

LAB 5: LUỒNG VÀO/RA

MỤC TIÊU:

Kết thúc bài thực hành này bạn có khả năng

- ✓ Sử dụng FileInputStream/FileOutputStream để làm việc với file nhị phân
- ✓ Sử dụng ObjectInputStream/ObjectOutputStream để làm việc với luồng vào/ra đối tượng
- ✓ Sử dụng FileReader/Writer để làm việc với file văn bản
- ✓ Sử dụng BufferedReader/BufferedWriter để làm việc với luồng đệm

PHẦN I

BÀI 1 (2 ĐIỂM)

Xây dựng một thư viện tiện ích XFile gồm các hàm read() và write() cho phép đọc và ghi file nhị phân.

Sử dụng thư viện trên để sao chép một file thành một file khác.

HƯỚNG DẪN:

- ✓ Tạo lớp XFile

```
package poly.io;

import java.io.FileInputStream;
import java.io.FileOutputStream;

public class XFile {
    /**
     * Đọc file nhị phân
     * @param path là đường dẫn file cần đọc
     * @return dữ liệu đọc được
```

```

    * @throws đọc file có lỗi
    */
    public static byte[] read(String path) {
        ...
    }
    /**
    * Ghi file nhị phân
    * @param path là đường dẫn file cần ghi
    * @param data là dữ liệu cần ghi vào file
    * @throws ghi file có lỗi
    */
    public static void write(String path, byte[] data) {
        ...
    }
}

```

✓ Viết mã cho hàm read();

```

try {
    FileInputStream fis = new FileInputStream(path);
    int n = fis.available();
    byte[] data = new byte[n];
    fis.read(data);
    fis.close();
    return data;
}
catch (Exception e) {
    throw new RuntimeException(e);
}

```

✓ Viết mã cho hàm write()

```

try {
    FileOutputStream fos = new FileOutputStream(path);

```

```

        fos.write(data);
        fos.close();
    }
    catch (Exception e) {
        throw new RuntimeException(e);
    }

```

- ✓ Tạo lớp XFileDemo chứa main() và sử dụng thư viện XFile như sau

```

public static void main(String[] args) {
    byte[] data = XFile.read("c:/temp/a.gif");
    XFile.write("c:/temp/b.gif", data);
}

```

BÀI 2 (2 ĐIỂM)

Bổ sung vào thư viện XFile 2 hàm cho phép đọc ghi đối tượng từ file.

```

/**
 * Đọc file đối tượng
 * @param path là đường dẫn file cần đọc
 * @return đối tượng đọc được
 * @throws đọc file có lỗi
 */
public static Object readObject(String path) {...}

/**
 * Ghi file đối tượng
 * @param path là đường dẫn file cần ghi
 * @param object đối tượng cần ghi vào file
 * @throws đọc file có lỗi
 */
public static void writeObject(String path, Object object) {...}

```

Viết mã cho hàm readObject()

```

try {
    ObjectInputStream ois =

```

```

        new ObjectInputStream(new FileInputStream(path));
        Object object = ois.readObject();
        ois.close();
        return object;
    }
    catch (Exception e) {
        throw new RuntimeException(e);
    }
}

```

Viết mã cho hàm writeObject()

```

try {
    ObjectOutputStream oos =
        new ObjectOutputStream(new FileOutputStream(path));
    oos.writeObject(object);
    oos.close();
}
catch (Exception e) {
    throw new RuntimeException(e);
}

```

Sử dụng các hàm readObject() và writeObject() để đọc và ghi List<Student>. Chú ý lớp Student phải implements interface Serializable.

```

public class Student implements Serializable{
    public String name;
    public double marks;
    public String major;

    public Student(String name, double marks, String major) {
        this.name = name;
        this.marks = marks;
        this.major = major;
    }
}

```

```

public String getGrade(){
    if(this.marks < 3){
        return "Kém";
    }
    if(this.marks < 5){
        return "Yếu";
    }
    if(this.marks < 6.5){
        return "Trung bình";
    }
    if(this.marks < 7.5){
        return "Khá";
    }
    if(this.marks < 9){
        return "Giỏi";
    }
    return "Xuất sắc";
}

public boolean isBonus(){
    return this.marks >= 7.5;
}
}

```

Tạo lớp chứa phương thức main() và viết mã đọc ghi List<Student> như sau

```

List<Student> list = new ArrayList<>();
list.add(new Student("Tuấn", 5, "CNTT"));
list.add(new Student("Cường", 7.5, "TKTW"));
list.add(new Student("Hạnh", 8.5, "CNTT"));

XFile.writeObject("c:/temp/students.dat", list);

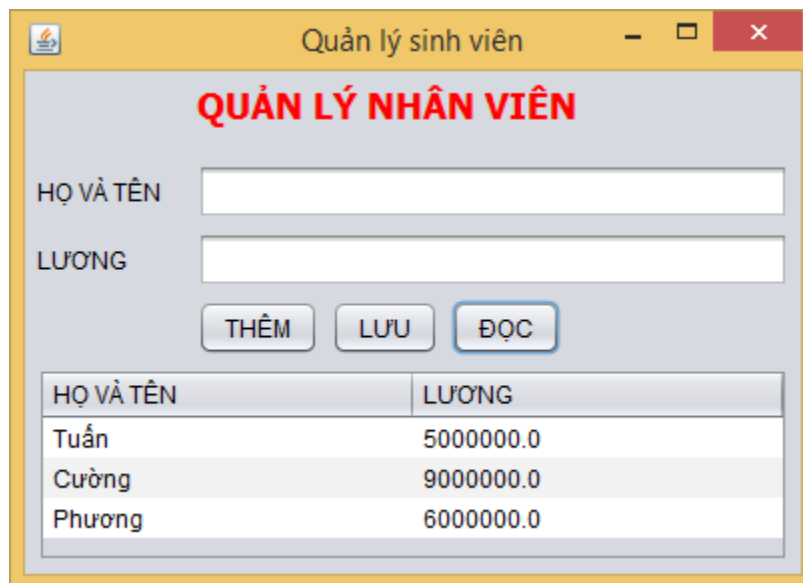
```

```
List<Student> list2 =
    (List<Student>) XFile.readObject("c:/temp/students.dat");
for(Student sv : list){
    System.out.println(">Họ và tên: " + sv.name);
}
```

