

BÀI TẬP THỰC HÀNH JAVA NÂNG CAO

Nội dung chính

Bài tập Java

- 1. Bài tập java cơ bản
- 2. Bài tập chuỗi trong Java
- 3. Bài tập mảng trong Java
- 4. Bài tập về các thuật toán sắp xếp trong Java
- **5. Bài tập java nâng cao**
 - Bài tập quản lý sinh viên trong Java – console HĐT
 - Bài tập quản lý sinh viên trong Java - Swing

Bài tập Java

- Bài này cung cấp cho bạn danh sách các dạng bài tập nâng cao khác nhau để bạn thực hành khi học java.



CHƯƠNG 3: Bài Tập Java Nâng Cao

Bài tập Java Swing

Bài tập quản lý sinh viên trong java

Nội dung chính

- [Bài tập quản lý sinh viên trong Java Swing](#)
- [Lời giải](#)
- [Tạo project qlsv-swing](#)
- [I. Tạo chức năng login](#)
- [1. Tạo lớp User.java](#)
- [2. Tạo lớp UserDao.java](#)
- [3. Tạo lớp LoginView.java](#)
- [4. Tạo lớp LoginController.java](#)
- [II. Tạo chức năng quản lý sinh viên](#)
- [1. Tạo lớp Student.java](#)
- [2. Tạo lớp StudentXML.java](#)
- [3. Tạo lớp StudentDao.java](#)
- [4. Tạo lớp StudentView.java](#)
- [5. Tạo lớp StudentController.java](#)
- [6. Tạo lớp FileUtils.java](#)
- [III. Tạo lớp App.java](#)
- [Run bài tập quản lý sinh viên trong java swing](#)
- [1. Login](#)
- [2. Thêm sinh viên](#)

Bài tập quản lý sinh viên trong Java Swing

Đề bài: Viết chương trình **quản lý sinh viên trong Java** , sử dụng Swing để tạo giao diện và áp dụng mô hình MVC. Mỗi đối tượng sinh viên có các thuộc tính sau: id, name, age, address và gpa (điểm trung bình). Với các chức năng sau:

1. Sử dụng **mô hình MVC**.
2. Tạo màn hình đăng nhập.
3. Add student.
4. Edit student.
5. Delete student.
6. Sắp xếp student theo GPA.
7. Sắp xếp student theo Name.

8. Hiển thị danh sách student.

9. Lưu danh sách sinh viên vào file "student.xml".

Trong đó sinh viên được lưu vào file "**student.xml**" với định dạng xml. Ví dụ:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<students>
  <student>
    <id>1</id>
    <name>Mai Thanh Toan</name>
    <age>22</age>
    <address>Ha Noi</address>
    <gpa>8.0</gpa>
  </student>
  <student>
    <id>2</id>
    <name>Vinh The Mac</name>
    <age>23</age>
    <address>Vinh Phuc</address>
    <gpa>9.5</gpa>
  </student>
</students>
```

Lời giải

Chúng ta sẽ áp dụng mô hình MVC và Java Swing để tạo chương trình quản lý sinh viên.

Sử dụng maven để quản lý project, tham khảo bài [tạo Maven project trong Eclipse](#) .

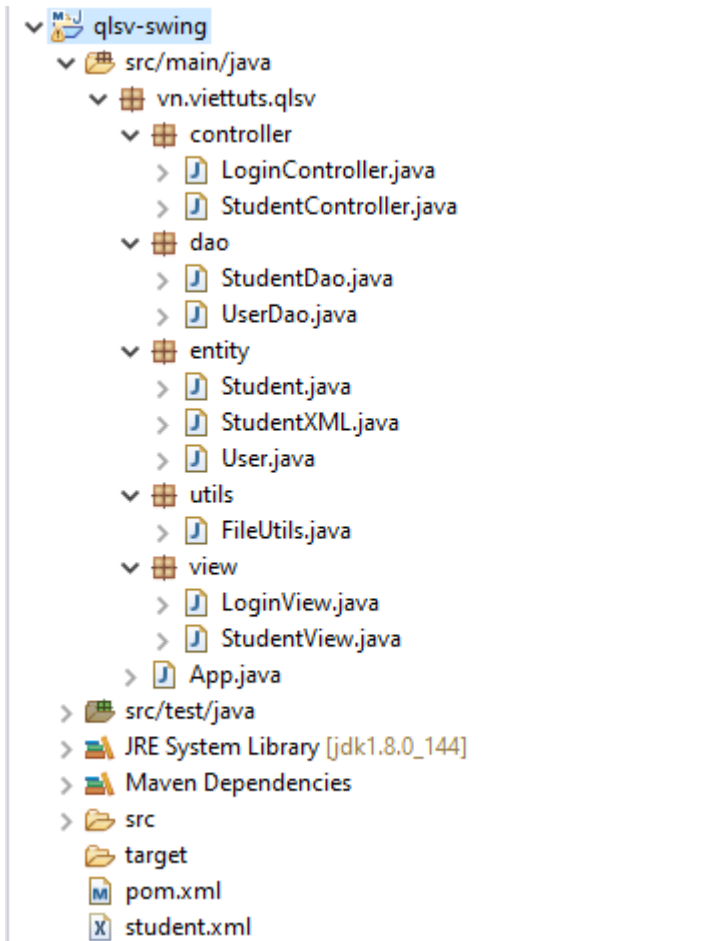
MVC (viết tắt của Model-View-Controller) là một mẫu kiến trúc phần mềm hay mô hình thiết kế để tạo lập giao diện người dùng trên máy tính.

Đa số các dự án trong thực tế sử dụng mô hình MVC.

Chú ý: trong bài này mình đã comment rất nhiều trong code. Các bạn có chỗ nào không hiểu hoặc có câu hỏi gì thì để lại ở phần comment nhé.

Tạo project Quản lý sinh viên: qlsv-swing

Tạo maven project với cấu trúc của project trên **eclipse**:



Tầng M (model) bao gồm package ***vn.viettuts.qlsv.dao*** và ***vn.viettuts.qlsv.entity***

- Lớp **User.java** để lưu thông tin người dùng.
- Lớp **UserDao.java** chứa phương thức `checkUser()` để kiểm tra thông tin đăng nhập.
- Lớp **Student.java** để lưu thông tin cho mỗi sinh viên.
- Lớp **StudentXML.java** để lưu thông tin danh sách sinh viên với định dạng XML vào file `student.xml`.
- Lớp **StudentDao.java** chứa các phương thức quản lý sinh viên như thêm, sửa, xóa, sắp xếp, đọc, ghi sinh viên.

Tầng V (view) bao gồm package ***vn.viettuts.qlsv.view***

- Lớp **LoginView.java** tạo màn hình login.
- Lớp **StudentView.java** tạo màn hình quản lý sinh viên.

Tầng C (controller) bao gồm package ***vn.viettuts.qlsv.controller***

- Lớp **LoginController.java** xử lý các sự kiện từ `LoginView.java`.

- Lớp **StudentController.java** xử lý các sự kiện từ StudentView.java.

Các file khác:

- Lớp **FileUtils.java** được sử dụng để đọc ghi file.
- Lớp **App.java** chứa hàm main để khởi chạy ứng dụng.
- File **student.xml** được sử dụng để lưu danh sách sinh viên.

Thêm các thư viện sau vào file pom.xml

- **jaxb-api-2.3.0.jar** : chuyển đổi đối tượng thành xml và lưu vào file, đọc file và chuyển xml thành đối tượng.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>vn.viettuts</groupId>
  <artifactId>qlsv-swing</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>qlsv</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <dependency>
      <groupId>javax.xml.bind</groupId>
      <artifactId>jaxb-api</artifactId>
      <version>2.3.0</version>
    </dependency>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

I. Tạo chức năng login

Tạo màn hình login chứa thông tin sau:

- Trường username.
 - Trường password.
 - Login button.
-

1. Tạo lớp User.java

File: User.java

```
package vn.vietttuts.qlsv.entity;

public class User {
    private String userName;
    private String password;

    public User() {
    }

    public User(String userName, String password) {
        super();
        this.userName = userName;
        this.password = password;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

2. Tạo lớp UserDao.java

File: UserDao.java

```
package vn.vietttuts.qlsv.dao;
import vn.vietttuts.qlsv.entity.User;
public class UserDao {
    public boolean checkUser(User user) {
        if (user != null) {
            if ("admin".equals(user.getUserName())
                && "admin".equals(user.getPassword())) {
                return true;
            }
        }
        return false;
    }
}
```

3. Tạo lớp LoginView.java

File: LoginView.java

```
package vn.viettuts.qlsv.view;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
import javax.swing.SpringLayout;
import javax.swing.WindowConstants;

import vn.viettuts.qlsv.entity.User;

public class LoginView extends JFrame implements ActionListener {
    private static final long serialVersionUID = 1L;
    private JLabel userNameLabel;
    private JLabel passwordlabel;
    private JPasswordField passwordField;
    private JTextField userNameField;
    private JButton loginBtn;

    public LoginView() {
        initComponents();
    }

    private void initComponents() {
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        userNameLabel = new JLabel("UserName");
        passwordlabel = new JLabel("Password");
        userNameField = new JTextField(15);
        passwordField = new JPasswordField(15);
        loginBtn = new JButton();

        loginBtn.setText("Login");
        loginBtn.addActionListener(this);

        // tạo spring layout
        SpringLayout layout = new SpringLayout();
        JPanel panel = new JPanel();
        // tạo đối tượng panel để chứa các thành phần của màn hình login
        panel.setSize(400, 300);
        panel.setLayout(layout);
        panel.add(userNameLabel);
        panel.add(passwordlabel);
        panel.add(userNameField);
        panel.add(passwordField);
        panel.add(loginBtn);
    }
}
```

```
// cài đặt vị trí các thành phần trên màn hình login
layout.putConstraint(SpringLayout.WEST, userNameLabel, 20,
SpringLayout.WEST, panel);
layout.putConstraint(SpringLayout.NORTH, userNameLabel, 80,
SpringLayout.NORTH, panel);
layout.putConstraint(SpringLayout.WEST, passwordlabel, 20,
SpringLayout.WEST, panel);
layout.putConstraint(SpringLayout.NORTH, passwordlabel, 105,
SpringLayout.NORTH, panel);
layout.putConstraint(SpringLayout.WEST, userNameField, 80,
SpringLayout.WEST, userNameLabel);
layout.putConstraint(SpringLayout.NORTH, userNameField, 80,
SpringLayout.NORTH, panel);
layout.putConstraint(SpringLayout.WEST, passwordField, 80,
SpringLayout.WEST, passwordlabel);
layout.putConstraint(SpringLayout.NORTH, passwordField, 105,
SpringLayout.NORTH, panel);
layout.putConstraint(SpringLayout.WEST, loginBtn, 80,
SpringLayout.WEST, passwordlabel);
layout.putConstraint(SpringLayout.NORTH, loginBtn, 130,
SpringLayout.NORTH, panel);

// add panel tới JFrame
this.add(panel);
this.pack();
// cài đặt các thuộc tính cho JFrame
this.setTitle("Login");
this.setSize(400, 300);
this.setResizable(false);
}

public void showMessage(String message) {
    JOptionPane.showMessageDialog(this, message);
}

public User getUser() {
    return new User(userNameField.getText(),
        String.valueOf(passwordField.getPassword()));
}

public void actionPerformed(ActionEvent e) {
}

public void addLoginListener(ActionListener listener) {
    loginBtn.addActionListener(listener);
}
}
```

4. Tạo lớp LoginController.java

File: LoginController.java

```
package vn.viettuts.qlsv.controller;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```



