深 圳 大 学 实 验 报 告

课程名称：Java 程序设计

实验项目名称：必实验 4 线程应用

学院：计算机与软件学院

专业：计算机科学与技术

指导教师：潘微科

报告人：黎浩然 学号：2018112061 班级：01

实验时间：2019 年 11 月 08 日（周五）-2019 年 11 月 20 日（周三）

实验报告提交时间：_____

教务部制

实验目的与要求：

实验目的： 掌握 Java 程序设计中的线程同步等技术。

实验要求：

(1). 运行以下三个程序（每个程序运行 10 次），并对输出结果给出分析。在报告中附上程序截图和详细的文字说明。（15 分）

程序 1:

```
// The task for printing a character a specified number of times
class PrintChar implements Runnable {
  private char charToPrint; // The character to print
  private int times; // The number of times to repeat

  /** Construct a task with a specified character and number of
   *  times to print the character
   */
  public PrintChar(char c, int t) {
    charToPrint = c;
    times = t;
  }

  @Override /** Override the run() method to tell the system
   *  what task to perform
   */
  public void run() {
    for (int i = 0; i < times; i++) {
      System.out.print(charToPrint);
    }
  }
}


// The task class for printing numbers from 1 to n for a given n
class PrintNum implements Runnable {
  private int lastNum;

  /** Construct a task for printing 1, 2, ..., n */
  public PrintNum(int n) {
    lastNum = n;
  }

  @Override /** Tell the thread how to run */
  public void run() {
    for (int i = 1; i <= lastNum; i++) {
      System.out.print(" " + i);
    }
  }
}
```

```
public class TaskThreadDemo {
  public static void main(String[] args) {
    // Create tasks
    Runnable printA = new PrintChar('a', 100);
    Runnable printB = new PrintChar('b', 100);
    Runnable print100 = new PrintNum(100);

    // Create threads
    Thread thread1 = new Thread(printA);
    Thread thread2 = new Thread(printB);
    Thread thread3 = new Thread(print100);

    // Start threads
    thread1.start();
    thread2.start();
    thread3.start();
  }
}
```

程序 2:

```
// The task for printing a character a specified number of times
class PrintChar implements Runnable {
  private char charToPrint; // The character to print
  private int times; // The number of times to repeat

  /** Construct a task with a specified character and number of
   *  times to print the character
   */
  public PrintChar(char c, int t) {
    charToPrint = c;
    times = t;
  }

  @Override /** Override the run() method to tell the system
   *  what task to perform
   */
  public void run() {
    for (int i = 0; i < times; i++) {
      System.out.print(charToPrint);
    }
  }
}


// The task class for printing numbers from 1 to n for a given n
class PrintNum implements Runnable {
  private int lastNum;

  /** Construct a task for printing 1, 2, ..., n */
  public PrintNum(int n) {
    lastNum = n;
  }

  @Override /** Tell the thread how to run */
  public void run() {
    for (int i = 1; i <= lastNum; i++) {
      System.out.print(" " + i);
    }
  }
}
```

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

public class ExecutorDemo {
  public static void main(String[] args) {
    // Create a fixed thread pool with maximum three threads
    ExecutorService executor = Executors.newFixedThreadPool(3);

    // Submit runnable tasks to the executor
    executor.execute(new PrintChar('a', 100));
    executor.execute(new PrintChar('b', 100));
    executor.execute(new PrintNum(100));

