

INTRODUCTION

- Java thừa hưởng cú pháp từ C. Một số ngôn ngữ được thiết kế để biên dịch thành mã máy tùy thuộc theo một số loại CPU (C++). Java sử dụng một cơ chế trung gian qua hai bước chính: biên dịch thành bytecode (.class) và chạy bằng máy ảo Java (JVM). Với cơ chế này java sẽ chạy được đa nền tảng vì bytecode là trung gian và không phụ thuộc vào hệ điều hành. JVM trên mỗi hệ điều hành biên dịch bytecode thành mã máy tương ứng.
- JVM: máy ảo java chịu trách nhiệm thực thi các chương trình Java. Gồm có Class Loader, Bytecode verifier và Execution Engine.
- JDK: bộ công cụ phát triển Java cung cấp tất cả những gì cần thiết để phát triển các ứng dụng Java.
- Để hỗ trợ các nguyên tắc của OOP, Java có kế thừa, đa hình và tính đóng gói.
- Các kiểu dữ liệu của Java nguyên thủy:

boolean	true/false value
int	32-bits integer
float	Single-precision floating point 32-bit
long	64-bits integer
short	16-bits integer
byte	8-bits integer
char	Character unsigned 16-bits Unicode
double	Double-precision floating point 64-bit

- Chuỗi kí tự thoát:

\'	Dấu nháy đơn
\"	Dấu nháy đôi
\\	Dấu gạch chéo ngược
\r	Ký tự quay về đầu dòng
\n	Ký tự kết thúc dòng
\f	Ký tự kết thúc trang và chuyển trang
\t	Ký tự khoảng trắng tab
\b	Xóa ký tự phía trước
\ddd	Hằng số bát phân
\uxxxx	Hằng số thập lục phân

- String trong Java không thể thay đổi, nghĩa là khi tạo ra một chuỗi thì không thể thay đổi nội dung của nó, nghĩa là khi thay đổi một chuỗi Java sẽ tạo ra một đối tượng chuỗi mới với giá trị đã thay đổi.
- Các phương thức của chuỗi:

str.length()	Trả về độ dài của chuỗi
str.charAt(index)	Trả về kí tự tại vị trí index
str.isEmpty()	Kiểm tra xem chuỗi có rỗng không
str.equals(str2)	Kiểm tra xem chuỗi 1 có bằng chuỗi 2 không
str.equalsIgnoreCase(str2)	Kiểm tra 2 chuỗi không quan trọng hoa thường
str.indexOf('e')	Trả về index có ký tự con trong chuỗi
str.indexOf("ee")	Trả về index có chuỗi con trong chuỗi

str.lastIndexOf('e' or "ee")	Trả về index cuối cùng
str.substring(id1, id2)	Trả về chuỗi con
Str.replace('l', 'e')	Thay thế tất cả các ký tự l thành e
Str.replace("hello", "hi")	Thay thế tất cả các chuỗi con
Str.trim()	Loại bỏ khoảng trắng từ đầu và cuối
Str.toLowerCase()	Chuyển chuỗi thành chữ thường
Str.toUpperCase()	Chuyển chuỗi thành chữ in hoa
Str.startsWith("He")	Kiểm tra chuỗi có bắt đầu bằng "He" không
Str.endsWith("He")	Kiểm tra chuỗi có bắt đầu bằng "He" không
Str1.concat(str2)	Nối 2 chuỗi
Str.split(",")	Chia chuỗi thành mảng các chuỗi con
Str.toCharArray()	Chuyển chuỗi thành một mảng ký tự

DATA TYPE AND OPERATOR

- Type[] arrayname = new type[size];
- Type arrayname[][] = new type[size][size];
- Array.length: độ dài mảng
- Cast trong java:
 - + Ép kiểu rộng: dùng khi chuyển đổi từ kiểu dữ liệu nhỏ sang lớp hơn
 - + Ép kiểu hẹp: dùng khi chuyển đổi từ kiểu dữ liệu lớn sang nhỏ
 - + Ép kiểu đối tượng: objectA a = (B) object;

CLASS, OBJECT AND METHOD

- Public static void main(String[] args): khai báo hàm main
- **Garbage Collection**: quá trình tự động quản lý bộ nhớ, giúp dọn dẹp và giải phóng bộ nhớ mà các đối tượng không còn sử dụng, giúp tránh tình trạng rò rỉ bộ nhớ. Nó sẽ tham chiếu đến bộ nhớ heap, nếu không có đối tượng nào tham chiếu đến đối tượng đó nó sẽ được đánh dấu là rác và được giải phóng bộ nhớ.
- Trong java có 3 kiểu truy cập là public (mặc định), private và protected.
- Khi truyền một biến vào hàm, Java sẽ có hai phương thức là truyền theo giá trị (**pass-by-value**) hoặc truyền theo tham chiếu (**pass-by-reference**).
 - + Nếu truyền tham số nguyên thủy, Java sẽ sao chép giá trị và những thay đổi không làm ảnh hưởng biến gốc ngoài hàm.
 - + Nếu truyền tham số kiểu đối tượng như String, Array, Java sẽ sao chép tham chiếu đến các đối tượng đó và truyền vào hàm.
- Nạp chồng phương thức trong java cho phép định nghĩa nhiều phương thức trong cùng lớp với cùng tên nhưng tham số khác nhau.
- static trong Java là một từ khóa rất quan trọng và được sử dụng để khai báo các thành phần (biến, phương thức, block hoặc lớp) có thể truy cập mà không cần tạo đối tượng của lớp chứa chúng. Đối tượng đó trở thành một thành phần của lớp thay vì một đối tượng cụ thể và có thể truy cập thông qua tên lớp.
- **Block static** là một khối mã được thực thi chỉ một lần khi lớp được nạp vào bộ nhớ. Sử dụng để thực hiện các công việc khởi tạo hoặc cấu hình ban đầu cho các biến static trong lớp.
- Block static sẽ được thực thi trước khi bất kỳ phương thức static nào được gọi.

INHERITANCE

- class child extends father {}

- **super** được sử dụng để gọi phương thức của lớp cha hoặc truy cập thuộc tính của lớp cha từ lớp con.
- Khi từ khóa **final** được sử dụng với một biến, biến đó sẽ không thể thay đổi giá trị sau khi được gán giá trị lần đầu tiên:
 - + Với biến cơ bản, final không thể thay đổi giá trị sau khi gán một lần.
 - + Với đối tượng, final không thể tham chiếu đến một đối tượng khác sau khi đã được gán nhưng trạng thái hay các giá trị của thuộc tính của đối tượng đó vẫn có thể thay đổi.

ADVANCE CLASS

- Nested classes: là những lớp được định nghĩa bên trong một lớp khác. Java giới hạn hai loại:
 - + Static nested classes: lớp lồng nhau tĩnh không cần tham chiếu đến đối tượng của lớp bên ngoài để truy cập các thành phần của lớp ngoài.
 - + Non-static nested classes: Lớp lồng nhau không tĩnh có thể truy cập cả các thành phần tĩnh và không tĩnh của lớp ngoài.

```
class OuterClass {
```

```
    private int outerData = 10;
```

```
    // Static Nested Class
```

```
    static class StaticNestedClass {
```

```
        void display() { System.out.println("Inside Static Nested Class"); //
```

```
        Không thể truy cập outerData vì nó không phải static
```

```
    }
```

```
}
```

```
// Non-static Inner Class
```

```
class InnerClass {
```

```
    void display() { System.out.println("Inside Inner Class");
```

```
    System.out.println("Accessing outer data: " + outerData); // Có thể truy cập outerData
```

```
}
```

```
}
```

```
void createInnerClass() { InnerClass inner = new InnerClass(); // Cần đối tượng OuterClass để tạo InnerClass inner.display();
```

```
}
```

```
}
```

- Local class: lớp cục bộ được định nghĩa trong một hàm hay phương thức, có thể truy cập các biến final hoặc các biến hiệu quả của phương thức chứa nó.

- Anonymous class: lớp ẩn danh là những lớp không có tên được định nghĩa ngay tại chỗ khi tạo đối tượng của chúng. Thường sử dụng khi cần tạo một lớp con hoặc cài đặt một giao diện mà không muốn định nghĩa một lớp riêng biệt.

```
Public static void main(String[] args) {
```

```
    Greeting greeting = new greeting() {
```

```
        public void greet() { ... }
```

```
    }
```

```
    Greeting.greet();
```

```
}
```

- Lambda Expression: biểu thức lambda dùng để viết các hàm ẩn danh.
 - + (parameters) -> expression

EXCEPTION

- Trong java, mọi ngoại lệ được thừa hưởng từ lớp Throwable
- Các loại exception trong java:
 - + **SQLException**: lỗi cơ sở dữ liệu
 - + **IOException**: lỗi nhập xuất
 - + **RuntimeException**: lỗi logic

ArithmeticException	Lỗi toán học như chia một số cho 0
IndexOutOfBoundsException	Lỗi truy cập chỉ mục không hợp lệ
ArrayStoreException	Lỗi lưu giá trị không tương thích vào mảng
ClassCastException	Lỗi ép kiểu không phù hợp
IllegalArgumentException	Lỗi nhận tham số không phù hợp
NullPointerException	Lỗi truy cập hoặc gọi phương thức trên null
NumberFormatException	Lỗi chuyển đổi chuỗi thành số không hợp lệ

- In ra lỗi: e.printStackTrace()

PACKAGE AND INTERFACE

Java.lang	Cung cấp các tính năng cần thiết như chuỗi, số học, luồng
Java.io	Chứa các lớp để đọc ghi dữ liệu từ tệp, luồng và các thiết bị
Java.net	Lập trình ứng dụng mạng, có giao thức TCP/IP và UDP
Java.util	Xử lý cấu trúc dữ liệu, thời gian và thao tác tổng quát
Java.awt	Cung cấp thành phần giao diện như cửa sổ, nút bấm, menu,...

- Interface là cơ chế để định nghĩa một tập hợp các phương thức trừu tượng mà các lớp có thể thực hiện. Được sử dụng để đạt được tính đa kế thừa và trừu tượng.
- Interface name {}

JAVA I/O

- Java cung cấp hai loại luồng I/O:
 - + Byte stream: các luồng được thiết kế để xử lý dữ liệu dạng byte (8 bit), dùng để đọc và ghi dữ liệu nhị phân, hình ảnh, video, âm thanh.
 - + Dòng ký tự trong Java được thiết kế để xử lý dữ liệu ký tự (thay vì dữ liệu nhị phân như dòng byte). Điều này có nghĩa là khi bạn làm việc với dữ liệu dạng văn bản (như .txt, .html, v.v.), bạn sẽ sử dụng dòng ký tự thay vì dòng byte.
- Đọc tệp văn bản bằng FileInputStream:

```
import java.io.FileInputStream;

import java.io.IOException;

public class ByteStreamExample {

    public static void main(String[] args) {

        // Đường dẫn tệp cần đọc

        String filePath = "example.txt";

        try (FileInputStream fis = new FileInputStream(filePath)) {

            int byteData;

            // Đọc từng byte từ tệp

            while ((byteData = fis.read()) != -1) {

                // Chuyển đổi byte thành ký tự và in ra

                System.out.print((char) byteData);

            }

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

}
```

```
}

}

- Đọc tệp bằng FileReader:

import java.io.FileReader;

import java.io.IOException;

public class CharacterStreamExample {

    public static void main(String[] args) {

        // Đường dẫn tệp cần đọc

        String filePath = "example.txt";

        try (FileReader fr = new FileReader(filePath)) {

            int charData;

            // Đọc từng ký tự từ tệp

            while ((charData = fr.read()) != -1) {

                // In ra ký tự đã đọc

                System.out.print((char) charData);

            }

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

}

- Đọc tệp văn bản bằng BufferedReader:

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;
```

```

public class BufferedReaderExample {

    public static void main(String[] args) {

        // Đường dẫn tệp cần đọc

        String filePath = "example.txt";


        try (BufferedReader br = new BufferedReader(new FileReader(filePath))) {

            String line;


            // Đọc từng dòng từ tệp

            while ((line = br.readLine()) != null) {

                // In ra từng dòng đã đọc

                System.out.println(line);

            }

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

}

```

- Đọc file nhị phân

```

import java.io.FileInputStream;

import java.io.IOException;

```

```

public class BinaryFileReader {

    public static void main(String[] args) {

        // Đường dẫn tệp cần đọc

        String filePath = "example.bin"; // Đảm bảo tệp là nhị phân
    }

}

```

```

try (FileInputStream fis = new FileInputStream(filePath)) {

    int byteData;


    // Đọc từng byte từ tệp

    while ((byteData = fis.read()) != -1) {

        // Xử lý byte (in ra dưới dạng số thập phân)

        System.out.print(byteData + " ");

    }

} catch (IOException e) {

    e.printStackTrace();

}

}

```

COLLECTION FRAMEWORK

- Generic trong Java là một tính năng cho phép bạn tạo các lớp, giao diện và phương thức có thể hoạt động với các loại dữ liệu khác nhau mà không cần phải xác định kiểu dữ liệu cụ thể. Điều này giúp tăng tính tái sử dụng mã và kiểm tra kiểu dữ liệu tại thời điểm biên dịch, thay vì lúc chạy.

```

class Box<T> {

    private T value;


    public void setValue(T value) {

        this.value = value;

    }


    public T getValue() {

        return value;

    }

}

```

```

public class GenericExample {

    public static void main(String[] args) {

        Box<Integer> intBox = new Box<>();

        intBox.setValue(10);

        System.out.println("Integer value: " + intBox.getValue());


        Box<String> strBox = new Box<>();

        strBox.setValue("Hello, Generic!");

        System.out.println("String value: " + strBox.getValue());

    }

}

public class GenericMethodExample {

    // Phương thức Generic

    public static <T> void printArray(T[] array) {

        for (T element : array) {

            System.out.println(element);

        }

    }

}

public static void main(String[] args) {

    Integer[] intArray = {1, 2, 3, 4};

    String[] strArray = {"Apple", "Banana", "Cherry"};


    printArray(intArray); // In mảng số nguyên

    printArray(strArray); // In mảng chuỗi

}

}

```

- Array List

```
import java.util.ArrayList;
```

```

public class ArrayListExample {

    public static void main(String[] args) {

        // Khởi tạo ArrayList

        ArrayList<String> list = new ArrayList<>();


        // Thêm phần tử vào ArrayList

        list.add("Apple");

        list.add("Banana");

        list.add("Cherry");

        list.add("Date");


        // Thêm phần tử vào vị trí cụ thể

        list.add(1, "Orange");


        // Truy cập phần tử trong ArrayList

        System.out.println("First item: " + list.get(0)); // Apple


        // Lặp qua các phần tử trong ArrayList

        System.out.println("List items:");

        for (String item : list) {

            System.out.println(item);

        }


        // Sửa đổi phần tử

        list.set(2, "Grapes");
    }

}