```
import vn.viettuts.qlsv.dao.UserDao;
import vn.viettuts.qlsv.entity.User;
import vn.viettuts.qlsv.view.LoginView;
import vn.viettuts.qlsv.view.StudentView;

public class LoginController {
    private UserDao userDao;
    private LoginView loginView;
    private StudentView studentView;

    public LoginController(LoginView view) {
        this.loginView = view;
        this.userDao = new UserDao();
        view.addLoginListener(new LoginListener());
    }

    public void showLoginView() {
        loginView.setVisible(true);
    }

    /**
     * Lớp LoginListener
     * chứa cài đặt cho sự kiện click button "Login"
     *
     * @author plpsoft.vn
     */
    class LoginListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            User user = loginView.getUser();
            if (userDao.checkUser(user)) {
                // nếu đăng nhập thành công, mở màn hình quản lý sinh viên
                studentView = new StudentView();
                StudentController studentController = new
StudentController(studentView);
                studentController.showStudentView();
                loginView.setVisible(false);
            } else {
                loginView.showMessageDialog("username hoặc password không đúng.");
            }
        }
    }
}
```

II. Tạo chức năng quản lý sinh viên

Tạo màn hình quản lý sinh viên chứa các thông tin sau:

- Các trường tương ứng với các thuộc tính của sinh viên.
- Button Add.
- Button Edit.
- Button Delete.
- Button Clear.
- Bảng hiển thị danh sách sinh viên.
- Button "Sort By Name"

- Button "Sort By GPA"

1. Tạo lớp Student.java

Lớp này để lưu thông tin cho mỗi sinh viên.

File: Student.java

```
package vn.vietttuts.qlsv.entity;

import java.io.Serializable;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name = "student")
@XmlAccessorType(XmlAccessType.FIELD)
public class Student implements Serializable {
    private static final long serialVersionUID = 1L;
    private int id;
    private String name;
    private byte age;
    private String address;
    /* điểm trung bình của sinh viên */
    private float gpa;

    public Student() {
    }

    public Student(int id, String name, byte age, String address, float gpa) {
        super();
        this.id = id;
        this.name = name;
        this.age = age;
        this.address = address;
        this.gpa = gpa;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public byte getAge() {
        return age;
    }
```

```

    }

    public void setAge(byte age) {
        this.age = age;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public float getGpa() {
        return gpa;
    }

    public void setGpa(float gpa) {
        this.gpa = gpa;
    }
}

```

2. Tạo lớp StudentXML.java

File: StudentXML.java

```

package vn.viettuts.qlsv.entity;

import java.util.List;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name = "students")
@XmlAccessorType(XmlAccessType.FIELD)
public class StudentXML {

    private List<Student> student;

    public List<Student> getStudent() {
        return student;
    }

    public void setStudent(List<Student> student) {
        this.student = student;
    }
}

```

3. Tạo lớp StudentDao.java

Tạo file "student.xml" tại thư mục gốc của dự án để lưu danh sách sinh viên.

File: StudentDao.java

```
package vn.viettuts.qlsv.dao;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

import vn.viettuts.qlsv.entity.Student;
import vn.viettuts.qlsv.entity.StudentXML;
import vn.viettuts.qlsv.utils.FileUtils;

/**
 * StudentDao class
 *
 * @author plpsoft.vn
 */
public class StudentDao {
    private static final String STUDENT_FILE_NAME = "student.xml";
    private List<Student> listStudents;

    public StudentDao() {
        this.listStudents = readListStudents();
    }

    /**
     * Lưu các đối tượng student vào file student.xml
     *
     * @param students
     */
    public void writeListStudents(List<Student> students) {
        StudentXML studentXML = new StudentXML();
        studentXML.setStudent(students);
        FileUtils.writeXMLToFile(STUDENT_FILE_NAME, studentXML);
    }

    /**
     * Đọc các đối tượng student từ file student.xml
     *
     * @return list student
     */
    public List<Student> readListStudents() {
        List<Student> list = new ArrayList<Student>();
        StudentXML studentXML = (StudentXML) FileUtils.readXMLFile(
            STUDENT_FILE_NAME, StudentXML.class);
        if (studentXML != null) {
            list = studentXML.getStudent();
        }
        return list;
    }

    /**
     * thêm student vào listStudents và lưu listStudents vào file
     */
}
```

```

    * @param student
    */
    public void add(Student student) {
        int id = (listStudents.size() > 0) ? (listStudents.size() + 1) : 1;
        student.setId(id);
        listStudents.add(student);
        writeListStudents(listStudents);
    }

    /**
     * cập nhật student vào listStudents và lưu listStudents vào file
     *
     * @param student
     */
    public void edit(Student student) {
        int size = listStudents.size();
        for (int i = 0; i < size; i++) {
            if (listStudents.get(i).getId() == student.getId()) {
                listStudents.get(i).setName(student.getName());
                listStudents.get(i).setAge(student.getAge());
                listStudents.get(i).setAddress(student.getAddress());
                listStudents.get(i).setGpa(student.getGpa());
                writeListStudents(listStudents);
                break;
            }
        }
    }

    /**
     * xóa student từ listStudents và lưu listStudents vào file
     *
     * @param student
     */
    public boolean delete(Student student) {
        boolean isFound = false;
        int size = listStudents.size();
        for (int i = 0; i < size; i++) {
            if (listStudents.get(i).getId() == student.getId()) {
                student = listStudents.get(i);
                isFound = true;
                break;
            }
        }
        if (isFound) {
            listStudents.remove(student);
            writeListStudents(listStudents);
            return true;
        }
        return false;
    }

    /**
     * sắp xếp danh sách student theo name theo tứ tự tăng dần
     */
    public void sortStudentByName() {
        Collections.sort(listStudents, new Comparator<Student>() {
            public int compare(Student student1, Student student2) {
                return student1.getName().compareTo(student2.getName());
            }
        });
    }
}

```

```

/**
 * sắp xếp danh sách student theo GPA theo tứ tự tăng dần
 */
public void sortStudentByGPA() {
    Collections.sort(listStudents, new Comparator<Student>() {
        public int compare(Student student1, Student student2) {
            if (student1.getGpa() > student2.getGpa()) {
                return 1;
            }
            return -1;
        }
    });
}

public List<Student> getListStudents() {
    return listStudents;
}

public void setListStudents(List<Student> listStudents) {
    this.listStudents = listStudents;
}
}

```

4. Tạo lớp StudentView.java

File: StudentView.java

```

package vn.viettuts.qlsv.view;

import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.SpringLayout;
import javax.swing.WindowConstants;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
import javax.swing.table.DefaultTableModel;

import vn.viettuts.qlsv.entity.Student;

public class StudentView extends JFrame implements ActionListener,
ListSelectionListener {

```

```

private static final long serialVersionUID = 1L;
private JButton addStudentBtn;
private JButton editStudentBtn;
private JButton deleteStudentBtn;
private JButton clearBtn;
private JButton sortStudentGPABtn;
private JButton sortStudentNameBtn;
private JScrollPane jScrollPaneStudentTable;
private JScrollPane jScrollPaneAddress;
private JTable studentTable;

private JLabel idLabel;
private JLabel nameLabel;
private JLabel ageLabel;
private JLabel addressLabel;
private JLabel gpaLabel;

private JTextField idField;
private JTextField nameField;
private JTextField ageField;
private JTextArea addressTA;
private JTextField gpaField;

// định nghĩa các cột của bảng student
private String [] columnNames = new String [] {
    "ID", "Name", "Age", "Address", "GPA"};
// định nghĩa dữ liệu mặc định của bảng student là rỗng
private Object data = new Object [][] {};

public StudentView() {
    initComponents();
}

private void initComponents() {
    setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    // khởi tạo các phím chức năng
    addStudentBtn = new JButton("Add");
    editStudentBtn = new JButton("Edit");
    deleteStudentBtn = new JButton("Delete");
    clearBtn = new JButton("Clear");
    sortStudentGPABtn = new JButton("Sort By GPA");
    sortStudentNameBtn = new JButton("Sort By Name");
    // khởi tạo bảng student
    jScrollPaneStudentTable = new JScrollPane();
    studentTable = new JTable();

    // khởi tạo các label
    idLabel = new JLabel("Id");
    nameLabel = new JLabel("Name");
    ageLabel = new JLabel("Age");
    addressLabel = new JLabel("Address");
    gpaLabel = new JLabel("GPA");

    // khởi tạo các trường nhập dữ liệu cho student
    idField = new JTextField(6);
    idField.setEditable(false);
    nameField = new JTextField(15);
    ageField = new JTextField(6);
    addressTA = new JTextArea();
    addressTA.setColumns(15);

```

```

        addressTA.setRows(5);
        jScrollPaneAddress = new JScrollPane();
        jScrollPaneAddress.setViewportViewView(addressTA);
        gpaField = new JTextField(6);

        // cài đặt các cột và data cho bảng student
        studentTable.setModel(new DefaultTableModel((Object[][]) data,
columnNames));
        jScrollPaneStudentTable.setViewportViewView(studentTable);
        jScrollPaneStudentTable.setPreferredSize(new Dimension (480, 300));

        // tạo spring layout
        SpringLayout layout = new SpringLayout();
        // tạo đối tượng panel để chứa các thành phần của màn hình quản lý
Student
        JPanel panel = new JPanel();
        panel.setSize(800, 420);
        panel.setLayout(layout);
        panel.add(jScrollPaneStudentTable);

        panel.add(addStudentBtn);
        panel.add(editStudentBtn);
        panel.add(deleteStudentBtn);
        panel.add(clearBtn);
        panel.add(sortStudentGPABtn);
        panel.add(sortStudentNameBtn);

        panel.add(idLabel);
        panel.add(nameLabel);
        panel.add(ageLabel);
        panel.add(addressLabel);
        panel.add(gpaLabel);

        panel.add(idField);
        panel.add(nameField);
        panel.add(ageField);
        panel.add(jScrollPaneAddress);
        panel.add(gpaField);

        // cài đặt vị trí các thành phần trên màn hình login
        layout.putConstraint(SpringLayout.WEST, idLabel, 10,
SpringLayout.WEST, panel);
        layout.putConstraint(SpringLayout.NORTH, idLabel, 10,
SpringLayout.NORTH, panel);
        layout.putConstraint(SpringLayout.WEST, nameLabel, 10,
SpringLayout.WEST, panel);
        layout.putConstraint(SpringLayout.NORTH, nameLabel, 40,
SpringLayout.NORTH, panel);
        layout.putConstraint(SpringLayout.WEST, ageLabel, 10,
SpringLayout.WEST, panel);
        layout.putConstraint(SpringLayout.NORTH, ageLabel, 70,
SpringLayout.NORTH, panel);
        layout.putConstraint(SpringLayout.WEST, addressLabel, 10,
SpringLayout.WEST, panel);
        layout.putConstraint(SpringLayout.NORTH, addressLabel, 100,
SpringLayout.NORTH, panel);
        layout.putConstraint(SpringLayout.WEST, gpaLabel, 10,
SpringLayout.WEST, panel);
        layout.putConstraint(SpringLayout.NORTH, gpaLabel, 200,
SpringLayout.NORTH, panel);
    
```



```

        layout.putConstraint(SpringLayout.WEST, idField, 100,
SpringLayout.WEST, panel);
        layout.putConstraint(SpringLayout.NORTH, idField, 10,
SpringLayout.NORTH, panel);
        layout.putConstraint(SpringLayout.WEST, nameField, 100,
SpringLayout.WEST, panel);
        layout.putConstraint(SpringLayout.NORTH, nameField, 40,
SpringLayout.NORTH, panel);
        layout.putConstraint(SpringLayout.WEST, ageField, 100,
SpringLayout.WEST, panel);
        layout.putConstraint(SpringLayout.NORTH, ageField, 70,
SpringLayout.NORTH, panel);
        layout.putConstraint(SpringLayout.WEST, jScrollPaneAddress, 100,
SpringLayout.WEST, panel);
        layout.putConstraint(SpringLayout.NORTH, jScrollPaneAddress, 100,
SpringLayout.NORTH, panel);
        layout.putConstraint(SpringLayout.WEST, gpaField, 100,
SpringLayout.WEST, panel);
        layout.putConstraint(SpringLayout.NORTH, gpaField, 200,
SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, jScrollPaneStudentTable,
300, SpringLayout.WEST, panel);
        layout.putConstraint(SpringLayout.NORTH, jScrollPaneStudentTable,
10, SpringLayout.NORTH, panel);

        layout.putConstraint(SpringLayout.WEST, addStudentBtn, 20,
SpringLayout.WEST, panel);
        layout.putConstraint(SpringLayout.NORTH, addStudentBtn, 240,
SpringLayout.NORTH, panel);
        layout.putConstraint(SpringLayout.WEST, editStudentBtn, 60,
SpringLayout.WEST, addStudentBtn);
        layout.putConstraint(SpringLayout.NORTH, editStudentBtn, 240,
SpringLayout.NORTH, panel);
        layout.putConstraint(SpringLayout.WEST, deleteStudentBtn, 60,
SpringLayout.WEST, editStudentBtn);

        layout.putConstraint(SpringLayout.NORTH, clearBtn, 240,
SpringLayout.NORTH, panel);
        layout.putConstraint(SpringLayout.WEST, clearBtn, 80,
SpringLayout.WEST, deleteStudentBtn);

        layout.putConstraint(SpringLayout.NORTH, deleteStudentBtn, 240,
SpringLayout.NORTH, panel);
        layout.putConstraint(SpringLayout.WEST, sortStudentGPABtn, 300,
SpringLayout.WEST, panel);
        layout.putConstraint(SpringLayout.NORTH, sortStudentGPABtn, 330,
SpringLayout.NORTH, panel);
        layout.putConstraint(SpringLayout.WEST, sortStudentNameBtn, 115,
SpringLayout.WEST, sortStudentGPABtn);
        layout.putConstraint(SpringLayout.NORTH, sortStudentNameBtn, 330,
SpringLayout.NORTH, panel);

        this.add(panel);
        this.pack();
        this.setTitle("Student Information");
        this.setSize(800, 420);
        // disable Edit and Delete buttons
        editStudentBtn.setEnabled(false);
    