    // Shut down the executor
    executor.shutdown();
  }
}
```

程序 3:

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

public class AccountWithoutSync {
  private static Account account = new Account();

  public static void main(String[] args) {
    ExecutorService executor = Executors.newCachedThreadPool();

    // Create and launch 100 threads
    for (int i = 0; i < 100; i++) {
      executor.execute(new AddAPennyTask());
    }

    executor.shutdown();

    // Wait until all tasks are finished
    while (!executor.isTerminated()) {
    }

    System.out.println("What is balance? " + account.getBalance());
  }

  // A thread for adding a penny to the account
  private static class AddAPennyTask implements Runnable {
    public void run() {
      account.deposit(1);
    }
  }

  // An inner class for account
  private static class Account {
    private int balance = 0;

    public int getBalance() {
      return balance;
    }

    public void deposit(int amount) {
      int newBalance = balance + amount;

      // This delay is deliberately added to magnify the
      // data-corruption problem and make it easy to see.
      try {
        Thread.sleep(5);
      }
      catch (InterruptedException ex) {
      }

      balance = newBalance;
    }
  }
}
```

(2). 编写 Java 应用程序实现如下功能：第一个线程生成一个随机数，第二个线程每隔一段时间读取第一个线程生成的随机数，并判断它是否是奇数。要求采用实现 Runnable 接口和 Thread 类的构造方法的方式创建线程，而不是通过 Thread 类的子类的方式。在报告中附上程序截图、完整的运行结果截图和简要文字说明。（20 分）

(3). 编写 Java 应用程序实现如下功能：第一个线程输出数字 1-26，第二个线程输出字母 A-Z，输出的顺序为 1A2B3C...26Z，即每 1 个数字紧跟着 1 个字母的方式。要求线程间实现通信。要求采用实现 Runnable 接口和 Thread 类的构造方法的方式创建线程，而不是通过 Thread 类的子类的方式。在报告中附上程序截图、运行结果截图和详细的文字说明。（20 分）

(4). 编写 Java 应用程序实现如下功能：创建工作线程，模拟银行现金账户存款操作。多个线程同时执行存款操作时，如果不使用同步处理，会造成账户余额混乱，要求使用 syncrhonized 关键字同步代码块，以保证多个线程同时执行存款操作时，银行现金账户存款的有效和一致。要求采用实现 Runnable 接口和 Thread 类的构造方法的方式创建线程，而不是通过 Thread 类的子类的方式。在报告中附上程序截图、运行结果截图和详细的文字说明。（25 分）

报告写作。要求：主要思路有明确的说明，重点代码有详细的注释，行文逻辑清晰可读性强，报告整体写作较为专业。**（20 分，这一项的评分，采用 20 分、10 分和 0 分三个级别）**

说明：

（1）本次实验课作业满分为 100 分，占总成绩的比例（待定）。

（2）本次实验课作业截至时间 2019 年 11 月 20 日（周三）23:59。

（3）报告正文：请在指定位置填写，本次实验不需要单独提交源程序文件。

（4）个人信息：WORD 文件名中的“姓名”、“学号”，请改为你的姓名和学号；实验报告的首页，请准确填写“学院”、“专业”、“报告人”、“学号”、“班级”、“实验报告提交时间”等信息。

（5）提交方式：截至时间前，请在 Blackboard 平台中提交。

（6）发现抄袭（包括复制&粘贴整句话、整张图），该次作业记零分。

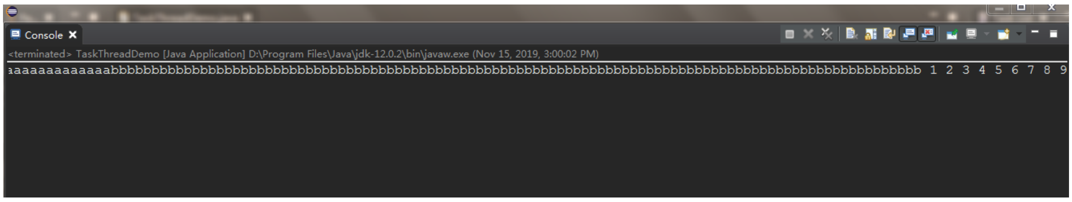
（7）延迟提交，不得分；如有特殊情况，请于截至日期之后 48 小时内发邮件到 panweike@szu.edu.cn，并在邮件中注明课程名称、作业名称、姓名、学号等信息，以及特殊情况说明，我收到后会及时回复。