```

```

// Xóa phần tử
list.remove("Banana");

list.remove(0); // Xóa phần tử tại vị trí 0


// Kiểm tra sự tồn tại của phần tử
if (list.contains("Grapes")) {

    System.out.println("List contains Grapes.");
}


// Kích thước của ArrayList
System.out.println("List size: " + list.size());


// Kiểm tra ArrayList có rỗng không
if (list.isEmpty()) {

    System.out.println("The list is empty.");
} else {

    System.out.println("The list is not empty.");
}


// In danh sách sau khi thay đổi
System.out.println("Updated List: " + list);
}

- LocalDateTime:
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

```

```

public class LocalDateTimeExample {

    public static void main(String[] args) {

        // Lấy thời gian hiện tại

        LocalDateTime now = LocalDateTime.now();

        System.out.println("Current date and time: " + now);


        // Định dạng ngày giờ

        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

        String formattedDate = now.format(formatter);

        System.out.println("Formatted date and time: " + formattedDate);


        // Tính toán với LocalDateTime

        LocalDateTime nextWeek = now.plusWeeks(1);

        System.out.println("One week later: " + nextWeek);


        // Trừ đi một tháng

        LocalDateTime lastMonth = now.minusMonths(1);

        System.out.println("One month ago: " + lastMonth);


        // Kiểm tra ngày trong tuần

        System.out.println("Day of the week: " + now.getDayOfWeek());


        // So sánh hai LocalDateTime

        LocalDateTime anotherDate = LocalDateTime.of(2024, 12, 31, 12, 0, 0, 0);

        if (now.isBefore(anotherDate)) {

            System.out.println("Current date is before December 31, 2024.");
        }
    }
}

```

```
// Tạo một LocalDateTime từ chuỗi

String dateString = "2024-12-31 10:00:00";

LocalDateTime parsedDate = LocalDateTime.parse(dateString, formatter);

System.out.println("Parsed date and time: " + parsedDate);

}

}

- Scanner

import java.util.Scanner;

public class ScannerExample {

    public static void main(String[] args) {

        // Khởi tạo Scanner để đọc từ bàn phím

        Scanner scanner = new Scanner(System.in);

        // Đọc một chuỗi

        System.out.print("Enter your name: ");

        String name = scanner.nextLine();

        System.out.println("Hello, " + name + "!");

        // Đọc một số nguyên

        System.out.print("Enter your age: ");

        int age = scanner.nextInt();

        System.out.println("You are " + age + " years old.");

        // Đọc một số thực

        System.out.print("Enter your height: ");

        double height = scanner.nextDouble();
```

```
System.out.println("Your height is " + height + " meters.");

// Đọc một ký tự

System.out.print("Enter a letter: ");

char letter = scanner.next().charAt(0); // Lấy ký tự đầu tiên của chuỗi

System.out.println("You entered the letter: " + letter);

// Đọc một dòng trống sau khi đọc số

scanner.nextLine(); // Đọc bỏ dòng mới còn lại sau khi đọc số

System.out.print("Enter your address: ");

String address = scanner.nextLine();

System.out.println("Your address is: " + address);

// Đóng Scanner

scanner.close();

}

}
```

SWING

- Một giao diện swing sẽ có 2 cái chính là components (label, textfield và button) và containers (frame, panel, ...).

JButton	Tạo một nút bấm mà người dùng có thể nhấp vào.
JCheckBox	Tạo một hộp kiểm để bật/tắt tùy chọn.
JCheckBoxMenuItem	Một mục menu với chức năng giống như hộp kiểm.
JColorChooser	Hộp thoại để chọn màu sắc.
JComboBox	Tạo một hộp thả xuống để chọn một mục trong danh sách.
JComponent	Lớp cơ sở cho tất cả các thành phần Swing.
JDesktopPane	Cung cấp môi trường để quản lý các cửa sổ nội bộ.
JDialog	Tạo một hộp thoại, thường để hiển thị thông báo hoặc nhập dữ liệu.

JEditorPane	Một trình soạn thảo văn bản hỗ trợ HTML hoặc RTF.
JFileChooser	Hộp thoại để chọn tệp hoặc thư mục từ hệ thống tệp
JFormattedTextField	Trường nhập văn bản với định dạng dữ liệu cụ thể (số, ngày tháng, v.v.)
JFrame	Cửa sổ chính trong ứng dụng Swing
JInternalFrame	Một cửa sổ nội bộ nằm trong JDesktopPane.
JLabel	Hiển thị văn bản hoặc hình ảnh tĩnh
JLayeredPane	Cho phép xếp chồng các thành phần Swing theo thứ tự lớp.
JList	Hiển thị danh sách các mục.
JMenu	Tạo menu trên thanh menu.
JMenuBar	Thanh menu chứa các menu con.
JMenuItem	Một mục menu có thể chọn trong JMenu.
JOptionPane	Hiển thị hộp thoại để nhập hoặc hiển thị thông báo.
JPanel	Một container đơn giản để tổ chức các thành phần khác.
JPasswordField	Trường nhập văn bản ẩn cho mật khẩu.
JPopupMenu	Menu bật lên hiển thị khi nhấp chuột phải.
JProgressBar	Hiển thị tiến trình của một tác vụ.
JRadioButton	Nút chọn cho phép chọn một trong nhiều tùy chọn.
JRadioButtonMenuItem	Mục menu với chức năng giống như JRadioButton.
JRootPane	Container gốc cho JFrame, JDialog, hoặc JApplet.
JScrollBar	Thanh cuộn ngang hoặc dọc.
JScrollPane	Container hỗ trợ cuộn cho các thành phần khác.
JSeparator	Một đường ngang hoặc dọc để tách các thành phần.
JSlider	Cho phép người dùng chọn giá trị trong một phạm vi bằng cách kéo thanh trượt.
JSpinner	Trường nhập có thể tăng/giảm giá trị.
JSplitPane	Tạo một vùng giao diện chia thành hai phần có thể điều chỉnh kích thước
JTabbedPane	Tạo giao diện tab
JTable	Hiển thị dữ liệu dưới dạng bảng.
JTextArea	Trường nhập văn bản nhiều dòng
TextField	Trường nhập văn bản một dòng.
JTextPane	Một trình soạn thảo văn bản có khả năng định dạng cao.
JToggleButton	Nút bấm có hai trạng thái (bật/tắt).
JToolBar	Thanh công cụ chứa các nút và công cụ

JToolTip	Hiển thị thông báo trợ giúp nhỏ khi di chuột qua thành phần.
JTree	Hiển thị dữ liệu dạng cây (cấu trúc phân cấp).
JViewport	Hỗ trợ hiển thị một phần nội dung lớn hơn kích thước của container.
JWindow	Cửa sổ không có thanh tiêu đề và khung viền.

- Có 2 loại container:
 - + Top-level containers: JFrame, JApplet, JWindow và JDialog: Các top-level containers không kế thừa từ lớp JComponent mà thay vào đó, chúng kế thừa từ các lớp AWT (Abstract Window Toolkit) như Component và Container. Chính vì điều này, chúng trở thành heavyweight containers, nghĩa là các top-level containers được quản lý trực tiếp bởi hệ điều hành, khác với các thành phần lightweight trong Swing như JButton hay JLabel, vốn chủ yếu dựa vào Java để xử lý giao diện.
 - + Lightweight: JPanel, JScrollPane, JRootPane Chúng thường được sử dụng để tổ chức và quản lý các nhóm thành phần liên quan, vì lightweight container có thể được chứa trong một container khác. Điều này cho phép tạo ra các nhóm con của các thành phần liên quan, được đặt trong một container bên ngoài để dễ dàng quản lý và sắp xếp giao diện.
- import javax.swing.*;

```
class SwingDemo {
    SwingDemo() {
        // Create a new JFrame container.
        JFrame jfrm = new JFrame("A Simple Swing Application");

        // Give the frame an initial size.
        jfrm.setSize(275, 100);

        // Terminate the program when the user closes the application.
        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Create a text-based label.
        JLabel jlab = new JLabel("Swing means powerful GUIs.");

        // Add the label to the content pane.
        jfrm.add(jlab);

        // Display the frame.
        jfrm.setVisible(true);
    }

    public static void main(String[] args) {
        // Create the frame on the event dispatching thread.
```



```
SwingUtilities.invokeLater(new Runnable() {
    public void run() {
        new SwingDemo();
    }
});
}
```

- BorderLayout là một layout manager sắp xếp các thành phần theo dòng, từ trái sang phải, và nếu hết không gian, các thành phần tiếp theo sẽ được đưa xuống dòng mới.

```
import javax.swing.*;
```

```
import java.awt.*;
```

```
class FlowLayoutDemo {
    public static void main(String[] args) {
        // Tạo một JFrame container.
        JFrame jfrm = new JFrame("FlowLayout Example");

        // Thiết lập FlowLayout cho JFrame.
        jfrm.setLayout(new FlowLayout());

        // Tạo một vài nút và thêm vào JFrame.
        jfrm.add(new JButton("Button 1"));
        jfrm.add(new JButton("Button 2"));
        jfrm.add(new JButton("Button 3"));

        // Thiết lập kích thước và hiển thị cửa sổ.
        jfrm.setSize(300, 100);
        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        jfrm.setVisible(true);
    }
}
```

```
}
```

jfrm.setLayout(new FlowLayout(FlowLayout.CENTER, 10, 10)); // Căn giữa, khoảng cách ngang và dọc là 10

- BorderLayout chia cửa sổ thành 5 khu vực chính: North, South, East, West, và Center. Các thành phần được thêm vào mỗi khu vực này.

```
import javax.swing.*;
```

```
import java.awt.*;
```

```
class BorderLayoutDemo {
    public static void main(String[] args) {
        // Tạo một JFrame container.
        JFrame jfrm = new JFrame("BorderLayout Example");

        // Thiết lập BorderLayout cho JFrame.
        jfrm.setLayout(new BorderLayout());

        // Thêm các thành phần vào các khu vực khác nhau của BorderLayout.
        jfrm.add(new JButton("North"), BorderLayout.NORTH);
        jfrm.add(new JButton("South"), BorderLayout.SOUTH);
        jfrm.add(new JButton("East"), BorderLayout.EAST);
        jfrm.add(new JButton("West"), BorderLayout.WEST);
        jfrm.add(new JButton("Center"), BorderLayout.CENTER);

        // Thiết lập kích thước và hiển thị cửa sổ.
        jfrm.setSize(300, 200);
        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        jfrm.setVisible(true);
    }
}
```

- Cách xử lý sự kiện:

```
import javax.swing.*;
```

```
import java.awt.event.*;
```

```
public class EventHandlingExample {  
    public static void main(String[] args) {  
        // Tạo JFrame  
        JFrame frame = new JFrame("Event Handling Example");  
        frame.setSize(300, 200);  
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
  
        // Tạo một JButton  
        JButton button = new JButton("Click Me");  
  
        // Đăng ký ActionListener cho nút  
        button.addActionListener(new ActionListener() {  
            @Override  
            public void actionPerformed(ActionEvent e) {  
                // Xử lý sự kiện khi nút được nhấn  
                JOptionPane.showMessageDialog(frame, "Button was clicked!");  
            }  
        });  
  
        // Thêm nút vào JFrame  
        frame.add(button);  
  
        // Hiển thị JFrame  
        frame.setVisible(true);  
    }  
}
```

```
}
```

```
}
```

- Sử dụng chuột để nhấn thả MouseListener:

```
import javax.swing.*;
```

```
import java.awt.event.*;
```

```
public class MouseListenerExample {  
    public static void main(String[] args) {  
        // Tạo JFrame  
        JFrame frame = new JFrame("MouseListener Example");  
        frame.setSize(400, 300);  
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
  
        // Tạo một JLabel  
        JLabel label = new JLabel("Click or move mouse over this label");  
        label.setHorizontalAlignment(SwingConstants.CENTER);  
  
        // Đăng ký MouseListener cho JLabel  
        label.addMouseListener(new MouseListener() {  
            @Override  
            public void mouseClicked(MouseEvent e) {  
                // Xử lý sự kiện nhấn chuột  
                JOptionPane.showMessageDialog(frame, "Mouse clicked at: " +  
                    e.getPoint());  
            }  
  
            @Override  
            public void mousePressed(MouseEvent e) {  
            }  
        });  
    }  
}
```

```

        // Xử lý sự kiện khi nhấn chuột
        System.out.println("Mouse pressed at: " + e.getPoint());
    }

    @Override
    public void mouseReleased(MouseEvent e) {
        // Xử lý sự kiện khi thả chuột
        System.out.println("Mouse released at: " + e.getPoint());
    }

    @Override
    public void mouseEntered(MouseEvent e) {
        // Xử lý sự kiện khi chuột di chuyển vào JLabel
        label.setText("Mouse entered");
    }

    @Override
    public void mouseExited(MouseEvent e) {
        // Xử lý sự kiện khi chuột di chuyển ra ngoài JLabel
        label.setText("Mouse exited");
    }
});

// Thêm JLabel vào JFrame
frame.add(label);

// Hiển thị JFrame
frame.setVisible(true);

```

```

    }
}

- Code mẫu menu:

import javax.swing.*;
import java.awt.event.*;

public class SimpleMenuExample {

    public static void main(String[] args) {

        // Tạo cửa sổ chính
        JFrame frame = new JFrame("Java Swing Menu Example");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 300);

        // Tạo thanh menu
        JMenuBar menuBar = new JMenuBar();

        // Tạo menu "File"
        JMenu fileMenu = new JMenu("File");
        JMenuItem newItem = new JMenuItem("New");
        JMenuItem openItem = new JMenuItem("Open");
        JMenuItem saveItem = new JMenuItem("Save");
        fileMenu.add(newItem);
        fileMenu.add(openItem);
        fileMenu.add(saveItem);

        // Tạo menu "Edit"
        JMenu editMenu = new JMenu("Edit");
        JMenuItem cutItem = new JMenuItem("Cut");
    }
}

```

```

JMenuItem copyItem = new JMenuItem("Copy");

JMenuItem pasteItem = new JMenuItem("Paste");

editMenu.add(cutItem);

editMenu.add(copyItem);

editMenu.add(pasteItem);


// Tạo menu "View"

JMenu viewMenu = new JMenu("View");

JMenuItem zoomInItem = new JMenuItem("Zoom In");

JMenuItem zoomOutItem = new JMenuItem("Zoom Out");

viewMenu.add(zoomInItem);

viewMenu.add(zoomOutItem);


// Thêm các menu vào thanh menu

menuBar.add(fileMenu);

menuBar.add(editMenu);

menuBar.add(viewMenu);


// Gán thanh menu vào cửa sổ

frame.setJMenuBar(menuBar);


// Hiển thị cửa sổ

frame.setVisible(true);
}
}