```

```

        deleteStudentBtn.setEnabled(false);
        // enable Add button
        addStudentBtn.setEnabled(true);
    }

    public void showMessage(String message) {
        JOptionPane.showMessageDialog(this, message);
    }

    /**
     * hiển thị list student vào bảng studentTable
     *
     * @param list
     */
    public void showListStudents(List<Student> list) {
        int size = list.size();
        // với bảng studentTable có 5 cột,
        // khởi tạo mảng 2 chiều students, trong đó:
        // số hàng: là kích thước của list student
        // số cột: là 5
        Object [][] students = new Object[size][5];
        for (int i = 0; i < size; i++) {
            students[i][0] = list.get(i).getId();
            students[i][1] = list.get(i).getName();
            students[i][2] = list.get(i).getAge();
            students[i][3] = list.get(i).getAddress();
            students[i][4] = list.get(i).getGpa();
        }
        studentTable.setModel(new DefaultTableModel(students, columnNames));
    }

    /**
     * điền thông tin của hàng được chọn từ bảng student
     * vào các trường tương ứng của student.
     */
    public void fillStudentFromSelectedRow() {
        // lấy chỉ số của hàng được chọn
        int row = studentTable.getSelectedRow();
        if (row >= 0) {
            idField.setText(studentTable.getModel().getValueAt(row,
0).toString());
            nameField.setText(studentTable.getModel().getValueAt(row,
1).toString());
            ageField.setText(studentTable.getModel().getValueAt(row,
2).toString());
            addressTA.setText(studentTable.getModel().getValueAt(row,
3).toString());
            gpaField.setText(studentTable.getModel().getValueAt(row,
4).toString());
            // enable Edit and Delete buttons
            editStudentBtn.setEnabled(true);
            deleteStudentBtn.setEnabled(true);
            // disable Add button
            addStudentBtn.setEnabled(false);
        }
    }

    /**
     * xóa thông tin student
     */

```

```

public void clearStudentInfo() {
    idField.setText("");
    nameField.setText("");
    ageField.setText("");
    addressTA.setText("");
    gpaField.setText("");
    // disable Edit and Delete buttons
    editStudentBtn.setEnabled(false);
    deleteStudentBtn.setEnabled(false);
    // enable Add button
    addStudentBtn.setEnabled(true);
}

/**
 * hiện thị thông tin student
 *
 * @param student
 */
public void showStudent(Student student) {
    idField.setText("" + student.getId());
    nameField.setText(student.getName());
    ageField.setText("" + student.getAge());
    addressTA.setText(student.getAddress());
    gpaField.setText("" + student.getGpa());
    // enable Edit and Delete buttons
    editStudentBtn.setEnabled(true);
    deleteStudentBtn.setEnabled(true);
    // disable Add button
    addStudentBtn.setEnabled(false);
}

/**
 * lấy thông tin student
 *
 * @return
 */
public Student getStudentInfo() {
    // validate student
    if (!validateName() || !validateAge() || !validateAddress() ||
!validateGPA()) {
        return null;
    }
    try {
        Student student = new Student();
        if (idField.getText() != null && !"".equals(idField.getText())) {
            student.setId(Integer.parseInt(idField.getText()));
        }
        student.setName(nameField.getText().trim());
        student.setAge(Byte.parseByte(ageField.getText().trim()));
        student.setAddress(addressTA.getText().trim());
        student.setGpa(Float.parseFloat(gpaField.getText().trim()));
        return student;
    } catch (Exception e) {
        showMessage(e.getMessage());
    }
    return null;
}

private boolean validateName() {
    String name = nameField.getText();
    if (name == null || "".equals(name.trim())) {

```

```

        nameField.requestFocus();
        showMessage("Name không được trống.");
        return false;
    }
    return true;
}

private boolean validateAddress() {
    String address = addressTA.getText();
    if (address == null || "".equals(address.trim())) {
        addressTA.requestFocus();
        showMessage("Address không được trống.");
        return false;
    }
    return true;
}

private boolean validateAge() {
    try {
        Byte age = Byte.parseByte(ageField.getText().trim());
        if (age < 0 || age > 100) {
            ageField.requestFocus();
            showMessage("Age không hợp lệ, age nên trong khoảng 0 đến 100.");
            return false;
        }
    } catch (Exception e) {
        ageField.requestFocus();
        showMessage("Age không hợp lệ!");
        return false;
    }
    return true;
}

private boolean validateGPA() {
    try {
        Float gpa = Float.parseFloat(gpaField.getText().trim());
        if (gpa < 0 || gpa > 10) {
            gpaField.requestFocus();
            showMessage("GPA không hợp lệ, gpa nên trong khoảng 0 đến 10.");
            return false;
        }
    } catch (Exception e) {
        gpaField.requestFocus();
        showMessage("GPA không hợp lệ!");
        return false;
    }
    return true;
}

public void actionPerformed(ActionEvent e) {
}

public void valueChanged(ListSelectionEvent e) {
}

public void addAddStudentListener(ActionListener listener) {
    addStudentBtn.addActionListener(listener);
}

public void addEdiStudentListener(ActionListener listener) {

```

```

        editStudentBtn.addActionListener(listener);
    }

    public void addDeleteStudentListener(ActionListener listener) {
        deleteStudentBtn.addActionListener(listener);
    }

    public void addClearListener(ActionListener listener) {
        clearBtn.addActionListener(listener);
    }

    public void addSortStudentGPAListener(ActionListener listener) {
        sortStudentGPABtn.addActionListener(listener);
    }

    public void addSortStudentNameListener(ActionListener listener) {
        sortStudentNameBtn.addActionListener(listener);
    }

    public void addListStudentSelectionListener(ListSelectionListener
listener) {
        studentTable.getSelectionModel().addListSelectionListener(listener);
    }
}

```

5. Tạo lớp StudentController.java

File: StudentController.java

```

package vn.viettuts.qlsv.controller;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;

import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;

import vn.viettuts.qlsv.dao.StudentDao;
import vn.viettuts.qlsv.entity.Student;
import vn.viettuts.qlsv.view.StudentView;

public class StudentController {
    private StudentDao studentDao;
    private StudentView studentView;

    public StudentController(StudentView view) {
        this.studentView = view;
        studentDao = new StudentDao();

        view.addAddStudentListener(new AddStudentListener());
        view.addEditStudentListener(new EditStudentListener());
        view.addDeleteStudentListener(new DeleteStudentListener());
    }
}

```

```

        view.addClearListener(new ClearStudentListener());
        view.addSortStudentGPAListener(new SortStudentGPAListener());
        view.addSortStudentNameListener(new SortStudentNameListener());
        view.addListStudentSelectionListener(new
ListStudentSelectionListener());
    }

    public void showStudentView() {
        List<Student> studentList = studentDao.getListStudents();
        studentView.setVisible(true);
        studentView.showListStudents(studentList);
    }

    /**
     * Lớp AddStudentListener
     * chứa cài đặt cho sự kiện click button "Add"
     *
     * @author plpsoft.vn
     */
    class AddStudentListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            Student student = studentView.getStudentInfo();
            if (student != null) {
                studentDao.add(student);
                studentView.showStudent(student);
                studentView.showListStudents(studentDao.getListStudents());
                studentView.showMessage("Thêm thành công!");
            }
        }
    }

    /**
     * Lớp EditStudentListener
     * chứa cài đặt cho sự kiện click button "Edit"
     *
     * @author plpsoft.vn
     */
    class EditStudentListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            Student student = studentView.getStudentInfo();
            if (student != null) {
                studentDao.edit(student);
                studentView.showStudent(student);
                studentView.showListStudents(studentDao.getListStudents());
                studentView.showMessage("Cập nhật thành công!");
            }
        }
    }

    /**
     * Lớp DeleteStudentListener
     * chứa cài đặt cho sự kiện click button "Delete"
     *
     * @author plpsoft.vn
     */
    class DeleteStudentListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            Student student = studentView.getStudentInfo();
            if (student != null) {
                studentDao.delete(student);
            }
        }
    }

```

```

        studentView.clearStudentInfo();
        studentView.showListStudents(studentDao.getListStudents());
        studentView.showMessage("Xóa thành công!");
    }
}

/**
 * Lớp ClearStudentListener
 * chứa cài đặt cho sự kiện click button "Clear"
 *
 * @author plpsoft.vn
 */
class ClearStudentListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        studentView.clearStudentInfo();
    }
}

/**
 * Lớp SortStudentGPAListener
 * chứa cài đặt cho sự kiện click button "Sort By GPA"
 *
 * @author plpsoft.vn
 */
class SortStudentGPAListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        studentDao.sortStudentByGPA();
        studentView.showListStudents(studentDao.getListStudents());
    }
}

/**
 * Lớp SortStudentGPAListener
 * chứa cài đặt cho sự kiện click button "Sort By Name"
 *
 * @author plpsoft.vn
 */
class SortStudentNameListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        studentDao.sortStudentByName();
        studentView.showListStudents(studentDao.getListStudents());
    }
}

/**
 * Lớp ListStudentSelectionListener
 * chứa cài đặt cho sự kiện chọn student trong bảng student
 *
 * @author plpsoft.vn
 */
class ListStudentSelectionListener implements ListSelectionListener {
    public void valueChanged(ListSelectionEvent e) {
        studentView.fillStudentFromSelectedRow();
    }
}
}

```

6. Tạo lớp FileUtils.java

File: FileUtils.java

```
package vn.viettuts.qlsv.utils;
import java.io.File;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
import javax.xml.bind.Unmarshaller;

public class FileUtils {

    /**
     * Chuyển đổi đối tượng object về định dạng XML
     * Sau đó lưu vào fileName
     *
     * @param fileName
     * @param object
     */
    public static void writeXMLToFile(String fileName, Object object) {
        try {
            // tạo đối tượng JAXB Context
            JAXBContext jaxbContext =
                JAXBContext.newInstance(object.getClass());
            // Create đối tượng Marshaller
            Marshaller jaxbMarshaller = jaxbContext.createMarshaller();
            // formatting
            jaxbMarshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
                Boolean.TRUE);
            // lưu nội dung XML vào file
            File xmlFile = new File(fileName);
            jaxbMarshaller.marshal(object, xmlFile);
        } catch (JAXBException e) {
            e.printStackTrace();
        }
    }

    /**
     * Đọc nội dung fileName, sau đó chuyển đổi nội dung của file
     * thành đối tượng có kiểu là clazz.
     *
     * @param fileName
     * @param clazz
     * @return
     */
    public static Object readXMLFile(String fileName, Class<?> clazz) {
        try {
            File xmlFile = new File(fileName);
            JAXBContext jaxbContext = JAXBContext.newInstance(clazz);
            Unmarshaller jaxbUnmarshaller =
                jaxbContext.createUnmarshaller();
            return jaxbUnmarshaller.unmarshal(xmlFile);
        } catch (JAXBException e) {
            e.printStackTrace();
        }
        return null;
    }
}
```


III. Tạo lớp App.java

File: App.java

Lớp này chứa phương thức main() để chạy ứng dụng.