PHẦN II

BÀI 3 (2 ĐIỂM)

Viết chương trình quản lý nhân viên có giao diện như sau. Sử dụng thư viên XFile để đọc ghi List<Staff>



HỌ VÀ TÊN	LƯƠNG
Tuấn	5000000.0
Cường	9000000.0
Phương	6000000.0

1. Thiết kế giao diện như trên
2. Xây dựng lớp Staff

```
public class Staff implements Serializable{
    public String fullname;
    public double salary;
}
```

3. Bổ sung mã vào JFrame

```
// Nắm giữ danh sách nhân viên nhập từ người dùng
List<Staff> list = new ArrayList<>();

// Tạo một nhân viên mới và bổ sung vào List<Staff>
private void addStaff() {...}

// Hiển thị List<Staff> lên bảng
private void fillToTable() {...}
```

4. Viết mã cho nút [THÊM]

- ✓ Xử lý sự kiện click

```
private void btnThemActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.addStaff();
    this.fillToTable();
}
```

- ✓ Mã phương thức addStaff()

```
/*
 * Tạo nhân viên với thông tin nhập từ form
 */
Staff nv = new Staff();
nv.fullname = txtHoTen.getText();
nv.salary = Double.parseDouble(txtLuong.getText());
// Bổ sung nhân viên vào List<Staff>
list.add(nv);
```

- ✓ Mã phương thức fillToTable()

```
/*
 * Lấy mô hình dữ liệu của bảng và xóa sách các hàng
 */
DefaultTableModel model =
    (DefaultTableModel) tblStaffs.getModel();
model.setRowCount(0);
/*
 * Duyệt List<Staff> và bổ sung các nhân viên vào bảng
 */
```

```
for(Staff nv : list){
    Object[] row = new Object[]{nv.fullname, nv.salary};
    model.addRow(row);
}
```

5. Viết mã cho nút [LƯU]

```
private void btnLuuActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    XFile.writeObject("c:/temp/staffs.dat", list); // lưu list vào file
}
```

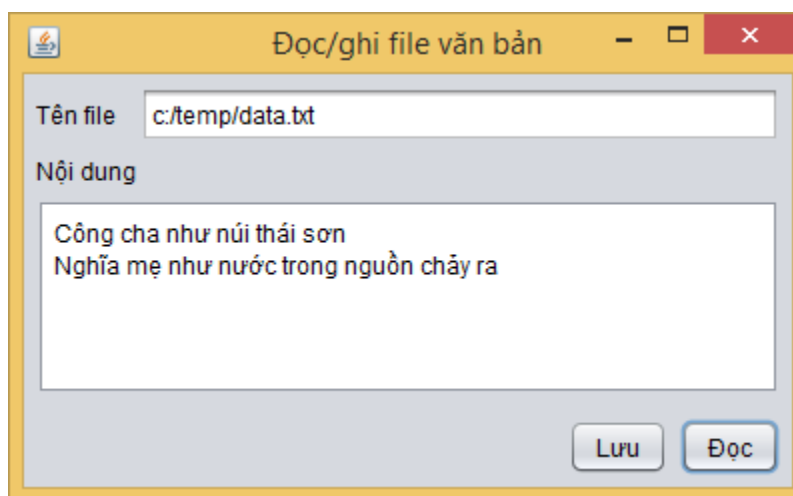
6. Viết mã cho nút [ĐỌC]

```
private void btnDocActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    list = (List<Staff>) XFile.readObject("c:/temp/staffs.dat"); // đọc list từ file
    this.fillToTable();
}
```

BÀI 4 (2 ĐIỂM)

Tạo form làm việc với file văn bản bằng cách sử dụng `BufferedReader` và `BufferWriter`.

- ✓ Thiết kế giao diện như sau



- Đặt tên các thành phần giao diện theo qui ước

✓ Viết mã cho nút [Lưu]

```
String path = txtTenFile.getText();
String noiDung = txtNoiDung.getText();
try {
    BufferedWriter bw = new BufferedWriter(new FileWriter(path));
    bw.write(noiDung);
    bw.newLine();
    bw.close();
    JOptionPane.showMessageDialog(this, "Ghi file thành công!");
}
catch (Exception e) {
    JOptionPane.showMessageDialog(this, "Lỗi ghi file");
}
```

✓ Viết mã cho nút [Đọc]

```
String path = txtTenFile.getText();
try {
    BufferedReader br = new BufferedReader(new FileReader(path));
    while(true){
        String line = br.readLine();
        if(line == null){
            break;
        }
        else{
            txtNoiDung.append(line);
            txtNoiDung.append("\r\n");
        }
    }
    br.close();
    JOptionPane.showMessageDialog(this, "Đọc file thành công!");
}
catch (Exception e) {
    JOptionPane.showMessageDialog(this, "Lỗi đọc file");
}
```

BÀI 5 (2 ĐIỂM)

Giảng viên cho thêm