```

- Code ví dụ tạo một pop up menu:

```

import javax.swing.*;

import java.awt.event.*;

```

```

public class PopupMenuExample {

    public static void main(String[] args) {

        // Tạo cửa sổ chính

        JFrame frame = new JFrame("Java Swing Popup Menu Example");

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        frame.setSize(400, 300);


        // Tạo một JPanel

        JPanel panel = new JPanel();


        // Tạo pop-up menu

        JPopupMenu popupMenu = new JPopupMenu();


        // Thêm các menu items vào pop-up menu

        JMenuItem cutItem = new JMenuItem("Cut");

        JMenuItem copyItem = new JMenuItem("Copy");

        JMenuItem pasteItem = new JMenuItem("Paste");

        JMenuItem deleteItem = new JMenuItem("Delete");


        popupMenu.add(cutItem);

        popupMenu.add(copyItem);

        popupMenu.add(pasteItem);

        popupMenu.add(deleteItem);


        // Thêm một MouseListener để hiển thị pop-up menu khi nhấp chuột phải

        panel.addMouseListener(new MouseAdapter() {

            public void mousePressed(MouseEvent e) {

```

```

        if (e.isPopupTrigger()) {
            popupMenu.show(e.getComponent(), e.getX(), e.getY());
        }
    }

    public void mouseReleased(MouseEvent e) {
        if (e.isPopupTrigger()) {
            popupMenu.show(e.getComponent(), e.getX(), e.getY());
        }
    }
});

// Thêm panel vào frame
frame.add(panel);

// Hiển thị cửa sổ
frame.setVisible(true);
}
}

- Nhận vào đường dẫn và chọn file:

import javax.swing.*;

import java.io.File;

public class FileChooserExample {

    public static void main(String[] args) {

        // Tạo cửa sổ chính

        JFrame frame = new JFrame("Java Swing File Chooser Example");

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

```

```

        frame.setSize(400, 300);

        // Tạo nút để mở hộp thoại chọn file

        JButton openButton = new JButton("Open File");

        openButton.addActionListener(e -> {

            // Tạo đối tượng JFileChooser

            JFileChooser fileChooser = new JFileChooser();

            // Cho phép người dùng chọn file (mặc định)

            int returnValue = fileChooser.showOpenDialog(frame);

            if (returnValue == JFileChooser.APPROVE_OPTION) {

                // Lấy đường dẫn của file đã chọn

                File selectedFile = fileChooser.getSelectedFile();

                String filePath = selectedFile.getAbsolutePath();

                JOptionPane.showMessageDialog(frame, "Selected file path: " + filePath);

            }

        });

        // Thêm nút vào cửa sổ

        frame.add(openButton);

        // Hiển thị cửa sổ

        frame.setVisible(true);
    }
}

- JTextField

```

```

import javax.swing.*;

import java.awt.event.*;

public class JTextFieldExample {

    public static void main(String[] args) {

        // Tạo cửa sổ chính

        JFrame frame = new JFrame("JTextField Example");

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        frame.setSize(400, 300);


        // Tạo JTextField

        JTextField textField = new JTextField(20); // 20 là số cột hiển thị


        // Tạo nút để lấy giá trị từ JTextField

        JButton button = new JButton("Show Text");


        button.addActionListener(e -> {

            String text = textField.getText(); // Lấy giá trị từ JTextField

            JOptionPane.showMessageDialog(frame, "You entered: " + text);

        });


        // Thêm các thành phần vào cửa sổ

        JPanel panel = new JPanel();

        panel.add(textField);

        panel.add(button);

        frame.add(panel);


        // Hiển thị cửa sổ

```

```

        frame.setVisible(true);

    }

}

- JSlider

import javax.swing.*;

import java.awt.event.*;

public class JSliderExample {

    public static void main(String[] args) {

        // Tạo cửa sổ chính

        JFrame frame = new JFrame("JSlider Example");

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        frame.setSize(400, 300);


        // Tạo JSlider

        JSlider slider = new JSlider(0, 100, 50); // Phạm vi từ 0 đến 100, giá trị mặc
        định là 50

        slider.setMajorTickSpacing(10); // Khoảng cách các vạch chính

        slider.setMinorTickSpacing(1); // Khoảng cách các vạch phụ

        slider.setPaintTicks(true); // Vẽ các vạch

        slider.setPaintLabels(true); // Hiển thị nhãn


        // Tạo nút để lấy giá trị từ JSlider

        JButton button = new JButton("Show Value");


        button.addActionListener(e -> {

            int value = slider.getValue(); // Lấy giá trị của JSlider

            JOptionPane.showMessageDialog(frame, "Slider Value: " + value);

```

```

});

// Thêm các thành phần vào cửa sổ

JPanel panel = new JPanel();
panel.add(slider);
panel.add(button);
frame.add(panel);

// Hiển thị cửa sổ

frame.setVisible(true);
}
}

- JOptionPane:

import javax.swing.*;

public class JOptionPaneExample {

    public static void main(String[] args) {

        // Hiển thị một thông báo đơn giản

        JOptionPane.showMessageDialog(null, "This is a simple message.");

        // Hiển thị một thông báo với tùy chọn "Yes" và "No"

        int choice = JOptionPane.showConfirmDialog(null, "Do you want to
continue?", "Confirmation", JOptionPane.YES_NO_OPTION);

        if (choice == JOptionPane.YES_OPTION) {

            JOptionPane.showMessageDialog(null, "You selected Yes!");

        } else {

            JOptionPane.showMessageDialog(null, "You selected No!");

```

```

}

// Hiển thị hộp thoại nhập liệu

String name = JOptionPane.showInputDialog("Enter your name:");

JOptionPane.showMessageDialog(null, "Hello, " + name);

}

}

GDBC

- Java application -> JDBC API -> JDBC Driver Manager
- Các component của JDBC
    + DriverManager: Quản lý danh sách các trình điều khiển cơ sở dữ liệu.
    Connection connection =
    DriverManager.getConnection("jdbc:mysql://localhost:3306/mydb",
    "username", "password");
    + Driver: Xử lý giao tiếp giữa ứng dụng Java và máy chủ cơ sở dữ liệu.
    Driver driver = new com.mysql.cj.jdbc.Driver();
    DriverManager.registerDriver(driver);
    + Connection: thiết lập một kết nối giữa ứng dụng Java và cơ sở dữ liệu
    Connection connection =
    DriverManager.getConnection("jdbc:mysql://localhost:3306/mydb",
    "username", "password");
    + Statement: Được sử dụng để gửi các câu lệnh SQL từ ứng dụng Java đến db.
    Có ba loại chính là + Statement: dành cho câu lệnh SQL tĩnh.
        + PreparedStatement: dành cho các câu lệnh SQL có tham số
        + CallableStatement: dành cho các thủ tục lưu trữ
    Statement statement = connection.createStatement();
    String sql = "SELECT * FROM employees";
    ResultSet resultSet = statement.executeQuery(sql);
    + ResultSet: Chứa dữ liệu được truy vấn từ cơ sở dữ liệu thông qua SQL query.
    while (resultSet.next()) {
        int id = resultSet.getInt("id");
        String name = resultSet.getString("name");
        System.out.println("ID: " + id + ", Name: " + name);
    }
    + SQLException: lớp xử lý các lỗi xảy ra khi làm việc với các cơ sở dữ liệu
- Connection.close() ngắt kết nối.
- Kiểu dữ liệu của Java qua JDBC:

```

SQL	JDBC/Java	setXXX
VARCHAR	Java.lang.String	setString
CHAR	Java.lang.String	setString
LONGVARCHAR	Java.lang.String	setString
BIT	Boolean	setBoolean
NUMERIC	Java.math.BigDecimal	setBigDecimal
TINYINT	Byte	setByte
SMALLINT	Short	setShort
INTEGER	Int	setInt
BIGINT	Long	setLong
REAL	Float	setFloat
FLOAT	Float	setFloat
DOUBLE	Double	setDouble
VARBINARY	Byte[]	setBytes
BINARY	Byte[]	setBytes
DATE	Java.sql.Date	setDate
TIME	Java.sql.Time	setTime
TIMESTAMP	Java.sql.Timestamp	setTimestamp
CLOB	Java.sql.Clob	setClob
BLOB	Java.sql.Blob	setBlob
ARRAY	Java.sql.Array	setARRAY
REF	Java.sql.Ref	setRef
STRUCT	Java.sql.Struct	setStruct

- Rs.wasNull() to check null

MULTITHREADING

- Luồng là đơn vị nhỏ nhất của mã có thể thực thi trong một chương trình.
- Đa luồng cho phép một chương trình chia nhỏ thành nhiều luồng, mỗi luồng thực hiện một nhiệm vụ riêng biệt cùng lúc.
- Khác biệt giữa tiến trình và luồng:
 - + Tiến trình là chương trình độc lập có vùng nhớ riêng. Mỗi tiến trình chạy tách biệt với các tiến trình khác.
 - + Luồng chạy bên trong một tiến trình và chia sẻ vùng nhớ cũng như tài nguyên với các luồng khác trong cùng tiến trình.
- Java hỗ trợ lập trình đa luồng thông qua lớp java.lang.Thread và Runnable.
- Ưu điểm của Multithreading:
 - + Tận dụng thời gian nhàn rỗi trong chương trình khi đang chờ gửi/nhận dữ liệu từ thiết bị để thực hiện tác vụ khác.
 - + Hỗ trợ trên mọi hệ thống. Với hệ thống đơn nhân các luồng chia sẻ CPU và luân phiên nhận lát thời gian để chạy (không thực sự đồng thời). Hệ thống đa nhân thì 2 hoặc nhiều luồng có thể thực thi đồng thời trên các nhân khác nhau.
- ➔ Tối ưu hóa tài nguyên và tăng hiệu suất ở trên mọi hệ thống,
- Thread có thể tồn tại ở các trạng thái:
 - + Running: đang chạy, đang thực thi các lệnh của nó.
 - + Suspended: bị tạm dừng thực thi.

- + Resumed: sau khi bị tạm dừng thì tiếp tục thực thi.
- + Blocked: bị chặn khi đang chờ tài nguyên hoặc điều kiện nào đó.
- + Terminated: đã hoàn thành nhiệm vụ hoặc bị hủy và không còn hoạt động.

- Để chạy thì implement runnable interface
 class MyRunnable implements Runnable {
 @Override
 public void run() {
 for (int i = 1; i <= 5; i++) {
 System.out.println("Runnable Thread: " + i);
 try {
 Thread.sleep(500); // Tạm dừng 500ms
 } catch (InterruptedException e) {
 System.out.println("Thread bị gián đoạn.");
 }
 }
 }
 }

 public class RunnableExample {
 public static void main(String[] args) {
 MyRunnable runnable = new MyRunnable();
 Thread thread = new Thread(runnable); // Tạo thread từ Runnable
 thread.start(); // Bắt đầu thread
 System.out.println("Main thread đang chạy song song.");
 }
 }

 - Hoặc tạo lớp kế thừa Thread:
 class MyThread extends Thread {
 @Override
 public void run() {
 for (int i = 1; i <= 5; i++) {
 System.out.println("Thread Class: " + i);
 try {
 Thread.sleep(500); // Tạm dừng 500ms
 } catch (InterruptedException e) {
 System.out.println("Thread bị gián đoạn.");
 }
 }
 }
 }

 public class ThreadExample {
 public static void main(String[] args) {
 MyThread thread = new MyThread(); // Tạo object của MyThread


```

        thread.start(); // Bắt đầu thread
        System.out.println("Main thread đang chạy song song.");
    }
}
- Thread.sleep(milliseconds) dùng để tạm dừng thread hiện tại trong khoảng thời gian nhất định.
- thread.isAlive(): kiểm tra xem thread còn hoạt động không.
- Thread.join(): đợi thread hoàn thành.
- Đoạn code xem độ ưu tiên của các thread

```

```

class Priority implements Runnable {

    int count; // Biến đếm số lần lặp

    Thread thrd; // Đối tượng thread

    static boolean stop = false; // Biến dùng để dừng các thread

    static String currentName; // Lưu tên của thread hiện tại đang chạy

```

// Constructor

```

Priority(String name) {

    thrd = new Thread(this, name);

    count = 0; // Khởi tạo biến đếm

    currentName = name; // Gán tên cho thread

}

```

@Override

```

public void run() {

    System.out.println(thrd.getName() + " bắt đầu.");

    do {

        count++; // Tăng biến đếm

        // Dừng thread nếu một thread khác hoàn thành

        if (!currentName.equals(thrd.getName())) {

            stop = true;

        }

    }

```

```

    } while (!stop && count < 10000000); // Lặp cho đến khi stop = true hoặc đạt giới hạn

```

```

        stop = true;

```

```

        System.out.println("\n" + thrd.getName() + " kết thúc.");

```

```

    }

}

```

```

public class PriorityDemo {

```

```

    public static void main(String[] args) {

```

```

        // Tạo các thread với các mức độ ưu tiên khác nhau

```

```

        Priority mt1 = new Priority("High Priority");

```

```

        Priority mt2 = new Priority("Low Priority");

```

```

        Priority mt3 = new Priority("Normal Priority #1");

```

```

        Priority mt4 = new Priority("Normal Priority #2");

```

```

        Priority mt5 = new Priority("Normal Priority #3");

```

```

        // Thiết lập độ ưu tiên

```

```

        mt1.thrd.setPriority(Thread.NORM_PRIORITY + 2); // Cao hơn bình thường

```

```

        mt2.thrd.setPriority(Thread.NORM_PRIORITY - 2); // Thấp hơn bình thường

```

```

        // Khởi động các thread

```

```

        mt1.thrd.start();

```

```

        mt2.thrd.start();

```

```

        mt3.thrd.start();

```

```

        mt4.thrd.start();

```

```

        mt5.thrd.start();

```

```

        // Đợi tất cả các thread kết thúc

```

```

        try {

```

```

            mt1.thrd.join();

```

```

            mt2.thrd.join();

```

```

            mt3.thrd.join();

```

```

        mt4.thrd.join();

        mt5.thrd.join();
    } catch (InterruptedException e) {

        System.out.println("Main thread bị gián đoạn.");
    }

    // In kết quả số lần đếm của mỗi thread

    System.out.println("\nHigh priority thread counted to " + mt1.count);
    System.out.println("Low priority thread counted to " + mt2.count);
    System.out.println("1st Normal priority thread counted to " + mt3.count);
    System.out.println("2nd Normal priority thread counted to " + mt4.count);
    System.out.println("3rd Normal priority thread counted to " + mt5.count);
}

}

- Sử dụng synchronization để đảm bảo rằng các thread không truy cập đồng thời vào một tài nguyên dùng chung dẫn đến xung đột dữ liệu.
- Cách thực hiện:
    + Dùng từ khóa synchronized trước phương thức
    + Khi một thread vào phương thức synchronized, nó khóa đối tượng và các thread khác phải chờ cho đến khi đối tượng được mở khóa.
- Thread communication: một thread có thể tạm thời chờ khi không thể tiếp tục xử lý thay vì chiếm tài nguyên vô ích, sử dụng phương pháp inter-thread communication để quản lý luồng.
- Các phương thức:
    + wait(): tạm thời dừng thread hiện tại giải phóng khóa
    + notify(): đánh thức một thread đang chờ trên đối tượng
    + notifyAll(): đánh thức tất cả các thread đang chờ
- Ví dụ về đồng bộ hóa 2 thread để in lần lượt tick và tock

class TickTock {

    String state;

    synchronized void tick(boolean running) {

        if (!running) { state = "ticked"; notify(); return; }

        System.out.print("Tick ");

        state = "ticked"; notify();
    }
}

```

```

    try { while (!state.equals("tocked")) wait(); }

    catch (InterruptedException exc) {}

}

synchronized void tock(boolean running) {

    if (!running) { state = "tocked"; notify(); return; }

    System.out.println("Tock");

    state = "tocked"; notify();

    try { while (!state.equals("ticked")) wait(); }

    catch (InterruptedException exc) {}

}

}

- Hai thread được đồng bộ để in lần lượt:

class ThreadCom {

    public static void main(String args[]) {

        TickTock tt = new TickTock();

        MyThread6 mt1 = new MyThread6("Tick", tt);

        MyThread6 mt2 = new MyThread6("Tock", tt);

        try {

            mt1.thrd.join();

            mt2.thrd.join();

        } catch (InterruptedException exc) {}

    }

}

- Các cách ngăn chặn deadlock:

    + Sử dụng thứ tự nhất quán khi khóa tài nguyên:

```

```

class ResourceLock {

    static final Object lock1 = new Object();

    static final Object lock2 = new Object();
}

```

```

}

class Thread1 extends Thread {

    public void run() {

        synchronized (ResourceLock.lock1) {

            System.out.println("Thread1 locked lock1");

            synchronized (ResourceLock.lock2) {

                System.out.println("Thread1 locked lock2");

            }

        }

    }

}

class Thread2 extends Thread {

    public void run() {

        synchronized (ResourceLock.lock1) { // Sử dụng thứ tự giống Thread1

            System.out.println("Thread2 locked lock1");

            synchronized (ResourceLock.lock2) {

                System.out.println("Thread2 locked lock2");

            }

        }

    }

}

public class DeadlockPrevention {

    public static void main(String[] args) {

        Thread1 t1 = new Thread1();

        Thread2 t2 = new Thread2();

        t1.start();

        t2.start();

    }

}

```

```

}

- Dùng tryLock với thời gian chờ:

import java.util.concurrent.locks.Lock;

import java.util.concurrent.locks.ReentrantLock;

class TryLockExample {

    private final Lock lock1 = new ReentrantLock();

    private final Lock lock2 = new ReentrantLock();

    public void method1() {

        try {

            if (lock1.tryLock() && lock2.tryLock()) {

                try {

                    System.out.println("Thread1 acquired both locks");

                } finally {

                    lock1.unlock();

                    lock2.unlock();

                }

            } else {

                System.out.println("Thread1 couldn't acquire locks, avoiding deadlock");

            }

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

    public void method2() {

        try {

```

```

if (lock2.tryLock() && lock1.tryLock()) {
    try {
        System.out.println("Thread2 acquired both locks");
    } finally {
        lock2.unlock();
        lock1.unlock();
    }
} else {
    System.out.println("Thread2 couldn't acquire locks, avoiding deadlock");
}
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

```

public class DeadlockAvoidance {
    public static void main(String[] args) {
        TryLockExample example = new TryLockExample();

        Thread t1 = new Thread(example::method1);
        Thread t2 = new Thread(example::method2);

        t1.start();
        t2.start();
    }
}

```

- Sử dụng một thread để quản lý tài nguyên

- Tránh giữ nhiều khóa cùng lúc
- Sử dụng Deadlock Detection
- Áp dụng nguyên lý tránh điều kiện cần của Deadlock:
 - + Mutual Exclusion: một tài nguyên chỉ được dùng bởi một thread tại một thời điểm.
 - + Hold and Wait: một thread giữ tài nguyên và chờ tài nguyên khác
 - + Không thể ép buộc giải phóng tài nguyên
 - + Các thread chờ lẫn nhau theo vòng tròn.
- Trong ứng dụng Swing điển hình sẽ có 3 luồng chính:
 - + Luồng chính dùng để chạy hàm main()
 - + Luồng phân phối sự kiện (event dispatching thread) dùng để xử lý các sự kiện của Swing.
 - + Phần công việc.

NETWORKING

- Socket xác định một điểm cuối trong mạng: socket là một giao diện phần mềm giúp kết nối các máy tính trong mạng, là điểm cuối cho việc truyền nhận dữ liệu giữa các hệ thống.
- Điều này được thực hiện qua việc sử dụng một cổng, là một socket có số hiệu trên một máy tính cụ thể.
- Một tiến trình Server được nói là lắng nghe một cổng cho đến khi có client kết nối. Server sẽ chờ các kết nối từ client qua cổng đã chỉ định. Khi một kết nối đến, Server sẽ chấp nhận và thiết lập kết nối với client đó.
- Một server có thể chấp nhận nhiều client kết nối vào cùng một số cổng, mặc dù mỗi phiên làm việc là duy nhất: Mặc dù các client có thể kết nối vào cùng một cổng trên server, mỗi kết nối sẽ được quản lý riêng biệt nhờ các thông tin như địa chỉ IP và số cổng của client. Mỗi kết nối tạo thành một "phiên làm việc" riêng biệt.
- Để quản lý nhiều kết nối của client, server phải sử dụng đa luồng hoặc một phương pháp khác để đa phân (multiplexing) I/O đồng thời: Khi có nhiều client kết nối, server cần sử dụng kỹ thuật đa luồng (multithreading) hoặc một phương pháp khác để xử lý đồng thời các yêu cầu từ các client mà không làm gián đoạn hoạt động của nhau.
- Giao tiếp qua socket diễn ra thông qua một giao thức: Socket giao tiếp sử dụng các giao thức mạng như TCP hoặc UDP. Giao thức xác định cách thức truyền tải dữ liệu giữa các hệ thống, đảm bảo rằng dữ liệu được gửi đi và nhận về đúng cách.
 - + 21 là cho FTP
 - + 23 là cho Telnet
 - + 25 là cho email
 - + 43 là cho whois
 - + 80 là cho HTTP
 - + 119 là cho netnews
- Ví dụ:

```
import java.net.*;
```

```
class InetAddressTest {  
  
    public static void main(String args[]) throws UnknownHostException {  
  
        // Lấy địa chỉ IP của máy tính hiện tại (localhost)  
  
        InetAddress address = InetAddress.getLocalHost();  
  
        System.out.println(address);  
  
  
  
        // Lấy địa chỉ IP của tên miền "www.vnexpress.net"  
  
        address = InetAddress.getByName("www.vnexpress.net");  
  
        System.out.println(address);  
  
  
  
        // Lấy tất cả các địa chỉ IP của tên miền "www.hcmus.edu.vn"  
  
        InetAddress SW[] = InetAddress.getAllByName("www.hcmus.edu.vn");  
  
        for (int i = 0; i < SW.length; i++) {  
  
            System.out.println(SW[i]);  
  
        }  
  
    }  
}
```

- Tìm và in ra địa chỉ IP của một máy tính hoặc trang web
- Các phương thức cơ bản:

boolean equals(Object other)	Trả về true nếu đối tượng này có cùng địa chỉ Internet với đối tượng other. Nếu không, trả về false.
byte[] getAddress()	Trả về một mảng byte biểu diễn địa chỉ IP của đối tượng trong thứ tự byte của mạng (network byte order).
String getHostAddress()	Trả về một chuỗi biểu diễn địa chỉ host (địa chỉ IP) liên kết với đối tượng InetAddress.
String getHostName()	Trả về một chuỗi biểu diễn tên host (hostname) liên kết với đối tượng InetAddress.
boolean isMulticastAddress()	Trả về true nếu địa chỉ này là địa chỉ multicast. Nếu không, trả về false.

String toString()	Trả về một chuỗi liệt kê tên host và địa chỉ IP để thuận tiện cho việc hiển thị.
-------------------	----------------------------------------------------------------------------------

- Có 2 loại TCP sockets:
 - + ServerSocket: dành cho máy chủ Server. Nó hoạt động như một listener chờ các kết nối từ client trước khi thực hiện bất kì hành động nào.
 - + Socket: dành cho client. Nó kết nối tới ServerSocket và bắt đầu các giao thức trao đổi dữ liệu.
- Lớp socket được thiết kế để kết nối với các socket của Server và bắt đầu giao tiếp. Khi được tạo nó ngầm định thiết lập kết nối giữa client và Server.
- Các phương thức của lớp socket:

InetAddress getAddress()	Lấy địa chỉ của server mà socket kết nối.
InetAddress getLocalAddress()	Lấy địa chỉ IP cục bộ (local) của socket.
int getPort()	Lấy cổng (port) mà socket đang kết nối.
int getLocalPort()	Lấy cổng (port) cục bộ mà socket sử dụng.
InputStream getInputStream()	Lấy luồng nhập (input stream) của socket để đọc dữ liệu từ server.
OutputStream getOutputStream()	Lấy luồng xuất (output stream) của socket để gửi dữ liệu tới server.
void close()	Đóng socket, chấm dứt kết nối.
boolean isClosed()	Kiểm tra xem socket đã bị đóng chưa.
boolean isConnected()	Kiểm tra xem socket có đang kết nối không.
void setSoTimeout(int timeout)	Thiết lập thời gian chờ (timeout) khi đọc dữ liệu.
int getSoTimeout()	Lấy giá trị thời gian chờ hiện tại.
void setReuseAddress(boolean on)	Cho phép tái sử dụng địa chỉ socket.
boolean getReuseAddress()	Kiểm tra xem socket có cho phép tái sử dụng địa chỉ không.
void setTcpNoDelay(boolean on)	Bật hoặc tắt chế độ không trì hoãn TCP (TCP_NODELAY).
boolean getTcpNoDelay()	Kiểm tra trạng thái của TCP_NODELAY.
void setKeepAlive(boolean on)	Bật hoặc tắt chế độ giữ kết nối (Keep-Alive).
boolean getKeepAlive()	Kiểm tra trạng thái Keep-Alive.

- Các constructor của lớp Socket

Socket(String host, int port)	Kết nối tới một server với địa chỉ host và cổng port.
Socket(InetAddress address, int port)	Kết nối tới một server với đối tượng InetAddress và cổng port.
Socket(String host, int port, InetAddress localAddr, int localPort)	Tạo một socket với địa chỉ cục bộ (local address) và cổng cụ thể.
Socket(InetAddress address, int port, InetAddress localAddr, int localPort)	Tương tự, nhưng sử dụng đối tượng InetAddress.

```

import java.io.InputStream;

import java.io.OutputStream;

import java.net.InetAddress;

import java.net.Socket;


public class SocketExample {

    public static void main(String[] args) {

        String serverAddress = "example.com"; // Địa chỉ server

        int port = 80; // Cổng của server (HTTP)


        try {

            // Tạo một socket và kết nối tới server

            Socket socket = new Socket(serverAddress, port);


            // Lấy thông tin về kết nối

            System.out.println("Connected to server: " + socket.getInetAddress());

            System.out.println("Server port: " + socket.getPort());

            System.out.println("Local address: " + socket.getLocalAddress());

            System.out.println("Local port: " + socket.getLocalPort());


            // Kiểm tra trạng thái kết nối

            System.out.println("Is connected: " + socket.isConnected());

            System.out.println("Is closed: " + socket.isClosed());


            // Giao tiếp với server

            InputStream inputStream = socket.getInputStream();

            OutputStream outputStream = socket.getOutputStream();

```

```

// Gửi một yêu cầu HTTP đơn giản tới server

String request = "GET / HTTP/1.1\r\nHost: " + serverAddress + "\r\n\r\n";

outputStream.write(request.getBytes());

outputStream.flush();


// Đọc phản hồi từ server

byte[] buffer = new byte[1024];

int bytesRead = inputStream.read(buffer);

System.out.println("Server response: ");

System.out.println(new String(buffer, 0, bytesRead));


// Cấu hình socket

socket.setSoTimeout(5000); // Đặt thời gian chờ (timeout) là 5 giây

System.out.println("Socket timeout: " + socket.getSoTimeout());


socket.setKeepAlive(true); // Bật chế độ Keep-Alive

System.out.println("Keep-Alive enabled: " + socket.getKeepAlive());


// Đóng socket

socket.close();

System.out.println("Socket closed: " + socket.isClosed());

} catch (Exception e) {

    e.printStackTrace();

}

}

}

- Cách tạo một HttpURLConnection

URL url = new URL("http://www.example.com");

```

URLConnection httpCon = (URLConnection) url.openConnection();

getContentType()	Trả về kiểu nội dung (Content-Type) của phản hồi từ máy chủ.
getDate()	Trả về thời gian mà tài nguyên được tạo hoặc thay đổi lần đầu (nếu có).
getExpiration()	Trả về thời gian hết hạn của tài nguyên (nếu có).
getLastModified()	Trả về thời gian tài nguyên được sửa đổi lần cuối.
getContentLengthLong()	Trả về độ dài của nội dung (nếu có).
setRequestMethod(String method)	Đặt phương thức HTTP (GET, POST, PUT, DELETE, etc.).
getRequestMethod()	Lấy phương thức HTTP hiện tại.
getResponseCode()	Trả về mã trạng thái HTTP (200, 404, 500, etc.).
getResponseMessage()	Trả về thông báo trạng thái HTTP (OK, Not Found, etc.).
setDoOutput(boolean dooutput)	Cho phép gửi dữ liệu tới máy chủ (áp dụng với các phương thức như POST).
getInputStream()	Lấy luồng đầu vào để đọc dữ liệu từ phản hồi của máy chủ.
getErrorStream()	Lấy luồng lỗi từ máy chủ (nếu có lỗi xảy ra).
disconnect()	Ngắt kết nối với máy chủ.
setConnectTimeout(int timeout)	Đặt thời gian chờ kết nối.
setReadTimeout(int timeout)	Đặt thời gian chờ đọc dữ liệu từ máy chủ.

- Minh họa:

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

```
import java.io.InputStreamReader;
```

```
import java.net.HttpURLConnection;
```

```
import java.net.URL;
```

```
public class HttpURLConnectionExample {
```

```
    public static void main(String[] args) {
```

```
        String url = "http://www.example.com";
```

```
        try {
```

```
            // Tạo đối tượng URL
```

```
            URL obj = new URL(url);
```

```
        // Mở kết nối
```

```
        HttpURLConnection httpCon = (HttpURLConnection) obj.openConnection();
```

```
        // Thiết lập phương thức yêu cầu
```

```
        httpCon.setRequestMethod("GET");
```

```
        // Lấy mã trạng thái HTTP
```

```
        int responseCode = httpCon.getResponseCode();
```

```
        System.out.println("Response Code: " + responseCode);
```

```
        // Đọc phản hồi từ máy chủ
```

```
        BufferedReader in = new BufferedReader(new  
        InputStreamReader(httpCon.getInputStream()));
```

```
        String inputLine;
```

```
        StringBuilder response = new StringBuilder();
```

```
        while ((inputLine = in.readLine()) != null) {
```

```
            response.append(inputLine);
```

```
        }
```

```
        in.close();
```

```
        // In ra phản hồi
```

```
        System.out.println("Response: " + response.toString());
```

```
        // Đóng kết nối
```

```
        httpCon.disconnect();
```

```
    } catch (Exception e) {
```

```
        e.printStackTrace();
```

```
}  
}  
}
```

- Datagrams là một cách khác để truyền dữ liệu giữa các máy tính, sử dụng giao thức UDP (User Datagram Protocol). TCP đảm bảo dữ liệu được gửi đi một cách đáng tin cậy, theo thứ tự, và sử dụng các thuật toán phức tạp để kiểm soát tắc nghẽn. UDP (và datagrams) không đảm bảo dữ liệu sẽ đến đích hoặc sẽ đến đúng thứ tự. Điều này làm cho nó nhanh hơn nhưng kém tin cậy hơn so với TCP.
- Phù hợp với các ứng dụng cần truyền dữ liệu nhanh và không yêu cầu độ tin cậy cao, như streaming video, game online, hoặc DNS queries.
- Chương trình ví dụ gửi dữ liệu:

```
import java.net.DatagramPacket;
```

```
import java.net.DatagramSocket;
```

```
import java.net.InetAddress;
```

```
public class DatagramSender {  
    public static void main(String[] args) {  
        try {  
            // Tạo socket  
            DatagramSocket socket = new DatagramSocket();  
  
            // Chuẩn bị dữ liệu  
            String message = "Hello, Datagram!";  
            byte[] buffer = message.getBytes();  
  
            // Tạo gói tin DatagramPacket  
            InetAddress receiverAddress = InetAddress.getByName("localhost");  
            int port = 9876;  
            DatagramPacket packet = new DatagramPacket(buffer, buffer.length,  
                receiverAddress, port);
```

```
// Gửi gói tin
```

```
socket.send(packet);
```

```
System.out.println("Message sent: " + message);
```

```
// Đóng socket
```

```
socket.close();
```

```
} catch (Exception e) {
```

```
    e.printStackTrace();
```

```
}
```

```
}
```

- Chương trình nhận dữ liệu:

```
import java.net.DatagramPacket;
```

```
import java.net.DatagramSocket;
```

```
public class DatagramReceiver {
```

```
    public static void main(String[] args) {
```

```
        try {
```

```
            // Tạo socket và gán vào cổng 9876
```

```
            DatagramSocket socket = new DatagramSocket(9876);
```

```
            // Chuẩn bị buffer để nhận dữ liệu
```

```
            byte[] buffer = new byte[1024];
```

```
            DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
```

```
            // Nhận gói tin
```

```
            System.out.println("Waiting for a message...");
```



```

socket.receive(packet);

// Hiển thị nội dung gói tin

String receivedMessage = new String(packet.getData(), 0,
packet.getLength());

System.out.println("Message received: " + receivedMessage);


// Đóng socket

socket.close();
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

Bài tập sao chép file và hiển thị tiến trình

```

package org;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.io.*;
import java.text.DecimalFormat;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class FileCopyApp extends JFrame {

    private final JTextField sourceField; // Trường văn bản để nhập đường dẫn file
    nguồn

```

```

        private final JTextField destinationField; // Trường văn bản để nhập đường dẫn
        thư mục đích

        private final JButton startButton; // Nút để bắt đầu quá trình sao chép

        private final JProgressBar progressBar; // Thanh tiến trình để hiển thị quá trình
        sao chép

        private final JLabel statusLabel; // Nhãn hiển thị trạng thái sao chép


    public FileCopyApp() {

        setTitle("File Copy Application"); // Tiêu đề của cửa sổ

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // Đóng ứng dụng khi
        cửa sổ bị đóng

        setSize(400, 200); // Kích thước cửa sổ

        setLayout(new GridLayout(4, 1, 10, 10)); // Bố trí các thành phần trong cửa sổ


        // Panel chứa các trường nhập file nguồn và thư mục đích

        JPanel inputPanel = new JPanel(new GridLayout(2, 3, 5, 5));

        sourceField = new JTextField(); // Trường nhập đường dẫn file nguồn

        destinationField = new JTextField(); // Trường nhập đường dẫn thư mục đích

        JButton browseSourceButton = new JButton("Browse..."); // Nút chọn file
        nguồn

        JButton browseDestinationButton = new JButton("Browse..."); // Nút chọn thư
        mục đích


        // Thêm các thành phần vào panel

        inputPanel.add(new JLabel("Source File:"));

        inputPanel.add(sourceField);

        inputPanel.add(browseSourceButton);

        inputPanel.add(new JLabel("Destination Folder:"));

        inputPanel.add(destinationField);

        inputPanel.add(browseDestinationButton);

```

```

// Panel chứa thanh tiến trình và nhãn trạng thái
JPanel progressPanel = new JPanel(new GridLayout(2, 1, 5, 5));
progressBar = new JProgressBar(0, 100); // Khởi tạo thanh tiến trình
progressBar.setStringPainted(true); // Hiển thị phần trăm trên thanh tiến trình
statusLabel = new JLabel("Copy Progress: 0 KB / 0 KB", JLabel.CENTER); // Nhãn trạng thái

// Thêm các thành phần vào panel
progressPanel.add(progressBar);
progressPanel.add(statusLabel);

// Panel chứa nút bắt đầu và nút reset
JPanel buttonPanel = new JPanel(new FlowLayout());
startButton = new JButton("Start Copy"); // Nút bắt đầu sao chép
JButton resetButton = new JButton("Reset"); // Nút reset

// Thêm các nút vào panel
buttonPanel.add(startButton);
buttonPanel.add(resetButton);

// Thêm các panel vào cửa sổ chính
add(inputPanel);
add(progressPanel);
add(buttonPanel);

// Thêm các sự kiện cho các nút
browseSourceButton.addActionListener(this::browseSource);

```

```

browseDestinationButton.addActionListener(this::browseDestination);

startButton.addActionListener(this::startCopy);
resetButton.addActionListener(this::resetFields);

setVisible(true); // Hiển thị cửa sổ
}

// Hàm xử lý khi người dùng chọn file nguồn
private void browseSource(ActionEvent e) {
    JFileChooser fileChooser = new JFileChooser();
    if (fileChooser.showOpenDialog(this) == JFileChooser.APPROVE_OPTION)
    {
        sourceField.setText(fileChooser.getSelectedFile().getAbsolutePath());
    }
}

// Hàm xử lý khi người dùng chọn thư mục đích
private void browseDestination(ActionEvent e) {
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY); // Chỉ cho phép chọn thư mục
    if (fileChooser.showOpenDialog(this) == JFileChooser.APPROVE_OPTION)
    {
        destinationField.setText(fileChooser.getSelectedFile().getAbsolutePath());
    }
}

// Hàm reset các trường và thanh tiến trình
private void resetFields(ActionEvent e) {

```

```

sourceField.setText("");

destinationField.setText("");

progressBar.setValue(0);

statusLabel.setText("Copy Progress: 0 KB / 0 KB");
}

// Hàm bắt đầu quá trình sao chép
private void startCopy(ActionEvent e) {

    String source = sourceField.getText();

    String destination = destinationField.getText();


    // Kiểm tra nếu các trường nhập liệu bị trống
    if (source.isEmpty()) {

        JOptionPane.showMessageDialog(this, "Source File is empty!");

        return;

    }

    if (destination.isEmpty()) {

        JOptionPane.showMessageDialog(this, "Please select destination folder!");

        return;

    }


    startButton.setEnabled(false); // Vô hiệu hóa nút start trong quá trình sao chép

    progressBar.setValue(0); // Đặt lại thanh tiến trình về 0

    statusLabel.setText("Copy Progress: 0 KB / 0 KB"); // Đặt lại trạng thái


    // Tạo một thread để sao chép file
    ExecutorService executor = Executors.newSingleThreadExecutor();

```

```

        executor.submit(new FileCopyTask(source, destination, progressBar,
statusLabel, startButton));

        executor.shutdown();

    }


    // Hàm main để chạy ứng dụng
    public static void main(String[] args) {

        java.awt.EventQueue.invokeLater(() -> new FileCopyApp().setVisible(true));
    // Hiển thị giao diện

    }

}


// Lớp thực thi quá trình sao chép file trong một thread riêng
class FileCopyTask implements Runnable {

    private final String sourcePath;

    private final String destinationPath;

    private final JProgressBar progressBar;

    private final JLabel statusLabel;

    private final JButton startButton;


    // Constructor khởi tạo các tham số

    public FileCopyTask(String sourcePath, String destinationPath, JProgressBar
progressBar, JLabel statusLabel, JButton startButton) {

        this.sourcePath = sourcePath;

        this.destinationPath = destinationPath;

        this.progressBar = progressBar;

        this.statusLabel = statusLabel;

        this.startButton = startButton;

    }

```

```

@Override
public void run() {

    File sourceFile = new File(sourcePath);

    File destinationFile = new File(destinationPath, sourceFile.getName());

    try (InputStream input = new FileInputStream(sourceFile);
        OutputStream output = new FileOutputStream(destinationFile)) {

        long totalBytes = sourceFile.length();

        long bytesCopied = 0;

        byte[] buffer = new byte[1024 * 8];

        int bytesRead;

        DecimalFormat df = new DecimalFormat("#.##");

        // Đọc và sao chép file theo từng block
        while ((bytesRead = input.read(buffer)) != -1) {

            output.write(buffer, 0, bytesRead);

            bytesCopied += bytesRead;

            final int progress = (int) ((bytesCopied * 100) / totalBytes);

            final String status = "Copy Progress: " + df.format(bytesCopied / 1024.0)
+ " KB / " + df.format(totalBytes / 1024.0) + " KB";

            // Cập nhật thanh tiến trình và trạng thái
            SwingUtilities.invokeLater(() -> {

                progressBar.setValue(progress);

```

```

                statusLabel.setText(status);

            });

        }

        // Cập nhật trạng thái khi sao chép hoàn tất
        SwingUtilities.invokeLater(() -> {

            progressBar.setValue(100);

            statusLabel.setText("Copy Progress: Successful");

        });

    } catch (IOException e) {

        SwingUtilities.invokeLater(() -> statusLabel.setText("Error: " +
e.getMessage()));

    } finally {

        SwingUtilities.invokeLater(() -> startButton.setEnabled(true)); // Kích hoạt
lại nút start

    }

}

}

```

XÓA FILE

```

import java.io.File;

public class FileDeleteExample {

    public static void main(String[] args) {

        // Đường dẫn tới file cần xóa

        File file = new File("path/to/your/file.txt");

        // Kiểm tra xem file có tồn tại không trước khi xóa

        if (file.exists()) {

```

```

if (file.delete()) {

    System.out.println("File đã được xóa thành công.");

} else {

    System.out.println("Không thể xóa file.");

}

} else {

    System.out.println("File không tồn tại.");

}

}

}

```

TẠO FILE MỚI

```

import java.io.File;

import java.io.IOException;


public class FileCreateExample {

    public static void main(String[] args) {

        // Đường dẫn tới file cần tạo

        File file = new File("path/to/your/newfile.txt");


        try {

            // Kiểm tra và tạo file nếu chưa tồn tại

            if (file.createNewFile()) {

                System.out.println("File đã được tạo thành công.");

            } else {

                System.out.println("File đã tồn tại.");

            }

        } catch (IOException e) {

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

```

```

    }

}

}

```

Câu 1: Viết chương trình thực hiện yêu cầu: nhập vào địa chỉ URL, download nội dung của resource từ địa chỉ này về. Cho phép người dùng lưu nội dung vừa download về (lưu thành file, đường dẫn do người dùng chọn. Trích xuất từ nội dung (vừa download về) tất cả các liên kết (weblink) nếu có.

```
package com.one;
```

```
import javax.swing.*;
```

```
import java.awt.*;
```

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

```
import java.io.*;
```

```
import java.net.*;
```

```
import org.jsoup.*;
```

```
import org.jsoup.nodes.Document;
```

```
import org.jsoup.nodes.Element;
```

```
import org.jsoup.select.Elements;
```

```
public class ResourceDownloader extends JFrame {
```

```
    private JTextField urlField;
```

```
    private JTextArea linkArea;
```

```
    private JButton downloadButton, saveButton;
```

```
    private String downloadedContent;
```

```
    public ResourceDownloader() {
```

```
        setTitle("Resource Downloader");
```

```
        setSize(600, 400);
```

```
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

```

<pre> setLayout(new BorderLayout()); <i>JPanel</i> topPanel = new JPanel(new FlowLayout()); <i>JLabel</i> urlLabel = new JLabel("Enter URL:"); urlField = new JTextField(30); downloadButton = new JButton("Download Resource"); topPanel.add(urlLabel); topPanel.add(urlField); topPanel.add(downloadButton); linkArea = new JTextArea(); linkArea.setEditable(false); <i>JScrollPane</i> scrollPane = new JScrollPane(linkArea); saveButton = new JButton("Save Resource"); saveButton.setEnabled(false); add(topPanel, BorderLayout.NORTH); add(scrollPane, BorderLayout.CENTER); add(saveButton, BorderLayout.SOUTH); downloadButton.addActionListener(<i>this</i>::downloadResource); saveButton.addActionListener(<i>this</i>::saveResource); } private void downloadResource(<i>ActionEvent</i> e) { <i>String</i> url = urlField.getText(); downloadedContent = null; </pre>	<pre> linkArea.setText(""); // Clear previous links try { // Fetch resource from URL <i>URL</i> resourceUrl = new URL(url); <i>URLConnection</i> connection = (<i>URLConnection</i>) resourceUrl.openConnection(); connection.setRequestMethod("GET"); <i>BufferedReader</i> in = new BufferedReader(new InputStreamReader(connection.getInputStream())); <i>StringBuilder</i> content = new StringBuilder(); <i>String</i> line; while ((line = in.readLine()) != null) { content.append(line).append("\n"); } in.close(); downloadedContent = content.toString(); JOptionPane.showMessageDialog(<i>this</i>, "Resource downloaded successfully!"); // Extract and display links <i>Document</i> doc = Jsoup.parse(downloadedContent); <i>Elements</i> links = doc.select("a[href]"); for (<i>Element</i> link : links) { linkArea.append(link.attr("href") + "\n"); } saveButton.setEnabled(true); } catch (<i>Exception</i> ex) { JOptionPane.showMessageDialog(<i>this</i>, "Error: " + ex.getMessage()); </pre>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

```

    }
}

```

```

private void saveResource(ActionEvent e) {
    if (downloadedContent == null) {
        JOptionPane.showMessageDialog(this, "No resource to save.");
        return;
    }

    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setDialogTitle("Save Downloaded Resource");
    if (fileChooser.showSaveDialog(this) == JFileChooser.APPROVE_OPTION) {
        File file = fileChooser.getSelectedFile();
        try (FileWriter writer = new FileWriter(file)) {
            writer.write(downloadContent);

            JOptionPane.showMessageDialog(this, "Resource saved successfully to: "
+ file.getAbsolutePath());
        } catch (IOException ex) {
            JOptionPane.showMessageDialog(this, "Error saving resource: " +
ex.getMessage());
        }
    }
}

```

```

public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> {
        new ResourceDownloader().setVisible(true);
    });
}

```

```

}

```

Câu 2: Viết chương trình đồ họa như bên dưới



```

package com.one;

```

```

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

```

```

public class RunningGroupApp extends JFrame {
    private JPanel groupPanel;

    public RunningGroupApp() {
        setTitle("Running Group App");
        setSize(400, 400);
    }
}

```

```

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

setLayout(new BorderLayout());

JButton addGroupButton = new JButton("Add running group");
groupPanel = new JPanel();
groupPanel.setLayout(new BoxLayout(groupPanel, BoxLayout.Y_AXIS));

// Tạo sẵn 2 nhóm chạy ban đầu
addRunningGroup();
addRunningGroup();

addGroupButton.addActionListener(e -> addRunningGroup());

add(addGroupButton, BorderLayout.NORTH);
add(new JScrollPane(groupPanel), BorderLayout.CENTER);
}

private void addRunningGroup() {
    JPanel runningGroup = new JPanel();
    runningGroup.setLayout(new FlowLayout());

    JLabel valueLabel = new JLabel("Value: 0");
    JButton runButton = new JButton("Run");
    JButton pauseButton = new JButton("Pause");
    JButton resetButton = new JButton("Reset");

    Timer timer = new Timer(1000, new ActionListener() {
        int value = 0;

```

```

        @Override
        public void actionPerformed(ActionEvent e) {
            value++;
            valueLabel.setText("Value: " + value);
        }
    });

    runButton.addActionListener(e -> timer.start());
    pauseButton.addActionListener(e -> timer.stop());
    resetButton.addActionListener(e -> {
        timer.stop();
        valueLabel.setText("Value: 0");
    });

    runningGroup.add(valueLabel);
    runningGroup.add(runButton);
    runningGroup.add(pauseButton);
    runningGroup.add(resetButton);

    groupPanel.add(runningGroup);
    groupPanel.revalidate();
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> {
        new RunningGroupApp().setVisible(true);
    });
}

```



```
}
```

```
}
```

ĐỌC 2 FILE NHỊ PHÂN VÀ SO SÁNH

```
import java.io.*;
```

```
import java.util.*;
```

```
public class MergeAndSortBinaryFiles {
```

```
    public static void main(String[] args) {
```

```
        String fileA = "a.bin";
```

```
        String fileB = "b.bin";
```

```
        String resultFile = "kq.bin";
```

```
        try {
```

```
            // Đọc dữ liệu từ file a.bin và b.bin
```

```
            List<Integer> numbers = new ArrayList<>();
```

```
            numbers.addAll(readBinaryFile(fileA));
```

```
            numbers.addAll(readBinaryFile(fileB));
```

```
            // Sắp xếp danh sách
```

```
            Collections.sort(numbers);
```

```
            // Ghi kết quả vào file kq.bin
```

```
            writeBinaryFile(resultFile, numbers);
```

```
            System.out.println("Kết quả đã được ghi vào file " + resultFile);
```

```
        } catch (IOException e) {
```

```
            System.err.println("Có lỗi xảy ra: " + e.getMessage());
```

```
        }
```

```
}
```

```
// Hàm đọc file nhị phân và trả về danh sách các số nguyên
```

```
private static List<Integer> readBinaryFile(String fileName) throws  
IOException {
```

```
    List<Integer> numbers = new ArrayList<>();
```

```
    try (DataInputStream dis = new DataInputStream(new  
FileInputStream(fileName))) {
```

```
        int count = dis.readInt(); // Đọc số đầu tiên là số lượng phần tử
```

```
        for (int i = 0; i < count; i++) {
```

```
            numbers.add(dis.readInt()); // Đọc từng số nguyên tiếp theo
```

```
        }
```

```
    }
```

```
    return numbers;
```

```
}
```

```
// Hàm ghi danh sách số nguyên vào file nhị phân
```

```
private static void writeBinaryFile(String fileName, List<Integer> numbers)  
throws IOException {
```

```
    try (DataOutputStream dos = new DataOutputStream(new  
FileOutputStream(fileName))) {
```

```
        dos.writeInt(numbers.size()); // Ghi số lượng phần tử
```

```
        for (int num : numbers) {
```

```
            dos.writeInt(num); // Ghi từng số nguyên
```

```
        }
```

```
}
}
}
```

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG-HCM
ĐỀ THI KẾT THÚC HỌC PHẦN
Học kỳ 1 – Năm học 2022-2023

MÃ LƯU TRỮ
(sử dụng K7-ĐDCL ghi)
BA CK222b1
CSC 43402

Tên học phần: Lập trình ứng dụng Java MA HP: CSC 43402
Thời gian làm bài: 100 phút Ngày thi: 04/10/2023
Ghi chú: Sinh viên ☒ được phép / ☐ không được phép sử dụng tài liệu GIẤY khi làm bài.

Họ tên sinh viên: MSSV: STT:

Câu 1 Ứng dụng quản lý học sinh

Học sinh gồm các thông tin sau: Mã học sinh (số nguyên), String Họ tên, String Ngày sinh, Hình ảnh.

Yêu cầu: Viết chương trình cho phép thực hiện các thao tác sau:

1. Hiện thị danh sách (có tìm kiếm theo mã học sinh, họ tên) và xem thông tin chi tiết một học sinh.
2. Thêm một học sinh.

Lưu ý:

- Chương trình phải đồ giao diện đồ họa.
- Chỉ cần lưu trữ đường dẫn đến nơi chứa hình ảnh.
- SV lưu trữ dữ liệu bằng cơ sở dữ liệu SQL Server hoặc MySQL.

Câu 2 Hệ thống bán vé xem phim

Xét một hệ thống bán vé xem phim đơn giản, theo mô hình client server, kết nối với nhau thông qua TCP/IP. Để đơn giản, client và server đều là các ứng dụng dạng console.

Danh sách các vé xem phim được lưu trữ trong một file văn bản, đặt ở server. Thông tin một vé xem phim gồm: mã vé (mỗi vé có một mã số duy nhất, không trùng lặp), tên film, vị trí ghế ngồi, ngày chiếu, giờ bắt đầu chiếu, phút bắt đầu chiếu (ví dụ 21h, 30ph), giá vé, họ tên của người mua vé (nếu dữ liệu này là null, nghĩa là vé đó chưa được bán).

Hệ thống gồm 2 ứng dụng riêng biệt, client và server. Sẽ có 1 instance duy nhất của server được chạy. Server sẽ nhận các yêu cầu của client, tiến hành đọc ghi dữ liệu và trả kết quả về client. Dĩ nhiên, có nhiều instance của client, sẽ được chạy đồng thời, cùng kết nối và gửi yêu cầu đến 1 server, thông qua cơ chế socket. Do vậy, hệ thống ngoài việc xử lý mạng, cần xử lý lập trình multi-thread.

Trên mỗi ứng dụng client, người dùng có thể thực hiện 2 tác vụ:

(Đề thi gồm 2 trang)
[Trang 1/2]

Họ tên người ra đề/MSCB: Chữ ký:
Họ tên người duyệt đề: Chữ ký:

CÂU 1

```
import javax.swing.*;

import javax.swing.table.DefaultTableModel;

import java.awt.*;
```

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

import java.awt.event.ActionListener;

import java.sql.*;

import java.util.Vector;
```

```
public class StudentManagement {
```

```
// Connection URL for the MySQL database
```

```
private static final String DB_URL = "jdbc:mysql://localhost:3306/StudentDB";
```

```
private static final String DB_USERNAME = "root";
```

```
private static final String DB_PASSWORD = "password";
```

```
public static void main(String[] args) {
```

```
SwingUtilities.invokeLater(() -> new StudentManagement().createGUI());
```

```
}
```

```
private void createGUI() {
```

```
JFrame frame = new JFrame("Student Management");
```

```
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
frame.setSize(800, 600);
```

```
// Main Panel
```

```
JPanel panel = new JPanel();
```

```
panel.setLayout(new BorderLayout());
```

```
// Table to display students
```

```
JTable studentTable = new JTable();
```

```
DefaultTableModel tableModel = new DefaultTableModel();
```

```
tableModel.setColumnIdentifiers(new String[] {"Mã HS", "Họ Tên", "Ngày  
Sinh", "Hình Ảnh"});
```

```
studentTable.setModel(tableModel);
```

```
JScrollPane scrollPane = new JScrollPane(studentTable);
```

```
panel.add(scrollPane, BorderLayout.CENTER);
```

```
// Buttons for actions
```

```
JPanel buttonPanel = new JPanel();
```

```
JButton addButton = new JButton("Thêm Học Sinh");
```

```
JButton loadButton = new JButton("Hiển Thị Danh Sách");
```

```
buttonPanel.add(addButton);
```

```
buttonPanel.add(loadButton);
```

```
panel.add(buttonPanel, BorderLayout.SOUTH);
```

```
frame.add(panel);
```

```
frame.setVisible(true);
```

```
// Load students when the "Hiển Thị Danh Sách" button is clicked
```

```
loadButton.addActionListener(e -> loadStudents(tableModel));
```

```
// Open a dialog to add a new student when "Thêm Học Sinh" button is clicked
```

```
addButton.addActionListener(e -> openAddStudentDialog(frame,  
tableModel));
```

```
}
```

```
// Load students from the database into the table
```

```
private void loadStudents(DefaultTableModel tableModel) {
```

```
tableModel.setRowCount(0); // Clear the table
```

```
try (Connection conn = DriverManager.getConnection(DB_URL,  
DB_USERNAME, DB_PASSWORD);
```

```
Statement stmt = conn.createStatement();
```

```
ResultSet rs = stmt.executeQuery("SELECT * FROM Students")) {
```

```
while (rs.next()) {
```

```
Vector<Object> row = new Vector<>();
```

```
row.add(rs.getInt("student_id"));
```

```
row.add(rs.getString("full_name"));
```

```
row.add(rs.getDate("dob"));
```

```
row.add(rs.getString("image_path"));
```

```
tableModel.addRow(row);
```

```
}
```

```
} catch (SQLException ex) {
```

```
JOptionPane.showMessageDialog(null, "Lỗi khi tải dữ liệu: " +  
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
```

```
}
```

```
}
```

```
// Open a dialog to add a new student
```

```
private void openAddStudentDialog(JFrame parent, DefaultTableModel  
tableModel) {
```

```
JDialog dialog = new JDialog(parent, "Thêm Học Sinh", true);
```

```
dialog.setSize(400, 300);
```

```
dialog.setLayout(new GridLayout(5, 2));
```

```
// Input fields
```

```

JTextField idField = new JTextField();

JTextField nameField = new JTextField();

JTextField dobField = new JTextField();

JTextField imagePathField = new JTextField();


dialog.add(new JLabel("Mã Học Sinh:"));
dialog.add(idField);
dialog.add(new JLabel("Họ Tên:"));
dialog.add(nameField);
dialog.add(new JLabel("Ngày Sinh (YYYY-MM-DD):"));
dialog.add(dobField);
dialog.add(new JLabel("Đường Dẫn Hình Ảnh:"));
dialog.add(imagePathField);


JButton saveButton = new JButton("Lưu");
dialog.add(saveButton);


saveButton.addActionListener(e -> {

    int id = Integer.parseInt(idField.getText());

    String name = nameField.getText();

    String dob = dobField.getText();

    String imagePath = imagePathField.getText();


    // Save student to the database

    saveStudent(id, name, dob, imagePath);


    // Refresh the table

    loadStudents(tableModel);

```

```

        dialog.dispose();

    });


    dialog.setVisible(true);

}


// Save a student to the database

private void saveStudent(int id, String name, String dob, String imagePath) {

    String sql = "INSERT INTO Students (student_id, full_name, dob, image_path)
VALUES (?, ?, ?, ?)";


    try (Connection conn = DriverManager.getConnection(DB_URL,
DB_USERNAME, DB_PASSWORD);

        PreparedStatement pstmt = conn.prepareStatement(sql)) {

        pstmt.setInt(1, id);

        pstmt.setString(2, name);

        pstmt.setDate(3, Date.valueOf(dob));

        pstmt.setString(4, imagePath);


        pstmt.executeUpdate();

        JOptionPane.showMessageDialog(null, "Thêm học sinh thành công!",
"Success", JOptionPane.INFORMATION_MESSAGE);

    } catch (SQLException ex) {

        JOptionPane.showMessageDialog(null, "Lỗi khi thêm học sinh: " +
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);

    }

```

```
    }  
}  
  
CÂU 2  
  
TICKET.JAVA  
  
package cau2;  
  
public class Ticket {  
  
    private int id;  
  
    private String movieName;  
  
    private String seat;  
  
    private String date;  
  
    private int hour;  
  
    private int minute;  
  
    private int price;  
  
    private String buyer;  
  