```
package vn.viettuts.qlsv;

import java.awt.EventQueue;

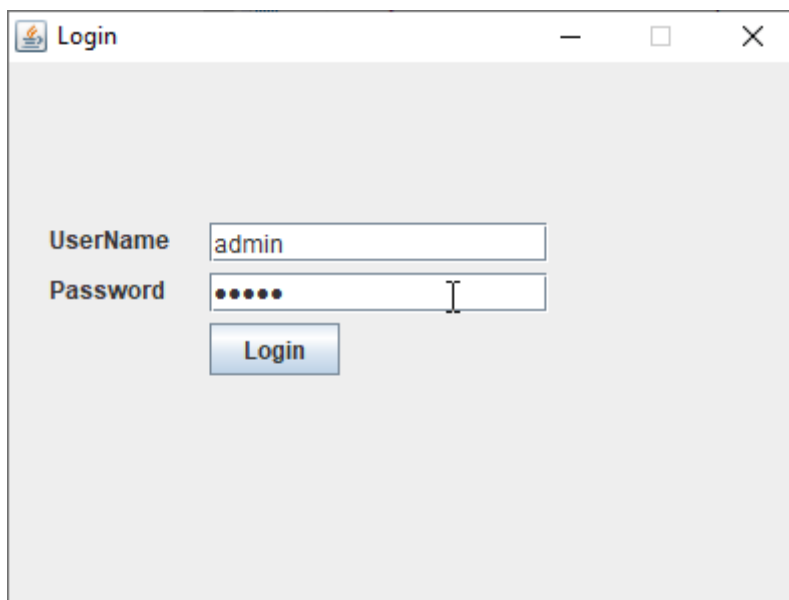
import vn.viettuts.qlsv.controller.LoginController;
import vn.viettuts.qlsv.view.LoginView;

public class App {
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                LoginView view = new LoginView();
                LoginController controller = new LoginController(view);
                // hiển thị màn hình login
                controller.showLoginView();
            }
        });
    }
}
```

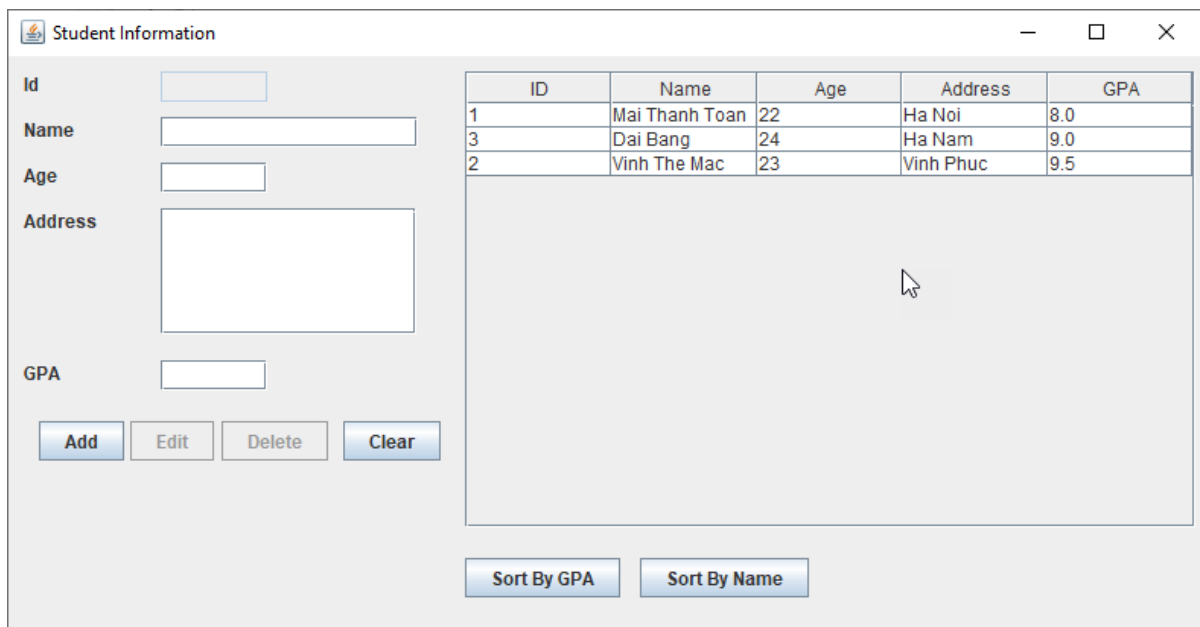
Run bài tập quản lý sinh viên trong java swing

1. Login

Login với username/pasword: **admin/admin**:



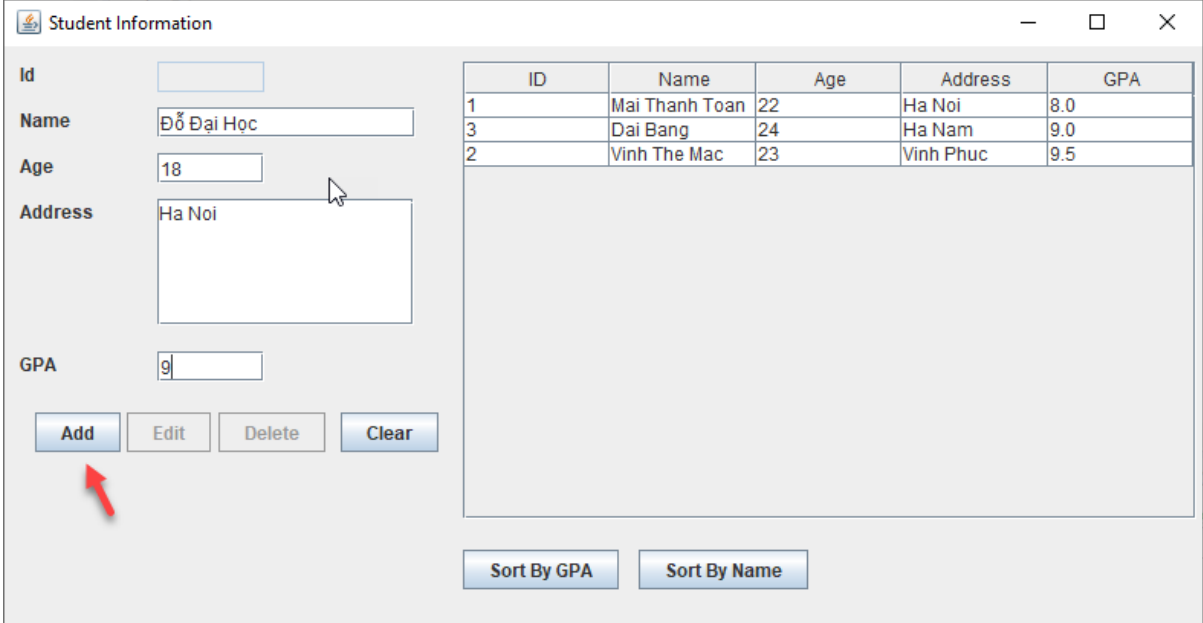
Màn hình quản lý sinh viên:



ID	Name	Age	Address	GPA
1	Mai Thanh Toan	22	Ha Noi	8.0
3	Dai Bang	24	Ha Nam	9.0
2	Vinh The Mac	23	Vinh Phuc	9.5

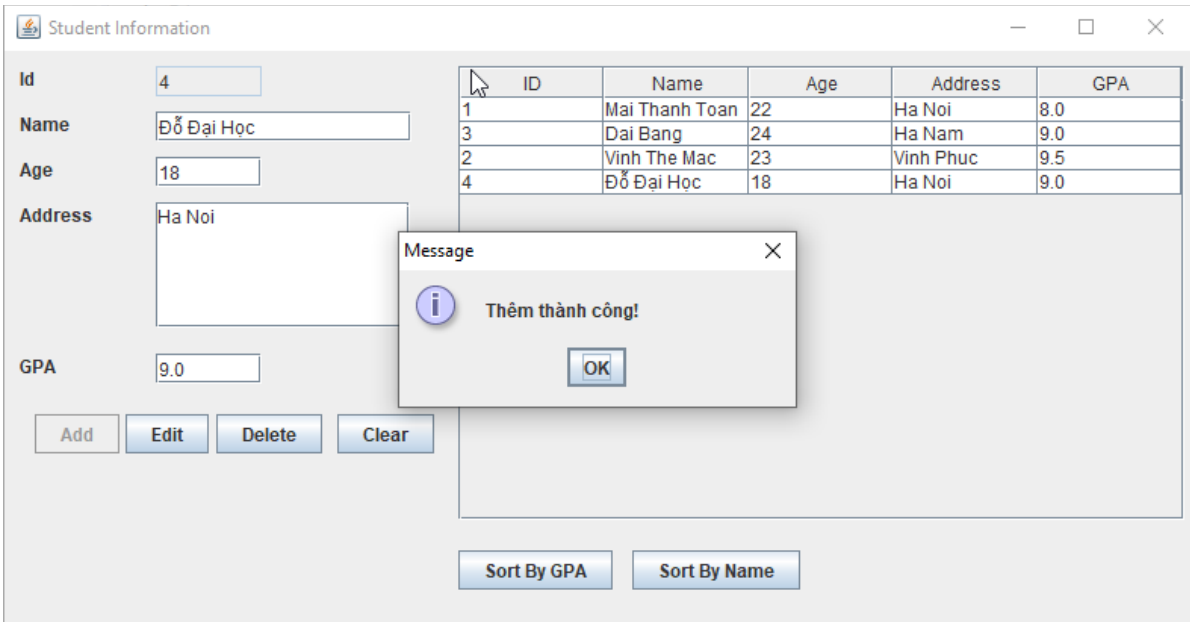
2. Thêm sinh viên

Nhập thông tin sinh viên:



ID	Name	Age	Address	GPA
1	Mai Thanh Toan	22	Ha Noi	8.0
3	Dai Bang	24	Ha Nam	9.0
2	Vinh The Mac	23	Vinh Phuc	9.5

Click Add button:



ID	Name	Age	Address	GPA
1	Mai Thanh Toan	22	Ha Noi	8.0
3	Dai Bang	24	Ha Nam	9.0
2	Vinh The Mac	23	Vinh Phuc	9.5
4	Đỗ Đại Học	18	Ha Noi	9.0

Ví dụ Java IO

Bài này cung cấp các ví dụ Java IO. Java cung cấp các lớp i/o để hỗ trợ việc input và output thông qua byte stream, character stream, buffered stream và hệ thống file. Dưới đây là danh các ví dụ về java i/o bao gồm thao tác với file, file tạm thời, thư mục, đường dẫn.

Nội dung chính

- *Thao tác với file trong java*
- *Thao tác với folder trong java*
- *Nén file trong java*

Thao tác với file trong java

[Tạo file trong java](#)

[Đọc file với BufferedInputStream trong java](#)

[Đọc file với BufferedReader trong java](#)

[Đọc ghi file trong java](#)

[Append nội dung vào file trong java](#)

[Delete file trong java](#)

[Delete nhiều file dựa vào phần mở rộng trong java](#)

[Tìm các file dựa vào phần mở rộng trong java](#)

[Đổi tên file trong java](#)

[Copy file trong java](#)

[Move file trong java](#)

[Check file tồn tại trong java](#)

[Check file ẩn trong java](#)

[Liệt kê tất cả file và folder trong một folder](#)

[Get thư mục hiện tại trong Java](#)

[Read properties file trong java](#)

Thao tác với folder trong java

[Check thư mục rỗng trong java](#)

[Tạo thư mục trong java](#)

[Xóa thư mục trong java](#)

[Copy thư mục trong java](#)

Nén file trong java

ZIP file trong java

Tạo file trong java

Phương thức **createNewFile()** của lớp File được sử dụng để tạo file trong java, phương thức này trả về giá trị *true* nếu tạo file thành công, *false* nếu tạo file thất bại.

Ví dụ về tạo file trong java:

File: CreateFileExample.java

```
import java.io.File;
import java.io.IOException;

public class CreateFileExample {
    public static void main(String[] args) {
        try {

            File file = new File("D:\\newfile.txt");

            if (file.createNewFile()) {
                System.out.println("File is created!");
            } else {
                System.out.println("File already exists.");
            }

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Đọc file trong java với BufferedInputStream

Lớp BufferedInputStream trong java được sử dụng để đọc thông tin từ stream. Trong nội bộ của lớp này sử dụng cơ chế đệm để làm cho hiệu suất đọc nhanh hơn.

