实验 4 Java 多线程编程

一、实验目的

1、理解线程概念和定义；
2、掌握创建、管理和控制 Java 线程对象的方法；
3、掌握实现线程互斥和线程同步的方法；
4、了解多线程编程模型。

二、实验内容

1、Thread 类现多线程

(1)定义线程类

```
class MyThread extends Thread {                    // 线程的主体类
    private String title;                          // 成员属性
    public MyThread(String title) {                // 属性初始化
        this.title = title;
    }
    @Override
    public void run() {                            // 【方法覆写】线程方法
        for (int x = 0; x < 10; x++) {
            System.out.println(this.title + "运行，x = " + x);
        }
    }
}
```

(2)多线程启动

```
public class ThreadDemo {
    public static void main(String[] args) {
        new MyThread("线程A").start() ;            // 实例化线程对象并启动
        new MyThread("线程B").start() ;            // 实例化线程对象并启动
        new MyThread("线程C").start() ;            // 实例化线程对象并启动
    }
}
```

2、Runnable 接口实现多线程

(1)定义线程

```
class MyThread implements Runnable {               // 线程的主体类
    private String title;                          // 成员属性
    public MyThread(String title) {                // 属性初始化
        this.title = title;
    }
    @Override
```

```
    public void run() {                                 // 【方法覆写】线程方法
        for (int x = 0; x < 10; x++) {
            System.out.println(this.title + "运行，x = " + x);
        }
    }
}
```

(2)启动线程

```
public class ThreadDemo {
    public static void main(String[] args) {
        Thread threadA = new Thread(new MyThread("线程对象A")) ;
        Thread threadB = new Thread(new MyThread("线程对象B")) ;
        Thread threadC = new Thread(new MyThread("线程对象C")) ;
        threadA.start();                                    // 启动多线程
        threadB.start();                                    // 启动多线程
        threadC.start();                                    // 启动多线程
    }
}
```

3、Callable 接口实现多线程

(1)定义线程主体类

```
class MyThread implements Callable<String> {            // 定义线程主体类
    @Override
    public String call() throws Exception {             // 线程执行方法
        for (int x = 0; x < 10; x++) {
            System.out.println("********* 线程执行、x = " + x);
        }
        return "www.mldn.cn";                           // 返回结果
    }
}
```

(2)启动线程并获取 Callable 返回值

```
public class ThreadDemo {
    public static void main(String[] args) throws Exception {
        // 将Callable实例包装在FutureTask类之中，这样可以就可以与Runnable接口关联
        FutureTask<String> task = new FutureTask<>(new MyThread()) ;
        new Thread(task).start();                           // 线程启动
        System.out.println("【线程返回数据】" + task.get());// 获取返回结果
    }
}
```

4、并发资源访问

```
class MyThread implements Runnable {                // 线程的主体类
    private int ticket = 5;                          // 定义总票数
    @Override
    public void run() {                              // 线程的主体方法
        for (int x = 0; x < 100; x++) {              // 进行100次的卖票处理
            if (this.ticket > 0) {                   // 有剩余票
                System.out.println("卖票，ticket = " + this.ticket--);
            }
        }
    }
}
public class ThreadDemo {
    public static void main(String[] args) {
        MyThread mt = new MyThread();                // 定义资源对象
        new Thread(mt).start();                      // 第一个线程启动
        new Thread(mt).start();                      // 第二个线程启动
        new Thread(mt).start();                      // 第三个线程启动
    }
}
```

三、按照要求编写程序（二选一）

1、多线程排序，即把数据存放在一维数组中，首先对数据进行分段，接着对每一段数据采用经典排序算法实现排序，最后把各段数据进行合并排序。请完成程序编写。

2、多线程求数组最大值，即把数据存放在一维数组中，首先对数据进行分段，接着对每一段数据求得最大值，最后把各段数据最大值进行比较从而得出整个数组的最大值。请完成程序编写。

3，采用线程实现“生产者-消费者”编程的基础模型。

四、实验结果

写实验报告。内容包括：

1. 习题的运行结果，源程序。
2. 程序调试中出现的错误提示。（英文、中文对照）

若有没通过的程序，分析原因。