第十七讲--并发编程基础

任务目标

- ✓ Thread类
- ✓ Runnable接口
- ✓ synchronized线程同步

相关知识

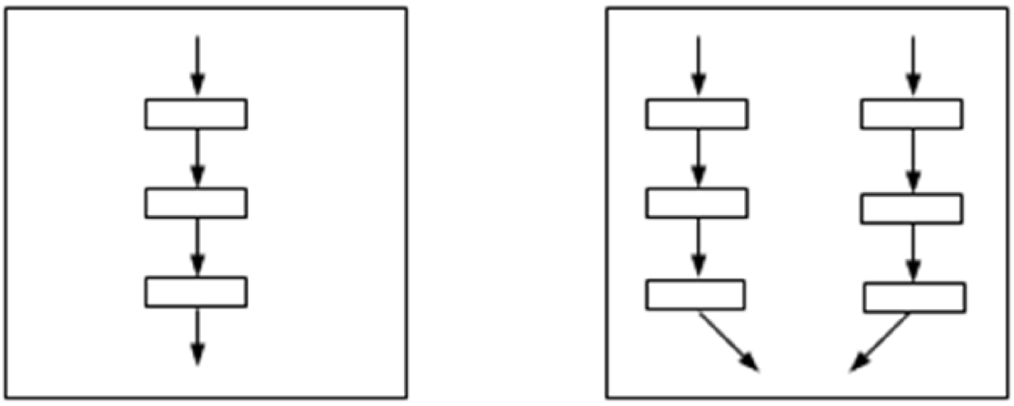
1、多线程与并发

2、线程创建的方式

3、线程的同步

1、多线程简介

1、单线程与多线程示意图。



2、创建线程

继承Thread类的子类，需要重新覆盖run()方法。

- ✓ public abstract void run()

```
class Thread1 extends Thread
{
    public void run()
    {
        int i = 1;
        int sum1 = 0;
        while(i<=20)
        {
            sum1+=i;
            System.out.println(sum1);
            i++;
        }
    }
}

public class TestThread
{
    public static void main(String[] args)
    {
        Thread1 t1 = new Thread1();
        Thread1 t2 = new Thread1();
        t1.start();
        t2.start();
    }
}
```

- [x] 实现Runnable接口的类不应该定义start()方法

```
class Thread1 implements Runnable
{
    public void run()
    {
        int i = 1;
        int sum1 = 0;
        while(i<=10)
        {
            sum1+=i;
            i++;
            System.out.println(sum1+"++++");
            try {
                Thread.sleep(5);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            System.out.print(Thread.currentThread().getName()+"\n");
        }
    }
}

class Thread2 implements Runnable
{
    public void run()
    {
        int i = 1;
        int product = 1;
        while(i<=10)
        {
            product*=i;
            i++;
            System.out.println(product+"***");
            try {
                Thread.sleep(12);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            System.out.print(Thread.currentThread().getName()+"\n");
        }
    }
}

public class ThreadTest {
    public static void main(String[] args)
    {
        Thread1 task1 = new Thread1();
        Thread t1 = new Thread(task1,"A线程");
        Thread2 task2 = new Thread2();
        Thread t2 = new Thread(task2,"B线程");
        t1.start();
        t2.start();
    }
}
```

- [x] setPriority(int x)， 数值越大，优先级越高

```
public class ThreadTest {

    public static void main(String[] args)
    {
        Thread1 task1 = new Thread1();
        Thread t1 = new Thread(task1,"A线程");
        Thread2 task2 = new Thread2();
        Thread t2 = new Thread(task2,"B线程");

        t1.setPriority(1);
        t2.setPriority(10);
        t1.start();
        t2.start();
    }
}
```

2、生产者-消费者模型（线程协调）

1、synchronized关键词可以协调线程的同步

```
class Box
{   private int data;
    public synchronized void put(int v)
    {       data=v;    }
    public synchronized int get()
    {        return data;    }
}
class Producer extends Thread
{
    private Box b;
    Producer(Box b)
    {
    this.b=b;
    }
    public void run()
    {
        for(int i=0;i<10;i++)
        {
            this.b.put(i);
            System.out.println("Producer put"+i);
```

```java
            try
            {sleep((int)(Math.random()*20));}
            catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
}

class Customer extends Thread
{
    private Box b;
    Customer(Box b)
    {
        this.b=b;
    }
    public void run()
    {
        for(int i=0;i<10;i++)
        {
            int value = 0;
            value = this.b.get();
            System.out.println("Customer get"+value);
            try
            {sleep((int)(Math.random()*20));}
            catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
}


public class ProducerCustomer {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Box b = new Box();
        Producer p = new Producer(b);
        Customer c = new Customer(b);
        p.start();
        c.start();
    }
}
```