Dưới đây là ví dụ đọc data từ file bằng cách sử dụng BufferedInputStream kết hợp với DataInputStream

File: BufferedInputStreamExample.java

```
import java.io.BufferedInputStream;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
```

```
public class BufferedInputStreamExample {
    public static void main(String[] args) throws IOException {
        File file = new File("D:\\testin.txt");
        FileInputStream fis = null;
        BufferedInputStream bis = null;
        DataInputStream dis = null;

        try {
            fis = new FileInputStream(file);
            bis = new BufferedInputStream(fis);
            dis = new DataInputStream(bis);

            while (dis.available() != 0) {
                System.out.println(dis.readLine());
            }

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            fis.close();
            bis.close();
            dis.close();
        }
    }
}
```

Giả sử nội dung file là:

Welcome to Java IO.

Output:

Welcome to Java IO.

Đọc file với BufferedReader trong java

Trong java, Lớp `BufferedReader` được sử dụng để đọc văn bản từ một input stream dựa trên các ký tự (character stream). Nó có thể được sử dụng để đọc dữ liệu theo dòng (line by line) bằng phương thức `readLine()`. Nó giúp hiệu suất nhanh.

Ví dụ về đọc file với `BufferedReader` trong java

```
import java.io.BufferedReader;
import java.io.FileReader;

public class BufferedReaderExample {
    public static void main(String args[]) throws Exception {
        FileReader fr = new FileReader("D:\\testout.txt");
        BufferedReader br = new BufferedReader(fr);

        int i;
        while ((i = br.read()) != -1) {
            System.out.print((char) i);
        }
    }
}
```

```
        br.close();
        fr.close();
    }
}
```

Giả sử file testout.txt có nội dung như sau:

Welcome to java.

Output:

Welcome to java.

Đọc ghi file trong java

Có 3 kiểu stream được sử dụng để **đọc ghi file trong java** đó là byte stream, character stream và buffered stream. Ngoài ra bạn còn có thể sử dụng Scanning (đọc) và Formatting (ghi).

Nội dung chính

- 1. Đọc ghi file trong java với byte stream
 - Sử dụng Byte Stream
- 2. Đọc ghi file trong java với character stream
 - Sử dụng Character Stream
- 3. Đọc ghi file trong java với buffered stream
 - Lớp wrapper cho byte stream
 - Lớp wrapper cho character stream
 - Sử dụng buffered stream để đọc ghi file
- 4. Đọc ghi file trong java với Scanning và Formatting
 - Scanning
 - Formatting

1. Đọc ghi file trong java với byte stream

Các chương trình sử dụng Byte Stream để đọc ghi dữ liệu theo từng byte(8bit). Tất cả các class Byte Stream có nguồn gốc từ InputStream và OutputStream.

Sử dụng Byte Stream

Có rất nhiều class Byte Stream, để hình dung Byte Stream hoạt động như thế nào, chúng ta sẽ tập trung vào FileInputStream và FileOutputStream, ví dụ:

```
public class CopyFileByte {
    public static void main(String [] args) throws IOException {
        FileInputStream inputStream = null;
```

```

        FileOutputStream outputStream = null;

        try {
            inputStream = new FileInputStream("inStream.txt");
            outputStream = new FileOutputStream("outStream.txt");

            int c;
            while ((c = inputStream.read()) != -1) {
                outputStream.write(c);
            }
        } finally {
            if (inputStream != null) {
                inputStream.close();
            }
            if (outputStream != null) {
                outputStream.close();
            }
        }
    }
}

```

Việc đóng một Stream khi mà không có nhu cầu sử dụng nó nữa là một việc rất quan trọng - tránh bị leak tài nguyên. Ví dụ trên sử dụng khối finally để đảm bảo cả 2 Streams (input, output) đều được đóng ngay cả khi có lỗi xảy ra.

2. Đọc ghi file trong java với character stream

Byte Stream trong Java được sử dụng để thực hiện input và output của các byte (8 bit), trong khi đó, Character Stream trong Java được sử dụng để thực hiện input và output cho Unicode 16 bit. Tất cả các class Character Stream có nguồn gốc từ Reader và Writer.

Sử dụng Character Stream

Mặc dù có nhiều lớp liên quan tới Character Stream nhưng các lớp thường dùng nhất là FileReader và FileWriter, ví dụ:

```

public class CopyFileCharacter {

    public static void main(String [] args) throws IOException {

```



```

    FileReader in = null;

    FileWriter out = null;

    try {

        in = new FileReader("input.txt");
        out = new FileWriter("output.txt");

        int c;
        while ((c = in.read()) != -1) {
            out.write(c);
        }
    } finally {
        if (in != null) {
            in.close();
        }
        if (out != null) {
            out.close();
        }
    }
}

```

3. Đọc ghi file trong java với buffered stream

Các ví dụ trên không sử dụng Buffered Streams, điều này có nghĩa là việc đọc và xuất dữ liệu được thực hiện trực tiếp dưới quyền điều khiển của hệ điều hành, gây lãng phí thời gian và tài nguyên. Để giảm thiểu những trên, Buffered Streams đã được sinh ra. Buffered Streams được sử dụng để tăng tốc độ hoạt động I/O, bằng cách đơn giản là tạo ra một khoảng nhớ đệm với kích thước cụ thể nào đó. Vì vậy chúng ta không cần phải truy cập vào ổ đĩa cứng khi thực hiện I/O. Một chương trình có thể chuyển đổi từ không sử dụng buffered stream (Byte Stream và Character Stream sang sử dụng buffered stream bằng việc sử dụng ý tưởng "Wrapping"

Lớp wrapper cho byte stream

1. BufferedInputStream
2. BufferedOutputStream

Lớp wrapper cho character stream

1. BufferedReader
2. BufferedWriter

Sử dụng buffered stream để đọc ghi file

Ví dụ: Sử dụng Wrapper cho byte stream

```
public class CopyFileBuffer1 {  
    public static void main(String [] args) throws IOException {  
        BufferedInputStream bufferIn = null;  
        BufferedOutputStream bufferOut = null;  
  
        try {  
            InputStream inputStream = new FileInputStream("input.txt");  
            OutputStream outputStream = new FileOutputStream("output.txt");  
  
            bufferIn = new BufferedInputStream(inputStream);  
            bufferOut = new BufferedOutputStream(outputStream);  
  
            int c;  
            while ((c = bufferIn.read()) != -1) {  
                bufferOut.write(c);  
            }  
        } finally {  
            if (bufferIn != null) {  
                bufferIn.close();  
            }  
        }  
    }  
}
```

```

    }

    if (bufferOut != null) {
        bufferOut.close();
    }
}

}
}

```

Ví dụ: Sử dụng Wrapper cho character stream

```

public class CopyFileBuffer2 {
    public static void main(String [] args) throws IOException {
        BufferedReader bufferedReader = null;
        BufferedWriter bufferedWriter = null;

        try {
            Reader reader = new FileReader("input.txt");
            Writer writer = new FileWriter("output.txt");

            bufferedReader = new BufferedReader(reader);
            bufferedWriter = new BufferedWriter(writer);

            int c;
            while ((c = bufferedReader.read()) != -1) {
                bufferedWriter.write(c);
            }
        } finally {
            if (bufferedReader != null) {
                bufferedReader.close();
            }
            if (bufferedWriter != null) {
                bufferedWriter.close();
            }
        }
    }
}

```

4. Đọc ghi file trong java với Scanning và Formatting

Scanning

Lớp Scanner có thể phân tích được các kiểu dữ liệu nguyên thủy và kiểu String bằng việc sử dụng biểu thức chính quy (regular expressions). Mặc định các khoảng trắng được Scanner dùng để phân biệt các ký tự trong một chuỗi.

Sử dụng Scanner

Scanner có thể được sử dụng để nhập dữ liệu từ bàn phím, phân tích chuỗi ký tự, nhập dữ liệu từ file.

Ví dụ 1: Đoạn code này cho phép người dùng đọc ký tự số từ bàn phím bằng cách khởi tạo Scanner sử dụng System.in:

```
Scanner sc = new Scanner(System.in);

inti = sc.nextInt();
```

Ví dụ 2: Đoạn code này lấy tất cả các số có kiểu long từ file myNumbers.txt:

```
Scanner sc = new Scanner(new File("myNumbers.txt"));

while (sc.hasNextLong()) {

    long aLong = sc.nextLong();

}
```

Ví dụ 3: Scanner cũng có thể sử dụng dụng bộ phân cách (Delimiters) khác do người dùng tự định nghĩa, đoạn code sau đọc và phân tích thành các mục từ một chuỗi.

```
String input = "1 fish 2 fish red fish blue fish";

Scanner s = new Scanner(input).useDelimiter("\\s*fish\\s*");

System.out.println(s.nextInt());

System.out.println(s.nextInt());

System.out.println(s.next());

System.out.println(s.next());

s.close();
```

Một số phương thức của lớp Scanner

Phương thức	Mô tả
nextInt()	trả về kiểu int
nextFloat()	trả về kiểu float

nextBoolean()	trả về kiểu boolean
nextByte()	trả về kiểu byte
nextLine()	trả về kiểu String

Formatting

Ngoài 2 phương thức print và println. Java cung cấp 2 phương thức định dạng in cho chúng ta là printf và format và 2 phương thức này có chức năng tương tự nhau. Bạn có thể gọi phương thức này ở bất kỳ nơi đâu trong code của bạn.

Ví dụ: định dạng in cho các biến số:

```
System.out.format("The value of " + "the float variable is " +
    "%f, while the value of the " + "integer variable is %d, " +
    "and the string is %s", floatVar, intVar, stringVar);
```

Một điều nữa là ta có thể in theo hệ thống định dạng Pháp, ngăn cách phần nguyên và phần thập phân của số thực thập phân bằng định dạng Locale.FRANCE, ví dụ:

```
System.out.format(Locale.FRANCE,
    "The value of the float " + "variable is %f, while the " +
    "value of the integer variable " + "is %d, and the string is %s%n",
    floatVar, intVar, stringVar);
```

Ví dụ: một chương trình đơn giản:

```
public class TestFormat {
    public static void main(String[] args) {
        long n = 461012;
        System.out.format("%d%n", n);          // --> "461012"
        System.out.format("%08d%n", n);       // --> "00461012"
        System.out.format("%+8d%n", n);       // --> " +461012"
        System.out.format("% ,8d%n", n);      // --> " 461,012"
        System.out.format("%+,8d%n%n", n);    // --> "+461,012"

        double pi = Math.PI;
        System.out.format("%f%n", pi);        // --> "3.141593"
        System.out.format("%.3f%n", pi);      // --> "3.142"
        System.out.format("%10.3f%n", pi);    // --> "      3.142"
        System.out.format("%-10.3f%n", pi);   // --> "3.142"
```

```

        System.out.format(Locale.FRANCE,
                        "%-10.4f%n%n", pi); // --> "3,1416"

        Calendar c = Calendar.getInstance();

        System.out.format("%tB %te, %tY%n", c, c, c); // --> "May 29, 2006"

        System.out.format("%tl:%tM %tp%n", c, c, c); // --> "2:34 am"

        System.out.format("%tD%n", c); // --> "05/29/06"
    }
}

```

Bảng sau thông kê một vài "Converters" và "Flags" được sử dụng trong ví dụ trên:

Converter	Flag	Giải thích
d		Một số nguyên thập phân.
f		Một số float.
n		Một ký tự dòng mới phù hợp với nền tảng chạy ứng dụng. Bạn nên luôn luôn sử dụng <code>%n</code> , hơn là <code>\n</code> .
tB		In ra tên tháng trong biến date & time, ví dụ: April.
td, te		In ra ngày trong biến date & time, 2 số của ngày trong tháng. td sẽ in ra cả số 0 chẳng hạn như ngày 07, te thì không.
ty, tY		In ra năm trong biến date & time, ty = 2-số cuối của năm, tY = 4-số của năm.
tl		In ra giờ trong biến date & time, theo định dạng 12-giờ.
tM		In ra phút trong biến date & time, bao gồm 2 số, cả số 0 nếu cần thiết.
tp		In ra date & time dưới dạng am/pm (chữ thường).
tm		In ra tháng của biến date & time, gồm 2 số, cả số 0 nếu cần thiết.
tD		In ra ngày của biến date & time như định dạng %tm%td%ty
	08	Chỉ định chiều rộng là 8 ký tự, bao gồm cả số 0 nếu cần thiết.

