**Tạo và quản lý luồng trong Java**

**1. Tạo luồng bằng cách kế thừa từ lớp Thread**

Để tạo luồng bằng cách kế thừa từ lớp Thread, chúng ta phải tạo một lớp kế thừa từ lớp Thread. Như đã nói trong bài trước, Thread là một lớp có thể tạo ra 1 lớp chạy đa tiến trình được. Trong ví dụ dưới đây tôi tạo ra một lớp có tên là MyThread kế thừa từ lớp Thread và dĩ nhiên lúc này lớp MyThread sẽ là một lớp có thể tạo luồng được.

**MyThread.java**

|  |  |
| --- | --- |
| 1  2  3  4  5 | package vidu;    public class MyThread extends Thread {    } |

Sau đó chúng ta sẽ tiến hành ghi đè phương thức run() của lớp Thread. Những gì có trong phương thức run() này sẽ được thực thi khi luồng bắt đầu chạy. Để ghi đè phương thức này, chúng ta vào Source → Override/Implement Methods, sau đó chọn phương thức run() và nhấn OK để kết thúc. Lúc này lớp MyThread sẽ có nội dung như sau:

**Java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | package vidu;    public class MyThread extends Thread {        @Override      public void run() {          // TODO Auto-generated method stub          super.run();      }    } |

Trong phương thức run(), để đơn giản tôi sẽ viết 1 vòng lặp for duyệt i từ 0 đến 4. Mỗi lần chạy sẽ hiển thị tên luồng đang chạy thông qua dòng System.out.println(Thread.currentThread().getName()); với kết quả hiển thị không giống nhau. Tức là luồng trong Java sẽ được chạy dưới dạng bất đồng bộ, chúng ta không biết được luồng nào chạy trước và luồng nào chạy sau (*phụ thuộc vào hệ điều hành. Hệ điều hành sẽ quyết định tiến trình nào được chạy trước và với mỗi lần chạy thì kết quả sẽ khác nhau, do đó đối với đa luồng thì chúng ta rất khó sửa lỗi*).

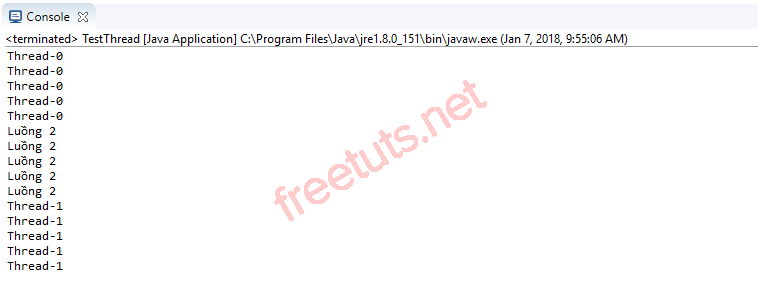
**MyThread.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | package vidu;    public class MyThread extends Thread {        @Override      public void run() {          super.run();          for (int i = 0; i < 5; i++) {              // Thread.currentThread().getName(): cho chúng ta biết tên luồng đang chạy              // và tên luồng này có thể thay đổi được.              System.out.println(Thread.currentThread().getName());          }      }    } |

**TestThread.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | package vidu;    public class TestThread {        public static void main(String[] args) {          // Tạo ra luồng myThread0 từ lớp MyThread          MyThread myThread0 = new MyThread();          myThread0.start();  // kích hoạt luồng            // Tạo ra luồng myThread1 từ lớp MyThread          MyThread myThread1 = new MyThread();          myThread1.start();            // Tạo ra luồng myThread2 từ lớp MyThread          MyThread myThread2 = new MyThread();          myThread2.setName("Luồng 2");   // thay đổi tên luồng thành Luồng 2          myThread2.start();      }    } |

Kết quả sau khi biên dịch chương trình:



Trong lớp TestThread tôi có dòng code myThread2.setName("Luồng 2");, đây là dòng lệnh dùng để thay đổi tên của luồng myThread2 thành "Luồng 2", mặc định thì tên gọi của mỗi luồng sẽ là Thread-số thứ tự, với số thứ tự của luồng được bắt đầu từ 0.

**Lưu ý:** Mọi câu lệnh, nhiệm vụ mà chúng ta muốn luồng thực thi thì chúng ta phải khai báo trong phương thức run() (*trong ví dụ này thì nhiệm vụ của mỗi luồng là hiển thị tên của luồng đó 5 lần*) nhưng nếu chúng ta muốn thực thi luồng đó thì chúng ta phải gọi phương thức start(). Phương thức start() là phương thức dùng để cấp phát tài nguyên cho luồng rồi mới gọi phương thức run() để chạy. Nếu chúng ta không gọi phương thức start() thì những câu lệnh có trong run() sẽ không được chạy.