（8）期末考试阶段补交无效。

(1). 运行以下三个程序（要求每个程序运行 10 次），并对输出结果给出分析。在报告中附上程序截图和详细的文字说明。（15 分）

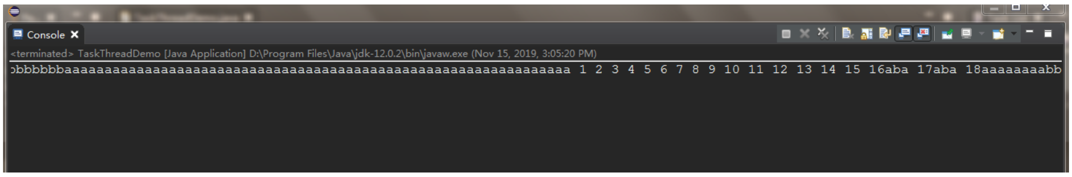
程序 1:

第一次结果



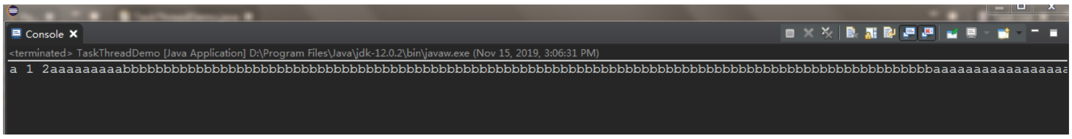
结果分析：程序先执行完第一个线程（打印 100 个 a），再执行第二个线程（打印 100 个 b），最后才执行第三个线程。