	+	In ra ký tự +.
	,	Nhóm các số bằng dấu ',' thay vì '.', ví dụ 10,000,900 thay vì 10.000.900 .
	-	Căn trái.
	.3	Ba số sau dấu thập phân.
	10.3	Rộng 10 ký tự, căn phải, với 3 số sau dấu thập phân.

Append nội dung vào file trong java

Trong java, bạn có thể sử dụng `FileWriter(file, true)` để append thêm nội dung mới vào cuối file.

1. Tất cả nội dung được ghi đè

```
1 new FileWriter(file);
```

2. Giữ nội dung đang tồn tại và thêm nội dung mới vào cuối file

```
1 new FileWriter(file, true);
```

Ví dụ: Append nội dung vào file trong java

```
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

public class AppendToFileExample {
    private static final String FILENAME = "D:\\testout.txt";

    public static void main(String[] args) {
        BufferedWriter bw = null;
        FileWriter fw = null;

        try {
            String data = " This is new content";
            File file = new File(FILENAME);
            // if file doesnt exists, then create it
            if (!file.exists()) {
                file.createNewFile();
            }
            // true = append file
            fw = new FileWriter(file.getAbsolutePath(), true);
            bw = new BufferedWriter(fw);
            bw.write(data);
            System.out.println("Success...");
        } catch (IOException e) {
```

```

        e.printStackTrace();
    } finally {
        try {
            if (bw != null)
                bw.close();
            if (fw != null)
                fw.close();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
}

```

Giả sử testout.txt có nội dung như sau:

Hello Java,

Output:

Hello Java, This is new content

Delete file trong java

Để delete file trong java, bạn có thể sử dụng phương thức **delete()** của lớp **File**. Nó trả về true nếu delete thành công và false nếu delete không thành công.

Dưới đây là ví dụ về delete file trong java

```

import java.io.File;

public class DeleteFileExample {

    public static void main(String[] args) {
        try {
            File file = new File("D:\\testout2.txt");
            if (file.delete()) {
                System.out.println(file.getName() + " is deleted!");
            } else {
                System.out.println("Delete operation is failed.");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

Output:

testout2.txt is deleted!

Xóa nhiều file dựa vào phần mở rộng trong java

Trong java, bạn có thể implements giao tiếp FilenameFilter và override phương thức accept(File dir, String name) để filter file theo phần mở rộng.

Ví dụ 1: implement giao tiếp FilenameFilter để liệt kê ra các file có phần mở rộng là ".txt", sau đó thực hiện xóa từng file.

```
import java.io.File;
import java.io.FilenameFilter;

public class DeleteFilesExample {
    private static final String FILE_DIR = "D:\\log";
    private static final String FILE_TEXT_EXT = ".txt";

    public static void main(String args[]) {
        new DeleteFilesExample().deleteFile(FILE_DIR, FILE_TEXT_EXT);
    }

    public void deleteFile(String folder, String ext) {

        GenericExtFilter filter = new GenericExtFilter(ext);
        File dir = new File(folder);

        // list out all the file name with .txt extension
        String[] list = dir.list(filter);

        // delete files
        if (list.length == 0) {
            File fileDelete;
            for (String file : list) {
                String temp = new
StringBuffer(FILE_DIR).append(File.separator)
                .append(file).toString();
                fileDelete = new File(temp);
                boolean isdeleted = fileDelete.delete();
                System.out.println("file : " + temp + " is deleted : " +
isdeleted);
            }
        }

        // inner class, generic extension filter
        public class GenericExtFilter implements FilenameFilter {

            private String ext;

            public GenericExtFilter(String ext) {
                this.ext = ext;
            }

            public boolean accept(File dir, String name) {
                return (name.endsWith(ext));
            }
        }
    }
}
```

Giả sử trong folder log có chứa các file: a.txt, b.txt, c.txt, d.txt:

Output:

```
file : D:\log\a.txt is deleted : true
file : D:\log\b.txt is deleted : true
file : D:\log\c.txt is deleted : true
file : D:\log\d.txt is deleted : true
```

Ví dụ 2, Delete nhiều file trong java sử dụng lớp nặc danh

```
import java.io.File;
import java.io FilenameFilter;

public class DeleteFilesExample2 {
    private static final String FILE_DIR = "D:\\log";
    private static final String FILE_TEXT_EXT = ".txt";

    public static void deleteFiles(String folder, String ext) {
        File dir = new File(folder);

        // list out all the file name with .txt extension
        String[] list = dir.list(new FilenameFilter() {

            @Override
            public boolean accept(File file, String name) {
                return name.endsWith(FILE_TEXT_EXT);
            }
        });

        // delete files
        if (list.length == 0) {
            File fileDelete;
            for (String file : list) {
                String temp = new
StringBuffer(FILE_DIR).append(File.separator)
                .append(file).toString();
                fileDelete = new File(temp);
                boolean isdeleted = fileDelete.delete();
                System.out.println("file : " + temp + " is deleted : " +
isdeleted);
            }
        }
    }
}
```

Giả sử trong folder log có chứa các file: a.txt, b.txt, c.txt, d.txt:

Output:

```
file : D:\log\a.txt is deleted : true
file : D:\log\b.txt is deleted : true
file : D:\log\c.txt is deleted : true
file : D:\log\d.txt is deleted : true
```

Tìm các file dựa vào phần mở rộng trong java

Ví dụ 1:

Dưới đây là ví dụ về tìm các file dựa vào phần mở rộng trong java bằng cách implements giao tiếp FilenameFilter, trong ví dụ này chúng ta tìm kiếm các file có phần mở rộng là .txt trong thư mục log.

File: FindFileByExtension.java

```
import java.io.File;
import java.io.FilenameFilter;

public class FindFileByExtension {
    private static final String FILE_DIR = "D:\\log";
    private static final String FILE_TEXT_EXT = ".txt";

    public static void main(String args[]) {
        new FindFileByExtension().listFile(FILE_DIR, FILE_TEXT_EXT);
    }

    public void listFile(String folder, String ext) {
        GenericExtFilter filter = new GenericExtFilter(ext);
        File dir = new File(folder);

        if (dir.isDirectory() == false) {
            System.out.println("Directory does not exists : " + FILE_DIR);
            return;
        }
        // list out all the file name and filter by the extension
        String[] list = dir.list(filter);
        if (list.length == 0) {
            System.out.println("no files end with : " + ext);
            return;
        }

        for (String file : list) {
            String temp = new
StringBuffer(FILE_DIR).append(File.separator).append(file).toString();
            System.out.println("file : " + temp);
        }
    }

    // inner class, generic extension filter
    public class GenericExtFilter implements FilenameFilter {
        private String ext;

        public GenericExtFilter(String ext) {
            this.ext = ext;
        }

        public boolean accept(File dir, String name) {
            return (name.endsWith(ext));
        }
    }
}
```

Ví dụ 2:

Dưới đây là ví dụ về tìm các file dựa vào phần mở rộng trong java bằng cách sử dụng lớp nặc danh, trong ví dụ này chúng ta tìm kiếm các file có phần mở rộng là .txt trong thư mục log.

File: FindFileByExtension2.java

```
import java.io.File;
import java.io FilenameFilter;

public class FindFileByExtension2 {
    private static final String FILE_DIR = "D:\\log";
    private static final String FILE_TEXT_EXT = ".txt";

    public static void main(String args[]) {
        new FindFileByExtension2().listFile(FILE_DIR, FILE_TEXT_EXT);
    }

    public void listFile(String folder, String ext) {
        File dir = new File(folder);

        if (dir.isDirectory() == false) {
            System.out.println("Directory does not exists : " + FILE_DIR);
            return;
        }
        // list out all the file name and filter by the extension
        String[] list = dir.list(new FilenameFilter() {
            @Override
            public boolean accept(File dir, String name) {
                return name.endsWith(FILE_TEXT_EXT);
            }
        });

        if (list.length == 0) {
            System.out.println("no files end with : " + ext);
            return;
        }

        for (String file : list) {
            String temp = new
            StringBuffer(FILE_DIR).append(File.separator).append(file).toString();
            System.out.println("file : " + temp);
        }
    }
}
```

Đổi tên file trong java

Phương thức **renameTo()** của lớp File được sử dụng để đổi tên file trong java. Phương thức này phụ thuộc vào hệ điều hành, ví dụ bạn có thể đổi tên file thành công trên *nix, nhưng thất bại trên Windows.

Dưới đây là ví dụ về đổi tên file trong java

File: RenameFileExample.java

```
import java.io.File;

public class RenameFileExample {
    public static void main(String[] args) {
        File oldfile = new File("D:\\oldfile.txt");
        File newfile = new File("D:\\newfile.txt");

        if (oldfile.renameTo(newfile)) {
            System.out.println("Rename succesful");
        } else {
            System.out.println("Rename failed");
        }
    }
}
```

Copy file trong java

Java không hỗ trợ sẵn phương thức nào để copy file. Tuy nhiên, bạn có thể tự tạo ra chức năng này. Để copy file trong java, bạn phải chuyển đổi file thành dạng byte stream với **FileInputStream** và ghi các bytes đó vào một file khác với **FileOutputStream**.

Dưới đây là ví dụ về copy file trong java

File: CopyFileExample.java

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

public class CopyFileExample {
    public static void main(String[] args) throws IOException {
        InputStream inStream = null;
        OutputStream outStream = null;

        try {
            inStream = new FileInputStream(new File("D:\\file1.txt"));
            outStream = new FileOutputStream(new File("D:\\file2.txt"));

            int length;
            byte[] buffer = new byte[1024];

            // copy the file content in bytes
            while ((length = inStream.read(buffer)) > 0) {
                outStream.write(buffer, 0, length);
            }
            System.out.println("File is copied successful!");
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            inStream.close();
            outStream.close();
        }
    }
}
```

```
}
}
```

Các bạn có thể xem các cách copy file trong java khác trong bài [Đọc ghi file trong java](#)

Move file trong java

Trong java không có sẵn phương thức nào để move (di chuyển) file từ thư mục này sang thư mục khác. Tuy vậy, bạn vẫn có thể move file trong java bằng 1 trong 2 cách sau:

1. File.renameTo().
2. Copy tới một file mới và xóa file cũ.

Dưới đây là 2 ví dụ cho 2 cách trên để move file "file1.txt" từ thư mục "D:\\log" tới thư mục "D:\\logtest".

1. Sử dụng File.renameTo()

Phương thức renameTo() hoạt động được trên nền tảng *nix.

```
import java.io.File;

public class MoveFileExample1 {

    public static void main(String[] args) {

        try {

            File afile = new File("D:\\log\\file1.txt");

            if (afile.renameTo(new File("D:\\logtest\\" + afile.getName()))) {

                System.out.println("File is moved successful!");

            } else {

                System.out.println("File is failed to move!");

            }

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}
```

2. Copy tới một file mới và xóa file cũ

Cách này thực hiện thành công trên cả *nix và Windows

```
import java.io.File;
```

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

public class MoveFileExample2 {
    public static void main(String[] args) throws IOException {
        InputStream inStream = null;
        OutputStream outStream = null;

        try {
            File fileLog1 = new File("D:\\log\\file1.txt");
            File fileLog2 = new File("D:\\logtest\\file1.txt");
            inStream = new FileInputStream(fileLog1);
            outStream = new FileOutputStream(fileLog2);

            int length;
            byte[] buffer = new byte[1024];
            // copy the file content in bytes
            while ((length = inStream.read(buffer)) > 0) {
                outStream.write(buffer, 0, length);
            }
            // delete the original file
            fileLog1.delete();
            System.out.println("File is copied successful!");
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            inStream.close();
            outStream.close();
        }
    }
}
```

Check file tồn tại trong java

Phương thức **exists()** của lớp File được sử dụng để check file có tồn tại không.

Dưới đây là ví dụ check file tồn tại trong java

```
import java.io.File;

public class FileChecker {
    public static void main(String args[]) {
        File f = new File("D:\\testout.txt");

        if (f.exists()) {
            System.out.println("File existed");
        } else {
            System.out.println("File not found!");
        }
    }
}
```

Check file ẩn trong java

Phương thức **isHidden()** của lớp File được sử dụng để check file có bị ẩn không.

Dưới đây là ví dụ check file ẩn trong java