**2. Tạo luồng bằng cách implement Interface Runnable**

Để tạo luồng bằng cách implement Interface Runnable, chúng ta phải tạo một lớp implement Interface này. Trong ví dụ dưới đây tôi tạo ra một lớp có tên là DemoThread implement Interface Runnable và dĩ nhiên lúc này lớp DemoThread sẽ là một lớp có thể tạo luồng được. **DemoThread.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | package vidu;    public class DemoThread implements Runnable {        @Override      public void run() {          // TODO Auto-generated method stub        }    } |

Tương tự như khi chúng ta tạo luồng bằng cách kế thừa từ lớp Thread thì khi tạo luồng bằng cách implement Interface Runnable thì chúng ta cũng có phương thức run() và những gì có trong phương thức run() này sẽ được thực thi khi luồng bắt đầu chạy, chỉ khác là khi tạo từ lớp Thread thì chúng ta phải tiến hành override lại phương thức này còn đối với tạo từ Runnable thì phương thức run() này đã được tự động override lại.

Trong phương thức run() của lớp DemoThread chúng ta thêm vào đoạn code như sau:

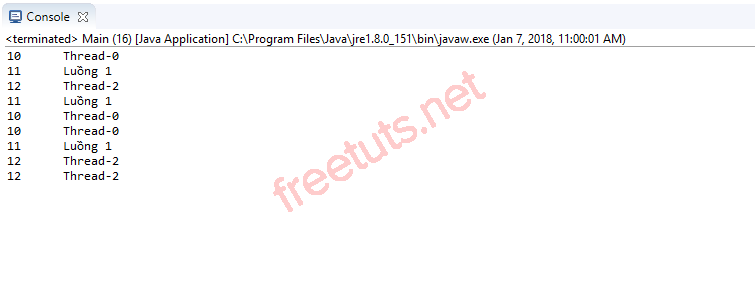
**DemoThread.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | package vidu;    public class DemoThread implements Runnable {        @Override      public void run() {          for (int i = 0; i < 3; i++) {              // Thread.currentThread().getId(): lấy id của luồng đang chạy              // nó dùng để phân biệt với các luồng khác cùng tiến trình hoặc cùng  tập luồng.              // Đây là thông số mà máy ảo java tự tạo ra khi ta tạo luồng              // nên ta không thể sửa đổi cũng như áp đặt thông số này khi tạo luồng.              System.out.println(Thread.currentThread().getId() + "\t" +                      Thread.currentThread().getName());          }      }    } |

**Main.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | package vidu;    public class Main {        public static void main(String[] args) {          DemoThread demoThread0 = new DemoThread();          Thread thread0 = new Thread(demoThread0);          thread0.start();            DemoThread demoThread1 = new DemoThread();          Thread thread1 = new Thread(demoThread1);          thread1.setName("Luồng 1");          thread1.start();            DemoThread demoThread2 = new DemoThread();          Thread thread2 = new Thread(demoThread2);          thread2.start();      }    } |

Kết quả sau khi biên dịch chương trình:



Chúng ta sẽ sử dụng cách tạo luồng bằng cách implement Interface Runnable khi chúng ta muốn chia sẻ thuộc tính giữa các luồng trong chương trình. Ví dụ dưới đây sé minh họa điều này:

**ShareThread.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | package vidu;    public class ShareThread implements Runnable {        private int shareVariable = 0;  // thuộc tính sử dụng chung        public int getShareVariable() {          return shareVariable;      }        @Override      public void run() {          for (int i = 0; i < 3; i++) {              System.out.println("ID:" + Thread.currentThread().getId() +                      ", Name: " + Thread.currentThread().getName()                      + ", shareVariable = " + shareVariable);              shareVariable += 2;          }      }    } |

**TestShareThread.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | package vidu;    public class TestShareThread {        public static void main(String[] args) {            ShareThread shareThread = new ShareThread();            Thread thread0 = new Thread(shareThread);          thread0.setName("Luồng 1");          thread0.start();            Thread thread1 = new Thread(shareThread);          thread1.setName("Luồng 2");          thread1.start();            Thread thread2 = new Thread(shareThread);          thread2.setName("Luồng 3");          thread2.start();            System.out.println("Giá trị thuộc tính shareVariable = " + shareThread.getShareVariable());      }    } |

Kết quả sau khi biên dịch chương trình:

