

Session 7**Introduction to Threads****Phần I - Thực hiện trong 120 phút****1. Mục tiêu**

- Giới thiệu Thread.
- Khởi tạo Thread, quản lý thread, hiểu bản chất daemon thread.
- Hiểu biết phương thức trong thread.

2. Thực hiện

Bài thực hành 1: Tạo một class tên là RunnableDemo implement Runnable. Override hàm run() và cho chạy vòng lặp từ 4 đếm lùi về 0, in ra tên của luồng và giá trị của biến i trong vòng lặp.

Bước 1: Tạo class RunnableDemo implement Runnable, viết code gán giá trị cho biến lưu tên trong constructor

```
package demo.jp2.lab04;

/**
 *
 * @author minhvuvc
 */
public class RunnableDemo implements Runnable {

    private Thread t;
    private String threadName;

    RunnableDemo(String name) {
        threadName = name;
        System.out.println("Creating " + threadName);
    }

    //Bước 2: Override hàm run()
    @Override
    public void run() {
        System.out.println("Running " + threadName);
        try {
            for (int i = 4; i > 0; i--) {
                System.out.println("Thread: " + threadName + ", " + i);
                // Let the thread sleep for a while.
                Thread.sleep(1500);
            }
        } catch (InterruptedException e) {
```

```
        System.out.println("Thread " + threadName + " interrupted.");
    }
    System.out.println("Thread " + threadName + " exiting.");
}

//Bước 3: Viết hàm start()
public void start() {
    System.out.println("Starting " + threadName);
    if (t == null) {
        t = new Thread(this, threadName);
        t.start();
    }
}
}
```

Bước 4: Viết class TestRunnable có hàm main(), gọi và sử dụng.

```
public class TestRunnable {

    /**
     * @param args the commandline arguments
     */
    public static void main(String[] args) {
        RunnableDemo R1 = new RunnableDemo("Thread-1");
        R1.start();

        RunnableDemo R2 = new RunnableDemo("Thread-2");
        R2.start();
    }
}
```

Bước 5: Kết quả

Output - demo.jp2.lab04 (run)

```
run:
Creating Thread-1
Starting Thread-1
Creating Thread-2
Starting Thread-2
Running Thread-1
Running Thread-2
Thread: Thread-2, 4
Thread: Thread-1, 4
Thread: Thread-2, 3
Thread: Thread-1, 3
Thread: Thread-1, 2
Thread: Thread-2, 2
Thread: Thread-1, 1
Thread: Thread-2, 1
Thread Thread-1 exiting.
Thread Thread-2 exiting.
BUILD SUCCESSFUL (total time: 6 seconds)
```

Bài thực hành 2: Viết một class tên là ThreadDemo kế thừa Thread. Override hàm run() và cho chạy vòng lặp từ 4 đếm lùi về 0, in ra tên của luồng và giá trị của biến i trong vòng lặp.

Bước 1: Tạo class ThreadDemo kế thừa class Thread và viết constructor cho nó.

```
package demo.jp2.lab04;

/**
 *
 * @author minhvuvc
 */
public class ThreadDemo extends Thread {

    private Thread t;
    private String threadName;

    ThreadDemo(String name) {
        threadName = name;
        System.out.println("Creating " + threadName);
    }

    //Bước 2: Override hàm run()
    @Override
    public void run() {
        System.out.println("Running " + threadName);
        try {
```

```

        for (int i = 4; i > 0; i--) {
            System.out.println("Thread: " + threadName + ", " + i);
            // Let the thread sleep for a while.
            Thread.sleep(50);
        }
    } catch (InterruptedException e) {
        System.out.println("Thread " + threadName + " interrupted.");
    }
    System.out.println("Thread " + threadName + " exiting.");
}

//Bước 3: Override hàm start()
@Override
public void start() {
    System.out.println("Starting " + threadName);
    if (t == null) {
        t = new Thread(this, threadName);
        t.start();
    }
}
}

```

Bước 4: Tạo TestThread có hàm main().

```

public class TestThread {
    /**
     * @param args thecommandlinearguments
     */
    public static void main(String[] args) {
        ThreadDemo T1 = new ThreadDemo("Thread-1");
        T1.start();

        ThreadDemo T2 = new ThreadDemo("Thread-2");
        T2.start();
    }
}

```

Bước 5: Kết quả

Output - demo.jp2.lab04 (run)

```
run:
Creating Thread-1
Starting Thread-1
Creating Thread-2
Running Thread-1
Starting Thread-2
Thread: Thread-1, 4
Running Thread-2
Thread: Thread-2, 4
Thread: Thread-1, 3
Thread: Thread-2, 3
Thread: Thread-1, 2
Thread: Thread-2, 2
Thread: Thread-1, 1
Thread: Thread-2, 1
Thread Thread-1 exiting.
Thread Thread-2 exiting.
BUILD SUCCESSFUL (total time: 0 seconds)
```

Bài thực hành 3: Tạo một class tên là PrintNumber kế thừa từ Thread và một class thực thi interface Runnable tên là PrintCharacter. Sau mỗi giây, PrintNumber sẽ in ra một số nguyên (cho vòng lặp từ 0 đến 100), tương tự PrintCharacter sẽ in ra chữ cái (từ A-Z). Tạo MainClass có hàm main() và khởi tạo 2 thread trên.

Bước 1: Tạo class PrintNumber kế thừa Thread

```
public class PrintNumber extends Thread {

    @Override
    public void run() {
        for (int i = 0; i < 100; i++) {
            System.out.println("#" + i);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException ex) {
                Logger.getLogger(PrintNumber.class.getName()).log(Level.SEVERE, null,
ex);
            }
        }
    }
}
```

```
}  
}
```

Bước 2: Tạo class printCharacter thực thi Runnable


```
public class PrintCharacter implements Runnable {  
  
    @Override  
    public void run() {  
        for (int i = 'A'; i < 'Z'; i++) {  
            System.out.println("\t" + (char) i);  
            try {  
                Thread.sleep(200);  
            } catch (InterruptedException ex) {  
                Logger.getLogger(PrintCharacter.class.getName()).log(Level.SEVERE, null,  
ex);  
            }  
        }  
    }  
}
```

Bước 3: Tạo MainClass có hàm main()

```
public class MainClass {  
  
    /**  
     * @param args thecommandlinearguments  
     */  
    public static void main(String[] args) {  
        PrintNumber printN = new PrintNumber();  
        Thread printC = new Thread(new PrintCharacter());  
        printN.start();  
        printC.start();  
    }  
}
```

```
}  
}
```

Bước 4: Kết quả



```
Output - demo.jp2.lab04 (run)  
run :  
#0 A  
#1 B  
#2 C  
#3 D  
#4 E  
#5 F  
#6  
#7  
#8  
#9  
#10  
#11
```

Bài thực hành 4: Tạo class tên là MyThread implement Runnable, tạo class MainClass có hàm main(). Khởi tạo mảng 5 MyThread và kích hoạt trong đối tượng ExecutorService đồng thời.

Bước 1: Viết class MyThread

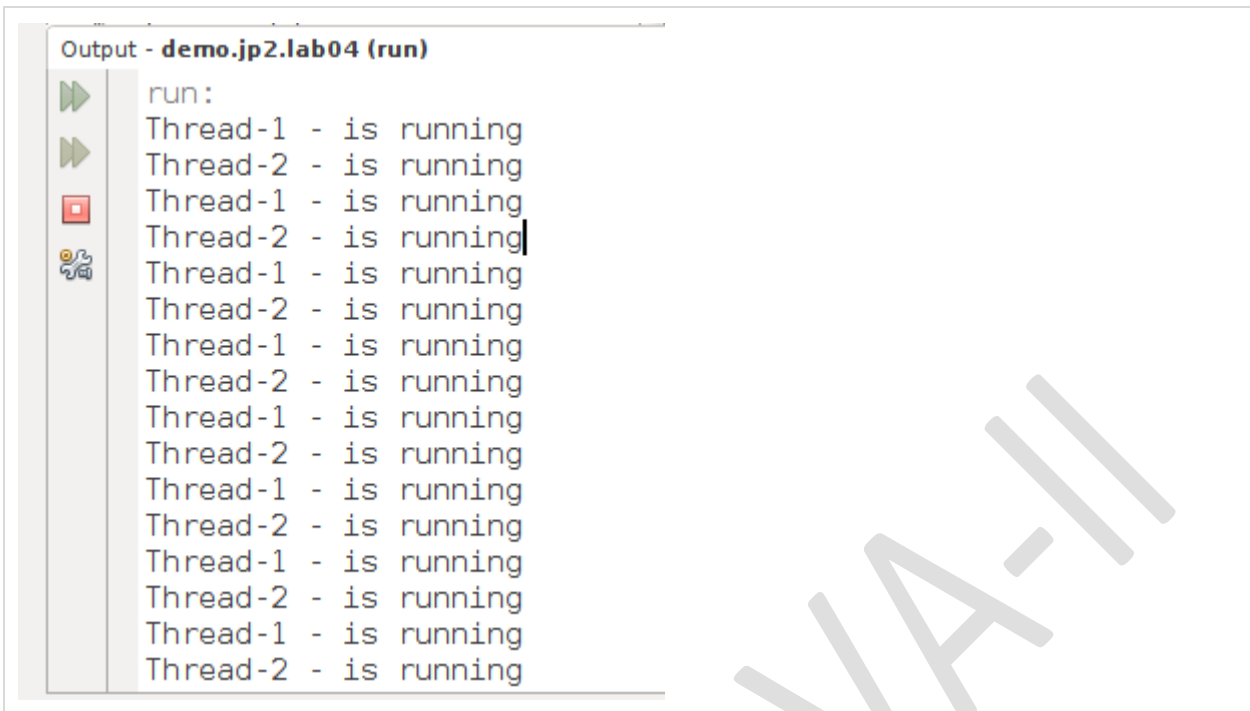
```
public class MyThread implements Runnable {  
  
    String name;  
  
    public MyThread(String name) {  
        this.name = name;  
    }  
  
    @Override  
    public void run() {
```

```
for (int i = 0; i < 10; i++) {  
    System.out.println(name + " - is running");  
    try {  
        Thread.sleep(300);  
    } catch (InterruptedException ex) {  
        Logger.getLogger(MyThread.class.getName()).log(Level.SEVERE, null, ex);  
    }  
}  
}
```

Bước 2: Viết class DemoExcutorService có hàm main()

```
public class DemoExcutorService {  
  
    /**  
     * @param args thecommandlinearguments  
     */  
    public static void main(String[] args) {  
        ExecutorService service = Executors.newFixedThreadPool(2);  
        for (int i = 0; i < 5; i++) {  
            MyThread mt = new MyThread("Thread-" + (i + 1));  
            service.submit(mt);  
        }  
    }  
}
```

Bước 3: Kết quả



```
Output - demo.jp2.lab04 (run)
run:
Thread-1 - is running
Thread-2 - is running
Thread-1 - is running
Thread-2 - is running
Thread-1 - is running
Thread-2 - is running
Thread-1 - is running
Thread-2 - is running
Thread-1 - is running
Thread-2 - is running
Thread-1 - is running
Thread-2 - is running
Thread-1 - is running
Thread-2 - is running
Thread-1 - is running
Thread-2 - is running
```

Phần II - Bài tập tự làm

1. Tạo một thread sau mỗi giây hiển thị một số ngẫu nhiên từ 1 đến 100.
2. Tạo một thread sau mỗi giây hiển thị một tên tỉnh bất kỳ hoặc tuần tự (tên các tỉnh được lưu trong một mảng ban đầu của thread1).
3. Tạo hai thread:
 - Thread thứ nhất sinh ra một số ngẫu nhiên và hiển thị số đó ra màn hình
 - Thread thứ hai lấy ra giờ của hệ thống và thông báo bây giờ là mấy giờ
4. Tạo Thread A hiển thị ngẫu nhiên tên sinh viên trong mảng(cho khoảng 5 sinh viên) sau mỗi 1,2s. Tạo Thread B hiển thị ngẫu nhiên hành động (cho khoảng 5 hành động) sau mỗi 1,2s. Tạo MainClass có hàm main() và khởi chạy đồng thời 2 thread trên.

```
String[] arrSinhVien = new String[]{"Hoàng", "Tuấn", "Quỳnh", "Trang", "Vũ"};
```

```
String[] arrHanhDong = new String[]{"...đang ăn", "...đang ngủ", "...đang xem phim",  
"...đang làm bài tập", "...đang học"};
```