```
import java.io.File;
import java.io.IOException;

public class FileHiddenExample {
    public static void main(String[] args) throws IOException {
        File file = new File("D:\\test_hidden.txt");

        if (file.isHidden()) {
            System.out.println("This file is hidden");
        } else {
            System.out.println("This file is not hidden");
        }
    }
}
```

Note: Phương thức isHidden() phụ thuộc vào nền tảng, trên nền tảng UNIX, một file được coi là ẩn nếu tên của nó bắt đầu với dấu "chấm" ('.'); Trên nền tảng Microsoft Windows, một tệp được coi là ẩn, nếu file đó được đánh dấu là ẩn trong thuộc tính tệp.

Liệt kê tất cả file và folder trong một folder

Dưới đây là ví dụ liệt kê tất cả file và folder trong một folder sử dụng hàm đệ quy:

```
import java.io.File;
import java.util.Arrays;
import java.util.Comparator;

public class MyFiles {
    public static void main(String[] args) {
        File fileOrDir = new File("E:\\web_tut");
        MyFiles myFiles = new MyFiles();
        myFiles.traverseDepthFiles(fileOrDir);
    }

    public void traverseDepthFiles(final File fileOrDir) {
        // check xem fileOrDir là file hay folder
        if (fileOrDir.isDirectory()) {
            // in tên folder ra màn hình
            System.out.println(fileOrDir.getAbsolutePath());

            final File[] children = fileOrDir.listFiles();
            if (children == null) {
                return;
            }
            // sắp xếp file theo thứ tự tăng dần
            Arrays.sort(children, new Comparator<File>() {
                public int compare(final File o1, final File o2) {
                    return o1.getName().compareTo(o2.getName());
                }
            });
            for (final File each : children) {
                // gọi lại hàm traverseDepthFiles()
                traverseDepthFiles(each);
            }
        } else {
            // in tên file ra màn hình
            System.out.println(fileOrDir.getAbsolutePath());
        }
    }
}
```

Output:

```
E:\web_tut
E:\web_tut\css
E:\web_tut\css\css-la-gi.txt
E:\web_tut\java
E:\web_tut\java\java-io
E:\web_tut\java\java-io\java-io-la-gi.txt
E:\web_tut\java\java-la-gi.txt
```

Get thư mục hiện tại trong Java

Bài này hướng dẫn bạn 3 cách để có get thư mục hiện tại trong java:

- Get thư mục hiện tại trong java bằng việc sử dụng property "user.dir".
- Get thư mục hiện tại trong java bằng việc sử dụng **ClassName.class.getProtectionDomain()**.
- Get thư mục hiện tại trong java bằng việc sử dụng **file.getAbsolutePath()**.

Nội dung chính

- 1. Sử dụng property "user.dir"
- 2. Sử dụng **ClassName.class.getProtectionDomain()**
- 3. Sử dụng **file.getAbsolutePath()**

1. Sử dụng property "user.dir"

Example 1: Get thư mục hiện tại trong java bằng việc sử dụng phương thức **getProperty()** với đối số "user.dir".

```
package com.realtut;

public class GetCurrentDirExample1 {

    public static void main(String[] args) {

        String currentDirectory = System.getProperty("user.dir");

        System.out.println("current dir: " + currentDirectory);

    }

}
```

Kết quả:

```
current dir: D:\workspace\java-learn
```

Read properties file trong java

Properties file (.properties) chủ yếu được sử dụng trong các công nghệ liên quan đến Java để lưu trữ các tham số có thể cấu hình của một ứng dụng. Properties là các giá trị được quản lý theo các cặp **key/value**. Trong mỗi cặp, key và value đều có kiểu String. Key được sử dụng để truy xuất value, giống như tên biến được sử dụng để truy xuất giá trị của biến.

Bài này hướng dẫn bạn cách **read properties file trong java**:

1. Read properties file từ thư mục hiện tại.

2. Read properties file từ resources (classpath).
3. Read properties file bằng cách sử dụng Singleton pattern.

Nội dung chính

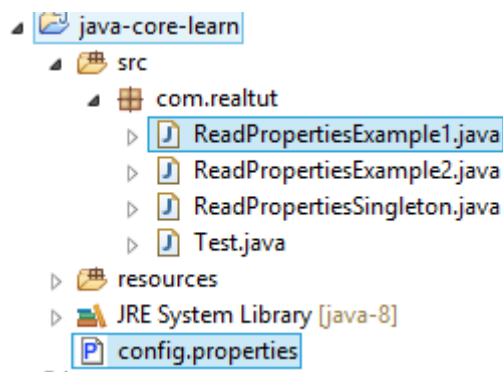
- Định nghĩa file properties
- 1. Read properties file từ thư mục hiện tại
- 2. Read properties file từ resources (classpath)
- 3. Read properties file sử dụng Singleton pattern

Định nghĩa file properties

File properties có nội dung như sau, được sử dụng trong các ví dụ của bài này:

```
1 username=admin
2 password=123456
```

1. Read properties file từ thư mục hiện tại



Tạo file config.properties trong cùng thư mục với java resources.

File: ReadPropertiesExample1.java

```
package com.realtut;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;

public class ReadPropertiesExample1 {
```

```
private static final String FILE_CONFIG = "\\config.properties";

public static void main(String[] args) {
    Properties properties = new Properties();
    InputStream inputStream = null;
    try {
        String currentDir = System.getProperty("user.dir");
        inputStream = new FileInputStream(currentDir + FILE_CONFIG);

        // load properties from file
        properties.load(inputStream);

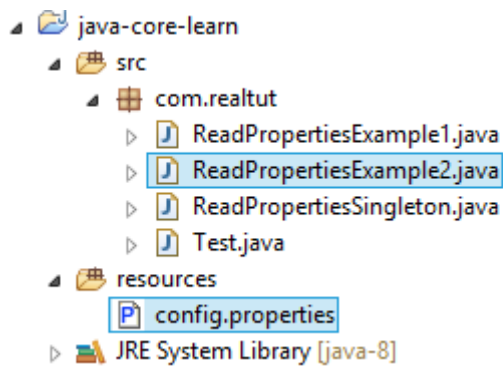
        // get property by name
        System.out.println(properties.getProperty("username"));
        System.out.println(properties.getProperty("password"));

    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        // close objects
        try {
            if (inputStream != null) {
                inputStream.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Kết quả:

```
admin
123456
```

2. Read properties file từ resources (classpath)



Tạo file config.properties trong java resources.

File: ReadPropertiesExample2.java

```
package com.realtut;

import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;

public class ReadPropertiesExample2 {

    private static final String FILE_CONFIG = "\\config.properties";

    public static void main(String[] args) {

        Properties properties = new Properties();

        InputStream inputStream = null;

        try {

            inputStream = ReadPropertiesExample2.class.getClassLoader()

                .getResourceAsStream(FILE_CONFIG);

            // load properties from file
            properties.load(inputStream);

            // get property by name
            System.out.println(properties.getProperty("username"));
            System.out.println(properties.getProperty("password"));

        } catch (IOException e) {

            e.printStackTrace();

        } finally {

            // close objects
        }
    }
}
```

```

        try {
            if (inputStream != null) {
                inputStream.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

Kết quả:

```

admin
123456

```

3. Read properties file sử dụng Singleton pattern

File: ReadPropertiesSingleton.java

```

package com.realtut;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;

public class ReadPropertiesSingleton {

    private static final String FILE_CONFIG = "\\config.properties";
    private static ReadPropertiesSingleton instance = null;
    private Properties properties = new Properties();

    /**
     * Use singleton pattern to create ReadConfig object one time and use
     * anywhere
     *
     * @return instance of ReadConfig object
     */
    public static ReadPropertiesSingleton getInstance() {
        if (instance == null) {

```

```

        instance = new ReadPropertiesSingleton();
        instance.readConfig();
    }
    return instance;
}

/**
 * get property with key
 *
 * @param key
 * @return value of key
 */
public String getProperty(String key) {
    return properties.getProperty(key);
}

/**
 * read file .properties
 */
private void readConfig() {
    InputStream inputStream = null;
    try {
        String currentDir = System.getProperty("user.dir");
        inputStream = new FileInputStream(currentDir + FILE_CONFIG);
        properties.load(inputStream);
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        // close objects
        try {
            if (inputStream != null) {
                inputStream.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

```
    }  
    }  
}
```

File: Test.java

```
package com.realtut;  
  
public class Test {  
    public static void main(String[] args) {  
        ReadPropertiesSingleton demo =  
ReadPropertiesSingleton.getInstance();  
        System.out.println(demo.getProperty("username"));  
        System.out.println(demo.getProperty("password"));  
    }  
}
```

Kết quả:

```
admin  
123456
```


Thao Tác Với Folder Trong Java

Check thư mục rỗng trong java

Để check thư mục rỗng trong java, bạn có thể sử dụng phương thức **java.io.File.list()** để check xem có bất kỳ thư mục hoặc file nào tồn tại trong thư mục đã cho không.

Nội dung chính

- Ví dụ check thư mục rỗng trong java

Ví dụ check thư mục rỗng trong java

```
import java.io.File;

public class CheckFolderEmptyExample {

    public static void main(String[] args) {

        File file = new File("D:\\testout");

        if (file.isDirectory()) {

            // check thu muc rong bang phuong thuc file.list()

            if (file.list().length == 0) {

                System.out.println("Thu muc "

                    + file.getAbsolutePath() + " la rong!");

            } else {

                System.out.println("Thu muc "

                    + file.getAbsolutePath() + " khong rong!");

            }

        } else {

            System.out.println(file.getAbsolutePath() + " khong phai la thu muc!");

        }

    }

}
```

Output:

```
Thu muc D:\testout la rong!
```

Tạo đường dẫn thư mục trong java

Để tạo đường dẫn thư mục trong java, bạn có thể sử dụng phương thức mkdir() hoặc mkdirs() của lớp java.io.File. Hoặc với JDK 7 trở lên bạn có thể sử dụng phương thức static createDirectories() của lớp java.nio.file.Files.

1. Phương thức mkdir() và mkdirs() của lớp java.io.File

1.1. Phương thức mkdir()

Phương thức mkdir() được sử dụng để tạo ra đường dẫn thư mục duy nhất.

Khai báo:

```
1 new File("C:\\Directory1").mkdir();
```

Ví dụ:

```
import java.io.File;

public class CreateDirectoryExample1 {
    public static void main(String[] args) {
        File file = new File("D:\\Directory1");
        if (!file.exists()) {
            if (file.mkdir()) {
                System.out.println("Directory is created!");
            } else {
                System.out.println("Failed to create directory!");
            }
        }
    }
}
```

1.2. Phương thức mkdirs()

Phương thức mkdirs() được sử dụng để tạo ra đường dẫn thư mục và tất cả các đường dẫn thư mục con của nó.

Khai báo:

```
new File("C:\\Directory2\\Sub2\\Sub-Sub2").mkdirs();
```

Ví dụ:

```
import java.io.File;

public class CreateDirectoryExample2 {
    public static void main(String[] args) {
        File files = new File("D:\\Directory2\\Sub2\\Sub-Sub2");
        if (!files.exists()) {
            if (files.mkdirs()) {
                System.out.println("Multiple directories are created!");
            }
        }
    }
}
```

```

        } else {
            System.out.println("Failed to create multiple
directories!");
        }
    }
}

```

2. Phương thức static createDirectories() của lớp java.nio.file.Files

Với JDK 7 trở lên bạn có thể sử dụng phương thức static createDirectories() của lớp java.nio.file.Files.

Cú pháp:

```

Path path = Paths.get("C:\\Directory1");
Files.createDirectories(path);

```

Ví dụ:

```

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;

public class CreateDirectoryExample2 {
    public static void main(String[] args) {
        Path path3 = Paths.get("D:\\Directory3");
        Path path4 = Paths.get("D:\\Directory4\\Sub4\\Sub-Sub4");

        // Tạo đường dẫn thư mục duy nhất
        if (!Files.exists(path3)) {
            try {
                Files.createDirectory(path3);
                System.out.println("Create path = " + path3 + " successfully!");
            } catch (IOException e) {
                e.printStackTrace();
            }
        } else {
            System.out.println("Path = " + path3 + " is existed!");
        }

        // Tạo đường dẫn thư mục với các thư mục con
        if (!Files.exists(path4)) {
            try {
                Files.createDirectories(path4);
                System.out.println("Create path = " + path4 + " successfully!");
            } catch (IOException e) {
                e.printStackTrace();
            }
        } else {
            System.out.println("Path = " + path4 + " is existed!");
        }
    }
}

```

Xóa thư mục trong java

[Xóa thư mục trong java - Delete folder trong java] Để xóa thư mục trong java thì thư mục bị xóa phải là thư mục trống, nghĩa là thư mục bị xóa đó không chứa bất kỳ thư mục con hoặc file nào. Do vậy để xóa thư mục trong java thì chúng ta phải xóa tất cả các thư mục con và các file bên trong nó.