  
    // Constructor để khởi tạo vé  
  
    public Ticket(int id, String movieName, String seat, String date, int hour, int minute, int price, String buyer) {  
  
        this.id = id;  
  
        this.movieName = movieName;  
  
        this.seat = seat;  
  
        this.date = date;  
  
        this.hour = hour;  
  
        this.minute = minute;  
  
        this.price = price;  
  
        this.buyer = buyer;  
  
    }  
}
```

```
// Getters và Setters  
  
public int getId() {  
  
    return id;  
  
}  
  
  
public String getMovieName() {  
  
    return movieName;  
  
}  
  
  
public String getSeat() {  
  
    return seat;  
  
}  
  
  
public String getDate() {  
  
    return date;  
  
}  
  
  
public int getHour() {  
  
    return hour;  
  
}  
  
  
public int getMinute() {  
  
    return minute;  
  
}  
  
  
public int getPrice() {  
  
    return price;  
  
}  
  
}
```

```

public String getBuyer() {
    return buyer;
}

public void setBuyer(String buyer) {
    this.buyer = buyer;
}

@Override
public String toString() {
    return "ID: " + id + ", Movie: " + movieName + ", Seat: " + seat + ", Date: " + date +
    ", Time: " + hour + ":" + minute + ", Price: " + price;
}
}

```

SERVER.JAVA

```

package cau2;

import java.io.*;
import java.net.*;
import java.util.*;

public class Server {
    // Port để server lắng nghe kết nối từ client
    private static final int PORT = 1234;

    // Đường dẫn đến file chứa dữ liệu vé
    private static final String FILE_NAME = "tickets.txt";

    // Danh sách các vé xem phim
    private static List<Ticket> tickets = new ArrayList<>();

```

```

public static void main(String[] args) {
    // Tải thông tin vé từ file vào danh sách
    loadTickets();

    // Khởi tạo ServerSocket để lắng nghe kết nối từ client
    try (ServerSocket serverSocket = new ServerSocket(PORT)) {
        System.out.println("Server is running...");

        // Chạy vòng lặp vô hạn để tiếp nhận và xử lý các kết nối từ client
        while (true) {
            // Chấp nhận kết nối từ client
            Socket clientSocket = serverSocket.accept();

            // Mỗi khi có kết nối, tạo một thread mới để xử lý kết nối này
            new ClientHandler(clientSocket).start();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}

```

// Hàm đọc file và tải dữ liệu vé vào danh sách

```

private static void loadTickets() {
    try (BufferedReader br = new BufferedReader(new FileReader(FILE_NAME))) {
        String line;

        // Đọc từng dòng trong file và tạo các đối tượng Ticket tương ứng
        while ((line = br.readLine()) != null) {
            String[] data = line.split("\\|");

```

```

        tickets.add(new Ticket(Integer.parseInt(data[0]), data[1], data[2], data[3],
Integer.parseInt(data[4]),

        Integer.parseInt(data[5]), Integer.parseInt(data[6]), data[7]));

    }

} catch (IOException e) {

    e.printStackTrace();

}

}

```

// Hàm xử lý yêu cầu từ client

```

private static synchronized String handleRequest(String request) {

    // Chia nhỏ yêu cầu để xác định hành động cần thực hiện

    String[] parts = request.split(";");

    if (parts[0].equals("search")) {

        // Tìm kiếm vé theo tên phim và ngày chiếu

        String result = searchTickets(parts[1], parts[2]);

        return result;

    } else if (parts[0].equals("buy")) {

        // Mua vé, nhận mã vé và tên người mua

        String result = buyTicket(Integer.parseInt(parts[1]), parts[2]);

        return result;

    }

    return "Invalid request."; // Trả về nếu yêu cầu không hợp lệ

}

```

// Hàm tìm kiếm vé theo tên phim và ngày chiếu

```

private static String searchTickets(String movieName, String date) {

    StringBuilder sb = new StringBuilder();

```

// Duyệt qua tất cả vé để tìm vé thỏa mãn điều kiện

```

    for (Ticket ticket : tickets) {

        if (ticket.getMovieName().contains(movieName) &&
ticket.getDate().equals(date) && ticket.getBuyer() == null) {

            sb.append(ticket).append("\n"); // Thêm vé vào kết quả nếu còn trống

        }

    }

    return sb.length() > 0 ? sb.toString() : "No tickets found."; // Trả về vé tìm được
hoặc thông báo không có vé

}

```

// Hàm mua vé

```

private static String buyTicket(int ticketId, String buyerName) {

    for (Ticket ticket : tickets) {

        if (ticket.getId() == ticketId) {

            if (ticket.getBuyer() == null) {

                ticket.setBuyer(buyerName); // Ghi tên người mua vào vé

                return "Ticket bought successfully."; // Trả về thông báo thành công

            } else {

                return "Ticket already bought."; // Trả về nếu vé đã có người mua

            }

        }

    }

    return "Ticket not found."; // Trả về nếu không tìm thấy vé với mã ticketId

}

```

// Lớp xử lý từng kết nối client

```

static class ClientHandler extends Thread {

    private Socket socket;

```

```

private PrintWriter out;

private BufferedReader in;

public ClientHandler(Socket socket) {
    this.socket = socket;
}

@Override

public void run() {
    try {
        // Tạo stream để nhận và gửi dữ liệu
        in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
        out = new PrintWriter(socket.getOutputStream(), true);

        // Đọc yêu cầu từ client
        String request = in.readLine();
        // Xử lý yêu cầu và trả kết quả về client
        String response = handleRequest(request);
        out.println(response);
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        try {
            socket.close(); // Đóng kết nối với client sau khi xử lý xong
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

```

    }
}

CLIENT.JAVA

package cau2;

import java.io.*;
import java.net.*;
import java.util.*;

public class Client {
    // Địa chỉ IP và port của server

    private static final String SERVER_IP = "192.168.10";
    private static final int SERVER_PORT = 1234;
    private static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        while (true) {
            // Hiển thị menu cho người dùng chọn hành động
            System.out.println("===== Movie Ticket System =====");
            System.out.println("1. Search tickets");
            System.out.println("2. Buy ticket");
            System.out.println("3. Exit");
            System.out.print("Choose an option: ");

            int choice = scanner.nextInt();
            scanner.nextLine(); // Đọc dòng mới sau khi nhập số

            switch (choice) {

```



```

case 1:
    // Tìm kiếm vé
    searchTickets();

    break;
case 2:
    // Mua vé
    buyTicket();

    break;
case 3:
    // Thoát chương trình
    System.out.println("Exiting...");

    return;
default:
    System.out.println("Invalid option. Please try again.");

}

}

}

// Hàm tìm kiếm vé
private static void searchTickets() {

    System.out.print("Enter movie name to search: ");

    String movieName = scanner.nextLine();

    System.out.print("Enter date to search (format: yyyy-mm-dd): ");

    String date = scanner.nextLine();

    try (Socket socket = new Socket(SERVER_IP, SERVER_PORT);

        PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

        BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream())) {

        // Gửi yêu cầu tìm kiếm đến server

        String request = "search;" + movieName + ";" + date;

        out.println(request);

        // Nhận kết quả tìm kiếm từ server

        String response = in.readLine();

        System.out.println("Search result:\n" + response);

    } catch (IOException e) {

        e.printStackTrace();

    }

}

}

// Hàm mua vé
private static void buyTicket() {

    System.out.print("Enter ticket ID to buy: ");

    int ticketId = scanner.nextInt();

    scanner.nextLine(); // Đọc dòng mới sau khi nhập số

    System.out.print("Enter your name: ");

    String buyerName = scanner.nextLine();

    try (Socket socket = new Socket(SERVER_IP, SERVER_PORT);

        PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

        BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream())) {

        // Gửi yêu cầu mua vé đến server

```

```

        BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream())) {

        // Gửi yêu cầu tìm kiếm đến server

        String request = "search;" + movieName + ";" + date;

        out.println(request);

        // Nhận kết quả tìm kiếm từ server

        String response = in.readLine();

        System.out.println("Search result:\n" + response);

    } catch (IOException e) {

        e.printStackTrace();

    }

}

}

// Hàm mua vé
private static void buyTicket() {

    System.out.print("Enter ticket ID to buy: ");

    int ticketId = scanner.nextInt();

    scanner.nextLine(); // Đọc dòng mới sau khi nhập số

    System.out.print("Enter your name: ");

    String buyerName = scanner.nextLine();

    try (Socket socket = new Socket(SERVER_IP, SERVER_PORT);

        PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

        BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream())) {

        // Gửi yêu cầu mua vé đến server

```

```
String request = "buy;" + ticketId + ";" + buyerName;
```

```
out.println(request);
```

```
// Nhận kết quả từ server về việc mua vé thành công hay thất bại
```

```
String response = in.readLine();
```

```
System.out.println("Response: " + response);
```


```
} catch (IOException e) {
```

```
    e.printStackTrace();
```

```
}
```

```
}
```

```
}
```

 **TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG-HCM**
ĐỀ THI KẾT THÚC HỌC PHẦN
Học kỳ 2 – Năm học 2021-2022

MÃ LƯU TRỮ
(mã phòng KT-ĐBCL ghi)
CK 2122-2
CSC 13102

Tên học phần: Lập trình ứng dụng Java MA HP: CSC13102
Thời gian làm bài: 120 phút Ngày thi: 17/6/2022
Ghi chú: Sinh viên ☒ được phép / ☐ không được phép sử dụng tài liệu khi làm bài.

Họ tên sinh viên: MSSV: STT:

Viết chương trình giám sát sự thay đổi của một thư mục trên các máy tính từ xa. Sinh viên cần viết hai chương trình, một chương trình client chạy tại các máy cần giám sát, một chương trình server chạy tại máy giám sát.

- Chương trình Client: kết nối và trao đổi thông tin với server.
- Chương trình Server:
 - o Quản lý danh sách các client.
 - o Chọn client cần giám sát, chọn thư mục trên máy client cần giám sát. Mọi thay đổi trên thư mục này (xóa/thêm các thư mục con, xóa/thêm/chỉnh sửa nội dung các tập tin bên trong thư mục) chương trình cần phải thông báo cho người dùng (tại server).
 - o Có thể giám sát cùng lúc nhiều client (mỗi client chỉ giám sát một thư mục)
 - o Giao diện đồ họa.

Hướng dẫn: Lớp **java.io.File** có các phương thức sau:

<code>File(String pathname)</code>	//tạo một đối tượng File đến một file hoặc thư mục cho trước
<code>boolean isDirectory()</code>	//đối tượng File hiện thời có phải là thư mục không
<code>boolean isFile()</code>	//đối tượng File hiện thời có phải là tập tin không
<code>long length()</code>	//size của file
<code>long lastModified()</code>	//trả về thời gian thay đổi cuối cùng của file/thư mục hiện thời
<code>File[] listFiles()</code>	//danh sách file/thư mục con trong thư mục hiện thời

```
import javax.swing.*;
```

```
import java.awt.*;
```

```
import java.io.*;
```

```

import java.net.*;

import java.util.concurrent.*;

public class DirectoryMonitorServer {

    private JFrame frame; // Giao diện chính của server

    private JTextArea logArea; // Khu vực hiển thị nhật ký

    private ExecutorService executorService; // Quản lý các luồng client

    public DirectoryMonitorServer() {

        initUI(); // Khởi tạo giao diện

        startServer(); // Bắt đầu server

    }

    private void initUI() {

        frame = new JFrame("Server - Directory Monitor");

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        frame.setSize(600, 400);

        logArea = new JTextArea(); // Hiển thị nhật ký hoạt động

        logArea.setEditable(false);

        JScrollPane scrollPane = new JScrollPane(logArea); // Thêm thanh cuộn

        frame.add(scrollPane, BorderLayout.CENTER);

        frame.setVisible(true);

    }

    private void startServer() {

```

```

        executorService = Executors.newCachedThreadPool(); // Sử dụng luồng để xử lý
        nhiều client

        new Thread() -> {

            try (ServerSocket serverSocket = new ServerSocket(12345)) { // Server lắng
            nghe trên cổng 12345

                logArea.append("Server started. Waiting for clients...\n");

                while (true) {

                    Socket clientSocket = serverSocket.accept(); // Chấp nhận kết nối từ client

                    logArea.append("Client connected: " + clientSocket.getInetAddress() +
                    "\n");

                    executorService.submit(() -> handleClient(clientSocket)); // Tạo luồng mới
                    để xử lý client

                }

            } catch (IOException e) {

                logArea.append("Error starting server: " + e.getMessage() + "\n");

            }

        }).start();

    }

    private void handleClient(Socket clientSocket) {

        try (BufferedReader in = new BufferedReader(new
        InputStreamReader(clientSocket.getInputStream())) {

            String message;

            while ((message = in.readLine()) != null) {

                logArea.append("Message from client: " + message + "\n"); // Hiển thị thông
                báo từ client

            }

        } catch (IOException e) {

            logArea.append("Client disconnected: " + e.getMessage() + "\n"); // Thông báo
            khi client ngắt kết nối

```

```

    }

}

public static void main(String[] args) {

    SwingUtilities.invokeLater(DirectoryMonitorServer::new); // Chạy chương trình
server
}

}

CLIENT

import javax.swing.*;
import java.awt.*;
import java.io.*;
import java.net.*;
import java.util.*;

public class DirectoryMonitorClient {

    private JFrame frame; // Cửa sổ giao diện chính

    private TextField serverAddressField, directoryField; // Trường nhập địa chỉ server
và thư mục

    private Button connectButton, selectFolderButton; // Nút bấm kết nối và chọn thư
mục

    private TextArea logArea; // Khu vực hiển thị nhật ký

    private Socket socket; // Socket để kết nối tới server

    private PrintWriter out; // Gửi dữ liệu đến server

    private File directoryToMonitor; // Thư mục được giám sát


    public DirectoryMonitorClient() {

        initUI(); // Khởi tạo giao diện người dùng

    }

```

```

private void initUI() {

    // Tạo khung giao diện chính

    frame = new JFrame("Client - Directory Monitor");

    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    frame.setSize(600, 400);


    // Panel trên cùng chứa các trường nhập

    JPanel topPanel = new JPanel(new GridLayout(2, 2));

    topPanel.add(new JLabel("Server Address:")); // Nhãn địa chỉ server

    serverAddressField = new JTextField("localhost"); // Mặc định là localhost

    topPanel.add(serverAddressField);

    topPanel.add(new JLabel("Directory to Monitor:")); // Nhãn chọn thư mục

    directoryField = new JTextField(); // Trường hiển thị thư mục được chọn

    directoryField.setEditable(false); // Không cho phép chỉnh sửa trực tiếp

    topPanel.add(directoryField);


    // Panel giữa chứa nút chọn thư mục và kết nối

    JPanel middlePanel = new JPanel(new FlowLayout());

    selectFolderButton = new JButton("Select Folder"); // Nút chọn thư mục

    selectFolderButton.addActionListener(e -> selectDirectory()); // Thêm sự kiện bấm
nút

    connectButton = new JButton("Connect"); // Nút kết nối

    connectButton.addActionListener(e -> connectToServer()); // Thêm sự kiện bấm
nút

    middlePanel.add(selectFolderButton);

    middlePanel.add(connectButton);


    // Khu vực hiển thị nhật ký