第二次结果



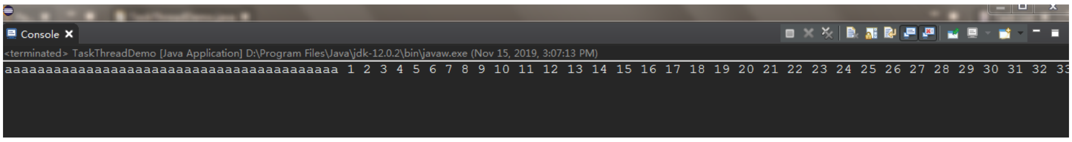
结果分析：程序在 3 个线程之间来回切换

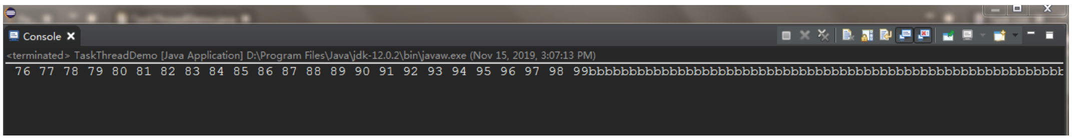
第三次结果



结果分析：程序在 3 个线程之间来回切换

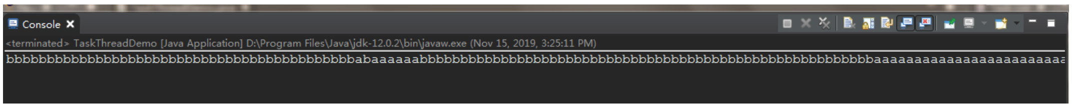
第四次结果

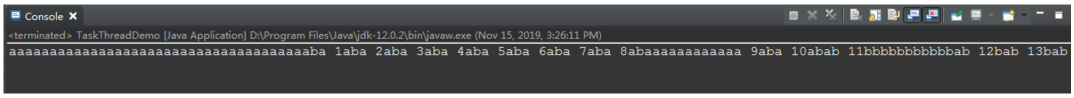


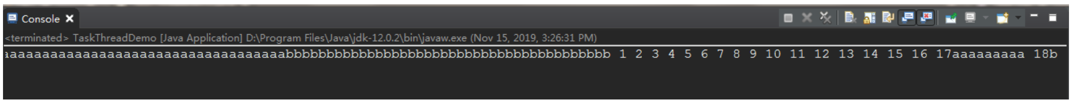


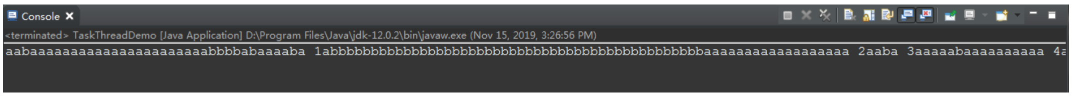
结果分析：程序先执行完第一个线程，再执行完第三个线程，最后才执行第二个线程。

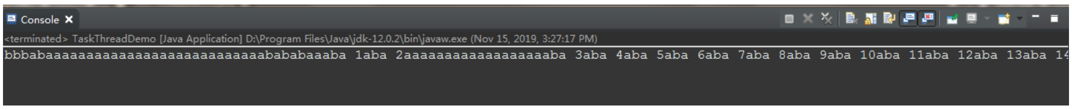
程序 2:











结果分析：程序在三个线程间来回切换

程序 3:

第三个程序：

第一次运行结果：　　　　　　　　　　第二次运行结果

```
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
```

```
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 1
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
```

```
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
```

```
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
What's the balance? 2
```

结果分析：由于线程在程序中随机切换，所以每个线程读取到的 balance 可能是已更新的 balance，也可能是未更新的 balance，从而导致有大量的重复数字（balance）出现。

(2). 编写 Java 应用程序实现如下功能：第一个线程生成一个随机数，第二个线程每隔一段时间读取第一个线程生成的随机数，并判断它是否是奇数。要求采用实现 Runnable 接口和 Thread 类的构造方法的方式创建线程，而不是通过 Thread 类的子类的方式。在报告中附上程序截图、完整的运行结果截图和简要文字说明。（20 分）

```
public class Proj {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Thread thread1 = new Thread(new GenerateRan());
        Thread thread2 = new Thread(new CheckIfOdd());
        thread1.start();
        thread2.start();
    }
}

class GenerateRan implements Runnable{   //生成随机数
    public static int num = 1;
    public void run() {
        while(true) {
            num = (int)(1+Math.random()*(100));   //产生1-100的整数
            try {
                Thread.sleep(1000); //休息一秒钟
            }
            catch(InterruptedException e) {}
            if(CheckIfOdd.times==5)      //如果已经检查五次，则结束
                break;
        }
    }
}

class CheckIfOdd implements Runnable{
    public static int times = 1;     //表示执行的次数
    public void run() {
        while(times++<=5) { //5次后结束
            if(GenerateRan.num%2==1)
                System.out.println(GenerateRan.num + " is odd.");
            else
                System.out.println(GenerateRan.num + " is even.");
            try {
                Thread.sleep(1000); //休息一秒钟
            }
            catch(InterruptedException e) {}
        }
    }
}
```

运行结果如下：

结果陈述：每隔一秒钟生成一个 1-100 的整数，然后另一个线程每隔一秒钟读取此整数并判断是否为奇数。（为了防止程序在输出结果时在线程间切换，生成随机数的线程也休息）

```
1 is odd.
24 is even.
27 is odd.
97 is odd.
97 is odd.
```

(3). 编写 Java 应用程序实现如下功能：第一个线程输出数字 1-26，第二个线程输出字母 A-Z，输出的顺序为 1A2B3C...26Z，即每 1 个数字紧跟着 1 个字母的方式。要求线程间实现通信。要求采用实现 Runnable 接口和 Thread 类的构造方法的方式创建线程，而不是通过 Thread 类的子类的方式。在报告中附上程序截图、运行结果截图和详细的文字说明。（20 分）

源程序如下：

打印数字的任务：

```
class PrintNum implements Runnable{
    private Object obj;

    public PrintNum(Object obj) {
        this.obj = obj;
    }

    public void run() {
        synchronized(obj) {
            for (int i = 1; i < 27; i++) {
                System.out.print(i);
                    obj.notifyAll();     //通知其他线程
                    try {
                        obj.wait();      //进入等待
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
            }
        }
    }
}
```

打印字母的任务：

```
class PrintChar extends Thread {
    private Object obj;

    public PrintChar(Object obj) {
        this.obj = obj;
    }

    public void run() {
        synchronized (obj) {
            for (int i = 0; i < 26; i++) {
                System.out.print((char) ('A' + i));
                obj.notifyAll();     //通知其他线程
                try {
                    obj.wait();      //进入等待
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

主线程：

```
public class Task {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Object obj = new Object();       //使用同一个对象以实现通信
        Thread t1 = new Thread(new PrintNum(obj));
        Thread t2 = new Thread(new PrintChar(obj));

        t1.start();      //启动线程
        t2.start();
    }

}
```

运行结果如下：

```
1A2B3C4D5E6F7G8H9I10J11K12L13M14N15O16P17Q18R19S20T21U22V23W24X25Y26Z
```

(4). 编写 Java 应用程序实现如下功能：创建工作线程，模拟银行现金账户存款操作。多个线程同时执行存款操作时，如果不使用同步处理，会造成账户余额混乱，要求使用 syncrhonized 关键字同步代码块，以保证多个线程同时执行存款操作时，银行现金账户存款的有效和一致。要求采用实现 Runnable 接口和 Thread 类的构造方法的方式创建线程，而不是通过 Thread 类的子类的方式。在报告中附上程序截图、运行结果截图和详细的文

字说明。（25 分）

源程序如下：

```
class Account {
    private double balance; //账户原始金额

    Account(double b){
        this.balance = b;
    }

    public synchronized void save(double d){     //不能同时存钱
        balance += d;
        System.out.println("Save: "+d+"; balance is "+balance);
    }
}

class SaveMoney implements Runnable{
    private Account a;
    SaveMoney(int money){
        a = new Account(money);
    }
    public void run() {
        a.save(Integer.parseInt(Thread.currentThread().getName()));
    }    //根据线程名称确定存入金额
}


public class Demo {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        SaveMoney save = new SaveMoney(1000);
        Thread [] thread = new Thread[20];
        for(int i = 0; i < 20; i++) //创建20个线程，每一个存的金额都不一样
        {
            thread[i] = new Thread(save);
            thread[i].setName(String.valueOf((i+1)*1000));
        }
        for(int i = 0;i < 20; i++)
            thread[i].start();
    }

}
```

运行结果如下：

```
Save: 1000.0; balance is 2000.0
Save: 20000.0; balance is 22000.0
Save: 19000.0; balance is 41000.0
Save: 18000.0; balance is 59000.0
Save: 17000.0; balance is 76000.0
Save: 16000.0; balance is 92000.0
Save: 15000.0; balance is 107000.0
Save: 14000.0; balance is 121000.0
Save: 13000.0; balance is 134000.0
Save: 12000.0; balance is 146000.0
Save: 11000.0; balance is 157000.0
Save: 10000.0; balance is 167000.0
Save: 9000.0; balance is 176000.0
Save: 8000.0; balance is 184000.0
Save: 7000.0; balance is 191000.0
Save: 6000.0; balance is 197000.0
Save: 5000.0; balance is 202000.0
Save: 4000.0; balance is 206000.0
Save: 3000.0; balance is 209000.0
Save: 2000.0; balance is 211000.0
```

可见，运行结果是正确的。

++

其他（例如感想、建议等等）。

1. Java 提供了多线程程序设计。
2. 一般情况下，线程的执行顺序是由 CPU 调度的，是不可预测的。
3. 通过实现线程同步或者线程通信可以解决上述问题。

指导教师批阅意见：   成绩评定：   指导教师签字：   2019 年　　月　　日
备注：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。