, dùng để đọc/ghi dữ liệu từ một nguồn.
* **Serialization (Tuần tự hóa):** Là quá trình chuyển đổi trạng thái của một **đối tượng** (object) thành một luồng byte để lưu trữ.
* **implements Serializable**: Là một "interface đánh dấu" (marker interface), dùng để "báo" cho Java biết rằng đối tượng của lớp này **được phép** tuần tự hóa.

**2. So sánh nhanh:**

|  |  |  |
| --- | --- | --- |
| **Lớp** | **Mục đích (Đọc hay Ghi?)** | **Phương thức chính** |
| ObjectOutputStream | **Ghi** | writeObject(obj) |
| ObjectInputStream | **Đọc** | readObject() |

**3. Cú pháp mới: try-with-resources**

Cú pháp này cho phép khai báo các tài nguyên (như Stream) *bên trong* dấu () của try. Lợi ích lớn nhất là nó sẽ **tự động** đóng các tài nguyên này, bất kể try thành công hay thất bại (có Exception).

**Phần 2: Ví dụ thực hành (Bắt buộc)**

**Bối cảnh:** Chúng ta sẽ nâng cấp lớp QuanLyNhanSu (từ Bài 12) để thêm 2 chức năng: luuFile() (để lưu ArrayList xuống file) và docFile() (để đọc ArrayList từ file lên).

**Yêu cầu:**

1. **Quan trọng:** Tạo file **NhanVien.java** (abstract) **MỚI** như bên dưới.
2. **Sử dụng lại** 2 file **LapTrinhVien.java** và **KiemThuVien.java** (chúng sẽ tự động được Serializable vì kế thừa từ **NhanVien**).
3. Tạo 2 file **MỚI**: **QuanLyNhanSu**.**java** và **Main.java**.
4. Copy chính xác nội dung dưới đây vào 3 file (NhanVien, QuanLyNhanSu, Main).
5. Hoàn thành các yêu cầu // TODO: trong file **QuanLyNhanSu.java**.
6. **Không sửa** file Main.java và NhanVien.java.
7. Chạy file Main.java. Kiểm tra thư mục dự án của bạn, bạn sẽ thấy một file mới tên là nhansu.dat.
8. Dán **code QuanLyNhanSu.java đã hoàn thiện** và **kết quả Console Output** vào bên dưới.

**File 1: NhanVien.java (Lớp Cha - Cần cập nhật để implements Serializable)**

import java.io.Serializable; // TODO: Phai import de su dung

// TODO 1: (CUC KY QUAN TRONG)

// Them 'implements Serializable' vao sau ten lop

// Neu khong, chuong trinh se nem ra 'NotSerializableException'

public abstract class NhanVien implements Serializable {

// Thuoc tinh nay de dam bao phien ban cua lop khi luu va doc la mot

private static final long serialVersionUID = 1L;

protected String maNV;

protected String ten;

protected double luongCoBan;

public NhanVien(String maNV, String ten, double luongCoBan) {

this.maNV = maNV;

this.ten = ten;

this.luongCoBan = luongCoBan;

}

public abstract double tinhLuong();

public String getTen() { return this.ten; }

}

**File 2: QuanLyNhanSu.java (File "xương sườn" cần hoàn thiện)**

import java.io.\*; // Import tat ca cac lop trong java.io

import java.util.ArrayList;

public class QuanLyNhanSu {

// ArrayList tu dong cung la Serializable roi

private ArrayList<NhanVien> danhSach;

public QuanLyNhanSu() {

this.danhSach = new ArrayList<>();

}

// Ham tu Bai 12

public void themNhanVien(NhanVien nv) {

this.danhSach.add(nv);

}

// Ham tu Bai 12

public void inDanhSach() {

if (danhSach.isEmpty()) {

System.out.println("Danh sach rong!");

return;

}

for (NhanVien nv : this.danhSach) {

System.out.println("Ten: " + nv.getTen() + ", Luong: " + nv.tinhLuong());

}

}

// TODO 2: Tao phuong thuc 'public void luuFile(String tenFile)'

public void luuFile(String tenFile) {

// Su dung try-with-resources de tu dong dong file

try (FileOutputStream fos = new FileOutputStream(tenFile);

ObjectOutputStream oos = new ObjectOutputStream(fos)) {

// TODO 3: Goi phuong thuc 'writeObject()' de luu

// TOAN BO doi tuong 'danhSach' (ArrayList) xuong file

oos.writeObject(this.danhSach);

System.out.println("Luu file thanh cong!");