Để xóa thư mục trong java các bạn có thể sử dụng 2 cách sau:

1) Phương thức delete() của lớp java.io.File

Cú pháp:

```
File file = new File(folder);
file.delete();
```

2) Phương thức static delete() của lớp java.nio.file.Files

```
Path path = Paths.get(folder);
Files.delete(path);
```

Nội dung chính

- Ví dụ về xóa thư mục trong java

Ví dụ về xóa thư mục trong java

Ví dụ về xóa thư mục trong java dưới đây sử dụng phương pháp đệ quy để xóa tất cả các thư mục con và file của thư mục bị xóa rồi mới xóa thư mục.

```
import java.io.File;

public class DeleteFolderExample {

    /**
     * main
     *
     * @param args
     */
    public static void main(String[] args) {
        File file = new File("D:\\testout");
        deleteDir(file);
    }

    /**
```

```

    * delete folder
    *
    * @param file
    */
    public static void deleteDir(File file) {
        // neu file la thu muc thi xoa het thu muc con va file cua no
        if (file.isDirectory()) {
            // liet ke tat ca thu muc va file
            String[] files = file.list();
            for (String child : files) {
                File childDir = new File(file, child);
                if (childDir.isDirectory()) {
                    // neu childDir la thu muc thi goi lai phuong thuc deleteDir()
                    deleteDir(childDir);
                } else {
                    // neu childDir la file thi xoa
                    childDir.delete();
                    System.out.println("File bi da bi xoa "
                        + childDir.getAbsolutePath());
                }
            }
            // Check lai va xoa thu muc cha
            if (file.list().length == 0) {
                file.delete();
                System.out.println("File bi da bi xoa " + file.getAbsolutePath());
            }
        } else {
            // neu file la file thi xoa
            file.delete();
            System.out.println("File bi da bi xoa " + file.getAbsolutePath());
        }
    }
}

```

Output:

```
File bị da bị xoa D:\testout\css\style.css
File bị da bị xoa D:\testout\css
File bị da bị xoa D:\testout\index.txt
File bị da bị xoa D:\testout\java\java-core\java-core1.txt
File bị da bị xoa D:\testout\java\java-core\java-core2.txt
File bị da bị xoa D:\testout\java\java-core
File bị da bị xoa D:\testout\java\java-io\java-io1.txt
File bị da bị xoa D:\testout\java\java-io
File bị da bị xoa D:\testout\java
File bị da bị xoa D:\testout
```

Copy thư mục trong java

Không có phương thức copy thư mục nào sẵn có trong java. Để **copy một thư mục trong java** từ vị trí này sang vị trí khác với tất cả các thư mục con và các file mà chúng chứa thì chúng ta phải copy từng thư mục con và file sang vị trí mới.

Nội dung chính

- Ví dụ về copy thư mục trong java

Ví dụ về copy thư mục trong java

Ví dụ 1: sử dụng **java.nio.file.Files.copy()**

```
import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.StandardCopyOption;

public class CopyFolderExample2 {

    /**
     * main
     *
     * @param args
     * @throws IOException
     */
}
```

```

public static void main(String[] args) throws IOException {
    // sourceFolder: thu mục nguồn được copy
    File sourceFolder = new File("D:\\temp");
    // targetFolder: thu mục đích được copy đến
    File targetFolder = new File("D:\\tempNew");
    // gọi phương thức copy
    copyFolder(sourceFolder, targetFolder);
}

/**
 * copy folder
 *
 * @param sourceFolder
 * @param targetFolder
 * @throws IOException
 */
private static void copyFolder(File sourceFolder, File targetFolder)
    throws IOException {
    // Check nếu sourceFolder là một thư mục hoặc file
    // nếu sourceFolder là file thì copy đến thư mục đích
    if (sourceFolder.isDirectory()) {
        // Xác nhận nếu targetFolder chưa tồn tại thì tạo mới
        if (!targetFolder.exists()) {
            targetFolder.mkdir();
            System.out.println("Thư mục đã được tạo " + targetFolder);
        }
        // Liệt kê tất cả các file và thư mục trong sourceFolder
        String files[] = sourceFolder.list();
        for (String file : files) {
            File srcFile = new File(sourceFolder, file);
            File tarFile = new File(targetFolder, file);
            // gọi lại phương thức copyFolder
            copyFolder(srcFile, tarFile);
        }
    } else {

```

```
// copy file tu thuc muc nguon den thu muc dich
Files.copy(sourceFolder.toPath(), targetFolder.toPath(),
           StandardCopyOption.REPLACE_EXISTING);
System.out.println("File da duoc copy " + targetFolder);
    }
}
}
```

Output:

```
Thu muc da duoc tao D:\tempNew
Thu muc da duoc tao D:\tempNew\java
Thu muc da duoc tao D:\tempNew\java\java-core
File da duoc copy D:\tempNew\java\java-core\javacore1.txt
Thu muc da duoc tao D:\tempNew\src
File da duoc copy D:\tempNew\src\test1.txt
File da duoc copy D:\tempNew\test2.txt
File da duoc copy D:\tempNew\test3.txt
```

Lưu ý: Sử dụng phương thức `java.nio.file.Files.copy()` có thể copy được thư mục. Tuy nhiên các file bên trong thư mục không được copy. Vì vậy cần phải tạo ra thư mục mới bằng lệnh `targetFolder.mkdir()`;

Ví dụ 2: sử dụng FileInputStream và FileOutputStream

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

public class CopyFolderExample1 {
    /**
     * main
     *
     * @param args
     * @throws IOException
     */
    public static void main(String[] args) throws IOException {
```



```
// sourceFolder: thu muc nguon duoc copy
File sourceFolder = new File("D:\\temp");

// targetFolder: thu muc dich duoc copy den
File targetFolder = new File("D:\\tempNew");

// goi phuong thuc copy
copyFolder(sourceFolder, targetFolder);
}

/**
 * copy folder
 *
 * @param sourceFolder
 * @param targetFolder
 * @throws IOException
 */
private static void copyFolder(File sourceFolder, File targetFolder)
    throws IOException {
    // Check neu sourceFolder la mot thu muc hoac file
    // neu sourceFolder la file thi copy den thu muc dich
    if (sourceFolder.isDirectory()) {
        // Xac nhan neu targetFolder chua ton tai thi tao moi
        if (!targetFolder.exists()) {
            targetFolder.mkdir();
            System.out.println("Thu muc da duoc tao " + targetFolder);
        }
        // Liet ke tat ca cac file va thu muc trong sourceFolder
        String files[] = sourceFolder.list();
        for (String file : files) {
            File srcFile = new File(sourceFolder, file);
            File tarFile = new File(targetFolder, file);
            // goi lai phuong thuc copyFolder
            copyFolder(srcFile, tarFile);
        }
    } else {
        // copy file tu thuc muc nguon den thu muc dich
    }
}
```

```

        InputStream in = new FileInputStream(sourceFolder);
        OutputStream out = new FileOutputStream(targetFolder);
        byte[] buffer = new byte[1024];
        int length;
        while ((length = in.read(buffer)) > 0) {
            out.write(buffer, 0, length);
        }
        System.out.println("File đã được copy " + targetFolder);
        in.close();
        out.close();
    }
}

```

Output:

```

Thu mục đã được tạo D:\tempNew
Thu mục đã được tạo D:\tempNew\java
Thu mục đã được tạo D:\tempNew\java\java-core
File đã được copy D:\tempNew\java\java-core\javacore1.txt
Thu mục đã được tạo D:\tempNew\src
File đã được copy D:\tempNew\src\test1.txt
File đã được copy D:\tempNew\test2.txt
File đã được copy D:\tempNew\test3.txt

```

ZIP File Trong Java

Trong bài này chúng ta sẽ học về làm thế nào để ZIP file trong java. Vậy thì ZIP là gì? ZIP là định dạng file mà ở đó data của file được mã hóa và nén lại, độ nén phụ thuộc vào dữ liệu của file và thuật toán nén. Nó thường được sử dụng để nén các file và thư mục.

Package **java.util.zip** cung cấp các lớp để thực thi việc ZIP file trong java.

Nội dung chính

- ZIP một file duy nhất

ZIP một file duy nhất

File: ZIPFileExample.java

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.zip.ZipEntry;
import java.util.zip.ZipOutputStream;

public class ZIPFileExample1 {

    /**
     * main
     *
     * @param args
     * @throws IOException
     */
    public static void main(String[] args) throws IOException {
        final File inputFile = new File("D:\\temp\\ziptest.txt");
        final String zipFilePath = "D:\\temp\\ziptest.zip";
        zipFile(inputFile, zipFilePath);
    }
}
```

```

/**
 * zip file
 *
 * @param inputFile
 * @param zipFilePath
 * @throws IOException
 */
private static void zipFile(File inputFile, String zipFilePath)
    throws IOException {
    FileInputStream fis = null;
    FileOutputStream fos = null;
    ZipOutputStream zipos = null;
    try {
        fos = new FileOutputStream(zipFilePath);
        zipos = new ZipOutputStream(fos);

        ZipEntry zipEntry = new ZipEntry(inputFile.getName());
        zipos.putNextEntry(zipEntry);

        fis = new FileInputStream(inputFile);
        byte[] buf = new byte[1024];
        int length;
        while ((length = fis.read(buf)) > 0) {
            zipos.write(buf, 0, length);
        }
        System.out.println("File " + inputFile
            + " đã được zip thành file " + zipFilePath);
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (zipos != null) {
            // close ZipEntry

```

```

        zipos.closeEntry();
        // close ZipOutputStream
        zipos.close();
    }
    // close FileOutputStream
    if (fos != null)
        fos.close();
    // close FileInputStream
    if (fis != null)
        fis.close();
}
}
}

```

Output:

```

File      D:\temp\ziptest.txt      da      duoc      zip      thanh      file
D:\temp\ziptest.zip

```

TÀI LIỆU THAM KHẢO

- [1] Giáo trình, bài giảng “Kỹ năng lập trình ứng dụng với Java ”, Ths. Võ Văn Phúc, Trường đại học Nam Cần Thơ
- [2] Giáo trình, bài giảng “Lập trình Java căn bản” ”, Ths. Võ Văn Phúc, Trường đại học Nam Cần Thơ
- [3] Allen B. Downey & Chris Mayfield, Think Java, 2016
- [4] Lập Trình Hướng Đối Tượng Với Java – Đoàn Văn Ban, NXB KHOA HỌC VÀ KỸ THUẬT, 2005
- [5] Java Cơ Bản – Nguyễn Văn Khoa, NXB Hồng Đức, 2007
- [6] Java Core (Tiếng Việt), UDS Ebook, updateSoft.
- [7] Slide Lập Trình Java - Phạm Quang Dũng, ĐH KHTN
- [8] Lập trình hướng đối tượng – ĐH Công nghệ, ĐHQGHN
- [9] Lập Trình Java: <https://viettuts.vn/java>

Tài liệu tham khảo khác:

- [10] Cay S. Horstmann, *Core Java Volume I*, Prentice Hall, 2016.
- [11] Java Tutorial, <https://www.tutorialspoint.com/java/index.htm>
- [12] Java cơ bản (o7planning.org): <https://o7planning.org/vi/10973/java-co-ban>
- [13] Java cơ bản (vncoder.vn) <https://vncoder.vn/java/lap-trinh-java-co-ban>
- [14] Lập trình Java: <https://yellowcodebooks.com/category/lap-trinh-java/>
- [15] Java nâng cao (o7planning.org): <https://o7planning.org/vi/10985/java-nang-cao>

CÁC LINK BÀI TẬP THAM KHẢO

- [16] 247 bài tập:
https://vietjack.com/bai_tap_java/bai_tap_mau_java_va_vi_du_java.jsp
- [17] 200 câu hỏi phỏng vấn Java:
https://vietjack.com/cau_hoi_phong_van_java/index.jsp
- [18] 200 câu hỏi phỏng vấn Java: <https://plpsoft.vn/interview/list-cau-hoi-phong-van-java-core>