```

```

logArea = new JTextArea();

logArea.setEditable(false);

JScrollPane scrollPane = new JScrollPane(logArea); // Thêm thanh cuộn


// Thêm các thành phần vào giao diện chính

frame.add(topPanel, BorderLayout.NORTH);

frame.add(middlePanel, BorderLayout.CENTER);

frame.add(scrollPane, BorderLayout.SOUTH);


frame.setVisible(true); // Hiển thị giao diện
}


// Hàm chọn thư mục giám sát

private void selectDirectory() {

    JFileChooser fileChooser = new JFileChooser(); // Hộp thoại chọn file/thư mục

    fileChooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY); // Chỉ
cho phép chọn thư mục

    int result = fileChooser.showOpenDialog(frame);

    if (result == JFileChooser.APPROVE_OPTION) {

        directoryToMonitor = fileChooser.getSelectedFile(); // Lấy thư mục được chọn

        directoryField.setText(directoryToMonitor.getAbsolutePath()); // Hiển thị thư
mục trong trường nhập

    }

}


// Hàm kết nối tới server

private void connectToServer() {

    String serverAddress = serverAddressField.getText(); // Lấy địa chỉ server từ
trường nhập

```

```

if (directoryToMonitor == null) {

    JOptionPane.showMessageDialog(frame, "Please select a directory first!"); //
Cảnh báo nếu chưa chọn thư mục

    return;

}


try {

    socket = new Socket(serverAddress, 12345); // Kết nối đến server qua cổng
12345

    out = new PrintWriter(socket.getOutputStream(), true); // Chuẩn bị gửi dữ liệu
tới server

    logArea.append("Connected to server.\n");

    logArea.append("Monitoring directory: " +
directoryToMonitor.getAbsolutePath() + "\n");


    // Bắt đầu luồng giám sát thư mục

    new Thread(this::monitorDirectory).start();

} catch (IOException e) {

    logArea.append("Error connecting to server: " + e.getMessage() + "\n");

}

}


// Hàm giám sát sự thay đổi của thư mục

private void monitorDirectory() {

    Map<String, Long> fileState = new HashMap<>(); // Lưu trạng thái của các
file/thư mục

    while (true) {

        try {

            // Lấy danh sách file/thư mục con trong thư mục đang giám sát

```

```

File[] files = directoryToMonitor.listFiles();

if (files != null) {

    Map<String, Long> newState = new HashMap<>();

    for (File file : files) {

        newState.put(file.getName(), file.lastModified()); // Lưu tên file và thời
gian sửa đổi

    }

    // Kiểm tra file mới hoặc bị sửa đổi

    for (String fileName : newState.keySet()) {

        if (!fileState.containsKey(fileName)) {

            logArea.append("New file detected: " + fileName + "\n");

            out.println("New file: " + fileName); // Gửi thông báo tới server

        } else if (!fileState.get(fileName).equals(newState.get(fileName))) {

            logArea.append("File modified: " + fileName + "\n");

            out.println("File modified: " + fileName); // Gửi thông báo tới server

        }

    }

    // Kiểm tra file bị xóa

    for (String fileName : fileState.keySet()) {

        if (!newState.containsKey(fileName)) {

            logArea.append("File deleted: " + fileName + "\n");

            out.println("File deleted: " + fileName); // Gửi thông báo tới server

        }

    }

    fileState = newState; // Cập nhật trạng thái file hiện tại

```

```

    }

    Thread.sleep(1000); // Giám sát mỗi giây

} catch (Exception e) {

    logArea.append("Error monitoring directory: " + e.getMessage() + "\n");

    break;

}

}

}

public static void main(String[] args) {

    SwingUtilities.invokeLater(DirectoryMonitorClient::new); // Chạy chương trình
client

}

}

```



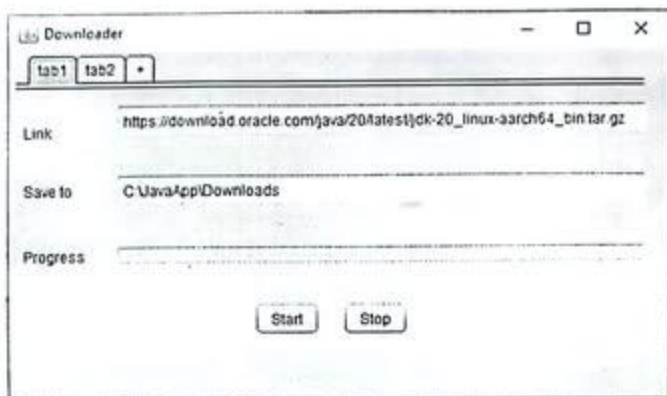
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG-HCM
ĐỀ THI KẾT THÚC HỌC PHẦN
Học kỳ 2 – Năm học 2022-2023

MÃ LƯU TRỮ
(do phòng KT-ĐBCL ghi)
CK 2223.2
CSC 13102

Tên học phần: Lập trình Ứng dụng Java Mã HP: CSC13102
Thời gian làm bài: 100 phút Ngày thi: 01/11/2023
Ghi chú: Sinh viên ☒ được phép / ☐ không được phép sử dụng tài liệu khi làm bài.

Họ tên sinh viên: MSSV: STT:

Sinh viên viết chương trình (giao diện đồ họa) bằng ngôn ngữ lập trình Java, cách thức hoạt động như mô tả bên dưới (xem hình)



Người dùng nhập vào liên kết (link) đến một tài nguyên (resource) trên internet và đường dẫn đến nơi cần lưu trữ tài nguyên này (trên đĩa cứng máy người dùng), sau đó người dùng nhấn nút Start thì chương trình sẽ download tài nguyên này về và lưu trữ xuống đường dẫn đã chọn. Thanh progress bar sẽ hiển thị phần trăm hoàn thành của quá trình download này.

Người dùng có thể nhấn Stop để chấm dứt việc download (có thể nhấn Start để bắt đầu download lại sau khi stop).

Người dùng có thể nhấn vào tab có dấu + để tạo tab mới, lúc đó tab mới sẽ có các thành phần tương tự để cho phép người dùng nhập vào và download một tài nguyên khác trên internet về máy tính của mình (tab mới chứa đầy đủ các thông tin liên kết, nơi lưu trữ, thanh progress bar, các nút Start, Stop).

Các lưu ý:

- Nếu có nhiều tab đang mở và yêu cầu download, chương trình phải thực hiện việc download đồng thời các tài nguyên được yêu cầu.
- Giao diện như hình bên trên chỉ để minh họa yêu cầu, sinh viên có thể thiết kế giao diện khác, miễn sao đáp ứng các chức năng của chương trình.

(Đề thi gồm 1 trang)

Họ tên người ra đề/MSCB: Chữ ký: [Trang 1/1]
Họ tên người duyệt đề: Chữ ký:

```
import java.io.*;
import java.net.*;
import java.util.concurrent.*;
import javax.swing.event.*;
```

```
public class DownloadManager extends JFrame {
```

```
    private JTabbedPane tabbedPane; // TabbedPane chứa các tab
```

```
    private ExecutorService executor; // Executor để quản lý các luồng tải xuống đồng thời
```

```
// Constructor để khởi tạo giao diện và các thành phần
```

```
public DownloadManager() {
```

```
    setTitle("Download Manager");
```

```
    setSize(600, 400);
```

```
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
    setLocationRelativeTo(null);
```

```
// Khởi tạo TabbedPane để quản lý các tab tải xuống
```

```
tabbedPane = new JTabbedPane();
```

```
tabbedPane.addTab("Tab 1", createDownloadPanel()); // Tab đầu tiên
```

```
tabbedPane.addChangeListener(new ChangeListener() {
```

```
    public void stateChanged(ChangeEvent e) {
```

```
        // Có thể thêm hành động khi người dùng thay đổi tab, nhưng không cần thiết trong trường hợp này
```

```
    }
```

```
});
```

```
add(tabbedPane, BorderLayout.CENTER); // Thêm TabbedPane vào khung giao diện
```

```
// Thêm nút "+" ở trên cùng để người dùng tạo thêm tab mới

JButton addButton = new JButton("+");

addButton.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {

        // Khi nhấn nút "+", tạo tab mới

        tabbedPane.addTab("Tab " + (tabbedPane.getTabCount() + 1),
createDownloadPanel());

    }

});

add(addButton, BorderLayout.NORTH); // Thêm nút "+" vào khung giao diện


// Khởi tạo ExecutorService để tải tài nguyên đồng thời

executor = Executors.newCachedThreadPool();

}


// Phương thức tạo panel tải xuống, mỗi tab sẽ có một panel riêng biệt

private JPanel createDownloadPanel() {

    JPanel panel = new JPanel();

    panel.setLayout(new GridLayout(5, 2)); // Thiết kế giao diện với lưới (GridLayout)


    // Các thành phần trong tab: TextField cho link và nơi lưu trữ, ProgressBar, các nút
điều khiển

    JTextField linkField = new JTextField();

    JTextField saveField = new JTextField();

    JProgressBar progressBar = new JProgressBar(0, 100); // Thanh tiến độ từ 0 đến
100%

    progressBar.setStringPainted(true); // Hiển thị phần trăm trên thanh tiến độ
```

```
JButton startButton = new JButton("Start"); // Nút bắt đầu tải xuống

JButton stopButton = new JButton("Stop"); // Nút dừng tải xuống


startButton.setEnabled(true); // Nút Start được kích hoạt khi bắt đầu

stopButton.setEnabled(false); // Nút Stop không được kích hoạt ngay khi bắt đầu


// Thêm các thành phần vào panel

panel.add(new JLabel("Link:"));

panel.add(linkField);

panel.add(new JLabel("Save to:"));

panel.add(saveField);

panel.add(progressBar);


panel.add(startButton);

panel.add(stopButton);


// Xử lý sự kiện khi nhấn nút Start

startButton.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {

        String link = linkField.getText(); // Lấy link tài nguyên

        String savePath = saveField.getText(); // Lấy đường dẫn lưu trữ

        startDownload(link, savePath, progressBar, startButton, stopButton); // Bắt
đầu tải tài nguyên

    }

});


// Xử lý sự kiện khi nhấn nút Stop

stopButton.addActionListener(new ActionListener() {
```



```

public void actionPerformed(ActionEvent e) {

    stopDownload(); // Dừng tải tài nguyên (Chưa thực hiện dừng thực sự trong
code này)

    startButton.setEnabled(true); // Kích hoạt lại nút Start

    stopButton.setEnabled(false); // Tắt nút Stop

}

});

return panel; // Trả về panel đã tạo

}

// Phương thức bắt đầu tải tài nguyên

private void startDownload(String link, String savePath, JProgressBar progressBar,
JButton startButton, JButton stopButton) {

    startButton.setEnabled(false); // Tắt nút Start khi tải xuống bắt đầu

    stopButton.setEnabled(true); // Kích hoạt nút Stop khi tải xuống bắt đầu


// Tạo một tác vụ tải tài nguyên

    DownloadTask task = new DownloadTask(link, savePath, progressBar, startButton,
stopButton);

    executor.submit(task); // Gửi tác vụ tải xuống vào executor để thực hiện đồng thời

}

// Phương thức dừng tải tài nguyên (chưa triển khai trong ví dụ này)

private void stopDownload() {

    // Dừng tải xuống (có thể cần hủy các luồng tải xuống đang hoạt động)

}

// Lớp đại diện cho tác vụ tải tài nguyên

```

```

private class DownloadTask implements Runnable {

    private String link; // Link tải tài nguyên

    private String savePath; // Đường dẫn lưu trữ tài nguyên

    private JProgressBar progressBar; // Thanh tiến độ

    private JButton startButton; // Nút Start

    private JButton stopButton; // Nút Stop


// Constructor khởi tạo các giá trị cần thiết cho tác vụ tải xuống

    public DownloadTask(String link, String savePath, JProgressBar progressBar,
JButton startButton, JButton stopButton) {

        this.link = link;

        this.savePath = savePath;

        this.progressBar = progressBar;

        this.startButton = startButton;

        this.stopButton = stopButton;

    }


// Phương thức thực thi tác vụ tải tài nguyên

    public void run() {

        try {

            URL url = new URL(link); // Chuyển link thành đối tượng URL

            HttpURLConnection connection = (HttpURLConnection)
url.openConnection(); // Mở kết nối HTTP

            connection.setRequestMethod("GET"); // Sử dụng phương thức GET để tải tài
nguyên

            connection.connect(); // Kết nối đến server


            int fileSize = connection.getContentLength(); // Lấy kích thước tệp tin

```

```
InputStream inputStream = connection.getInputStream(); // Lấy InputStream
từ kết nối

FileOutputStream fileOutputStream = new FileOutputStream(savePath); //
Tạo FileOutputStream để lưu tệp

byte[] buffer = new byte[1024]; // Bộ đệm để đọc dữ liệu
int bytesRead;
int bytesDownloaded = 0; // Biến lưu số byte đã tải xuống

// Đọc và ghi dữ liệu từ server vào tệp
while ((bytesRead = inputStream.read(buffer)) != -1) {
    fileOutputStream.write(buffer, 0, bytesRead); // Ghi dữ liệu vào tệp
    bytesDownloaded += bytesRead; // Cập nhật số byte đã tải xuống
    int progress = (int) ((bytesDownloaded / (float) fileSize) * 100); // Tính
phần trăm tiến độ
    progressBar.setValue(progress); // Cập nhật thanh tiến độ
}

fileOutputStream.close(); // Đóng FileOutputStream
inputStream.close(); // Đóng InputStream

// Hiển thị thông báo khi tải xuống hoàn tất
JOptionPane.showMessageDialog(null, "Download complete!");

} catch (IOException e) {
    // Hiển thị thông báo nếu có lỗi xảy ra trong quá trình tải
    JOptionPane.showMessageDialog(null, "Error: " + e.getMessage());
} finally {
    // Sau khi hoàn thành hoặc có lỗi, kích hoạt lại nút Start và tắt nút Stop
```

```
SwingUtilities.invokeLater(new Runnable() {

    public void run() {
        startButton.setEnabled(true);
        stopButton.setEnabled(false);
    }
});

}

}

// Phương thức main để chạy ứng dụng
public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            new DownloadManager().setVisible(true); // Hiển thị cửa sổ chính
        }
    });
}
}
```