} catch (IOException e) {

System.out.println("Loi khi luu file: " + e.getMessage());

e.printStackTrace();

}

}

// TODO 4: Tao phuong thuc 'public void docFile(String tenFile)'

public void docFile(String tenFile) {

// Su dung try-with-resources de tu dong dong file

try (FileInputStream fis = new FileInputStream(tenFile);

ObjectInputStream ois = new ObjectInputStream(fis)) {

// TODO 5: Goi phuong thuc 'readObject()' de doc doi tuong len

// Va EP KIEU (cast) no ve 'ArrayList<NhanVien>'

// roi gan vao 'this.danhSach'

this.danhSach = (ArrayList<NhanVien>) ois.readObject();

System.out.println("Doc file thanh cong!");

} catch (FileNotFoundException e) {

System.out.println("Khong tim thay file de doc! Tao danh sach moi.");

this.danhSach = new ArrayList<>(); // Khoi tao danh sach rong

} catch (IOException | ClassNotFoundException e) {

// Bat 2 loai Exception cung luc

System.out.println("Loi khi doc file: " + e.getMessage());

e.printStackTrace();

}

}

}

**File 3: Main.java (File thực thi - KHÔNG SỬA)** *(Giả định LapTrinhVien.java và KiemThuVien.java đã tồn tại)*

public class Main {

public static void main(String[] args) {

QuanLyNhanSu ql = new QuanLyNhanSu();

System.out.println("--- THEM NHAN VIEN ---");

// Giả định các lớp này kế thừa từ NhanVien

// LapTrinhVien(String maNV, String ten, double luongCoBan, int soGioOT)

// KiemThuVien(String maNV, String ten, double luongCoBan, int soBugTimDuoc)

ql.themNhanVien(new LapTrinhVien("DEV001", "Binh", 12000000, 10));

ql.themNhanVien(new KiemThuVien("QC001", "Chi", 11000000, 50));

System.out.println("--- LUU FILE ---");

ql.luuFile("nhansu.dat");

System.out.println("\n--- TAO DOI TUONG QUAN LY MOI (RONG) ---");

QuanLyNhanSu qlMoi = new QuanLyNhanSu();

System.out.println("Danh sach cua qlMoi truoc khi doc file:");

qlMoi.inDanhSach(); // Se in ra "Danh sach rong!"

System.out.println("\n--- DOC FILE VAO qlMoi ---");

qlMoi.docFile("nhansu.dat");

System.out.println("\n--- DANH SACH CUA qlMoi SAU KHI DOC FILE ---");

qlMoi.inDanhSach(); // Neu thanh cong, se in ra 2 nhan vien

}

}

*(Lưu ý: Để* ***Main.java*** *chạy, bạn cần có 2 lớp* ***LapTrinhVien.java*** *và* ***KiemThuVien.java*** *từ các bài học trước, và đảm bảo chúng kế thừa từ* ***NhanVien.java*** *mới nhất này).*

**NỘP BÀI THỰC HÀNH:**

**(1) Dán code QuanLyNhanSu.java đã hoàn thiện của bạn vào đây:**

import java.io.\*; // Import tất cả các lớp trong java.io

import java.util.ArrayList;

public class QuanLyNhanSu {

    // ArrayList tự động cũng là Serializable

    private ArrayList<NhanVien> danhSach;

    public QuanLyNhanSu() {

        this.danhSach = new ArrayList<>();

    }

    // Thêm nhân viên vào danh sách

    public void themNhanVien(NhanVien nv) {

        this.danhSach.add(nv);

    }

    // In danh sách nhân viên

    public void inDanhSach() {

        if (danhSach.isEmpty()) {

            System.out.println("Danh sach rong!");

            return;

        }

        for (NhanVien nv : this.danhSach) {

            System.out.println("Ten: " + nv.getTen() + ", Luong: " + nv.tinhLuong());

        }

    }

    // Lưu danh sách xuống file

    public void luuFile(String tenFile) {

        try (FileOutputStream fos = new FileOutputStream(tenFile);

             ObjectOutputStream oos = new ObjectOutputStream(fos)) {

            oos.writeObject(this.danhSach);

            System.out.println("Luu file thanh cong!");

        } catch (IOException e) {

            System.out.println("Loi khi luu file: " + e.getMessage());

            e.printStackTrace();

        }

    }

    // Đọc danh sách từ file

    public void docFile(String tenFile) {

        try (FileInputStream fis = new FileInputStream(tenFile);

             ObjectInputStream ois = new ObjectInputStream(fis)) {

            this.danhSach = (ArrayList<NhanVien>) ois.readObject();

            System.out.println("Doc file thanh cong!");

        } catch (FileNotFoundException e) {

            System.out.println("Khong tim thay file de doc! Tao danh sach moi.");

            this.danhSach = new ArrayList<>();

        } catch (IOException | ClassNotFoundException e) {

            System.out.println("Loi khi doc file: " + e.getMessage());

            e.printStackTrace();

        }

    }

}

**(2) Dán KẾT QUẢ CONSOLE (Console Output) khi chạy Main.java vào đây:**

--- THEM NHAN VIEN ---

--- LUU FILE ---

Luu file thanh cong!

--- TAO DOI TUONG QUAN LY MOI (RONG) ---

Danh sach cua qlMoi truoc khi doc file:

Danh sach rong!

--- DOC FILE VAO qlMoi ---

Doc file thanh cong!

--- DANH SACH CUA qlMoi SAU KHI DOC FILE ---

Ten: Binh, Luong: 17000000.0

Ten: Chi, Luong: 13500000.0

**Phần 3: Câu hỏi/Thắc mắc (Bắt buộc)**

Sau khi hoàn thành bài thực hành, bạn bắt buộc phải đặt 1 câu hỏi liên quan đến nội dung bài học (Serializable, ObjectStream, try-with-resources, ClassNotFoundException...).

Câu hỏi:

**Câu hỏi của tôi là:**

[ Nếu em thêm một thuộc tính mới (ví dụ String diaChi) vào lớp NhanVien sau khi đã lưu file, thì khi đọc file cũ lên chương trình có bị lỗi không? Và tại sao serialVersionUID lại giúp tránh lỗi đó? ]

**Phần 4: 💡 Mở rộng (Không bắt buộc)**

**"Cạm bẫy" của Serializable: Từ khóa transient**

Khi bạn implements Serializable, Java sẽ cố gắng lưu **tất cả** thuộc tính của đối tượng. Nhưng nếu có một thuộc tính "nhạy cảm" (như password) hoặc một thuộc tính không cần lưu (như scanner), bạn có thể "ra lệnh" cho Java bỏ qua nó bằng từ khóa transient.

public class NguoiDung implements Serializable {

private String username;

// 'transient' se bao Java KHONG LUU truong nay xuong file

private transient String password;

// 'static' cung se khong bi luu

private static int soNguoiDung = 0;

// ... constructors, getters, setters ...

}

Khi bạn đọc (deserialize) một đối tượng NguoiDung từ file lên, giá trị của trường password sẽ là null.

1. Ý nghĩa của từ khóa transient

- Khi một lớp implements Serializable, Java sẽ **tự động lưu tất cả các thuộc tính** của đối tượng xuống file.

- Nhưng nếu có một số thuộc tính **nhạy cảm** (như password, token, thông tin cá nhân) hoặc **không cần lưu** (như Scanner, Thread, Socket, v.v.), ta có thể **ra lệnh cho Java bỏ qua chúng** bằng cách thêm từ khóa transient.

2. Ví dụ minh họa

import java.io.\*;

public class NguoiDung implements Serializable {

private String username;

private transient String password; // Sẽ KHÔNG được lưu

private static int soNguoiDung = 0; // Cũng không được lưu

public NguoiDung(String username, String password) {

this.username = username;

this.password = password;

soNguoiDung++;

}

@Override

public String toString() {

return "User: " + username + ", Password: " + password;

}

}

3. Khi chạy chương trình test:

import java.io.\*;

public class Main {

public static void main(String[] args) {

NguoiDung nd = new NguoiDung("admin", "12345");

// Lưu đối tượng xuống file

try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("user.dat"))) {

oos.writeObject(nd);

} catch (IOException e) {

e.printStackTrace();

}

// Đọc đối tượng từ file lên

try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream("user.dat"))) {

NguoiDung nd2 = (NguoiDung) ois.readObject();

System.out.println("Sau khi đọc từ file:");

System.out.println(nd2);

} catch (IOException | ClassNotFoundException e) {

e.printStackTrace();

}

}

}

3. Kết quả

Sau khi đọc từ file:

User: admin, Password: null

5.Kết luận

- transient → Java **bỏ qua** thuộc tính đó khi ghi xuống file.

- Khi đọc lại, giá trị của thuộc tính **sẽ trở về mặc định** (với String là null, với int là 0, với boolean là false).

- static cũng **không được lưu** vì nó thuộc **lớp**, không thuộc **đối tượng**.