《Java 程序设计》

作业报告

作业名称：作业 1 限时手写代码编程

授课教师：毛斐巧

报告人：吴嘉楷 学号：2022150168 班级：国际班

报告提交时间：2024 年 11 月 18 日

成　绩：

1.作业内容与要求：

限时手写代码编程：

(1) 编写 Java 程序：验证 1024（含 1024）以内的数是否满足“卡拉兹猜想”，若有不满足“卡拉兹猜想”的数，请打印出来。所谓“卡拉兹猜想”是指对任何一个自然数 n，如果它是偶数，那么把它砍掉一半，如果它是奇数，那么把(3n+1)砍掉一半，这样一直反复砍下去，最后一定在某一步得到 1。

(2) 编写 Java 程序：第一个线程输出数字 1-13，第二个线程输出字母 A-Z，输出的顺序为 1AB2CD3EF...13YZ，即每 1 个数字（如 1）后紧跟着 2 个字母（如 AB）。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。在主方法中创建对象，测试效果。

要求：

（A） 计时手写代码编写符合上述两道题目要求的程序。建议阅读分析各题编程要求，拟定编程思路，然后，在一空白纸上手写出程序代码，并记录手写该题代码所花时间。将手写出的代码拍照贴在该题“手写代码”位置，要求图片中的源代码清晰可见，无遗漏。**若给出的是机器输入程序而非本人手写代码，该编码部分计 0 分。**

（B） 在任意一个支持 java 程序开发的集成开发环境中输入自己的“手写代码”（与“手写代码”完全一致，不能修改），截图并在“机器输入程序代码”位置给出在集成开发环境中输入的“机器输入程序”，并运行程序给出程序的实际运行结果。“机器输入程序”必须与“手写代码”完全一致，并据实给出编译运行结果。比如，可能无法运行，有错误提示或警告等，应据实截图给出实际运行情况。

（C） 修改程序给出程序正确运行结果。根据错误提示，直接在开发环境中修改程序，直到程序能够正确运行，给出正确运行结果截图。要求在“**程序修改与正确运行结果**”位置指示有哪些错误，进行了哪些修改，并贴出正确运行的结果截图。

2.解答报告正文

（1）题 1 编程实现如下：

（1-1）编程思路：

首先在 main 方法中初始化一个布尔变量 flag，用于判断 1 到 1024 的所有自然数是否都满足卡拉兹猜想。这个 flag 变量最初被设为 true，表示假设所有数字都符合猜想的规则。

接着，在 main 方法中，使用一个 for 循环，从 1 遍历到 1024，将每个数字传递给 isKalaz 方法进行验证。如果 isKalaz 返回 false，表示发现不满足猜想的数字，程序会输出该数字，并将 flag 设为 false。这样，如果有任何数字不符合卡拉兹猜想的规则，程序就会标记这一点。最终，根据 flag 的状态，如果所有数字都能通过验证，程序会输出“1024 以内所有数都满足卡拉兹猜想”；否则，会列出不满足猜想的数字。

isKalaz 方法的作用是检查给定的数字 n 是否满足卡拉兹猜想。方法中，利用 while 循环不断对 n 进行处理：如果 n 是偶数，就将其除以 2；如果 n 是奇数，则按照卡拉兹猜想的规则计算 (3 * n + 1) / 2。这个过程会持续进行，直到 n 的值变为 1，最终返回 true，表示这个数字 n 满足猜想。

（1-2）手写代码：

```
public class demo1 {
    public static void main(String[] args) {
        boolean flag = true;
        for (int i=1; i<=1024; i++){
            if (!isKalaz(i)){
                System.out.println("1024以内不满足卡拉兹猜想的数:"+i);
                flag = false;
            }
        }
        if (flag) System.out.println("1024以内的有数都满足卡拉兹猜想");
    }
    public boolean isKalaz(int n){
        while(n!=1){
            if(n%2==0){
                n /= 2;
            }else{
                n = (3*n+1)/2;
            }
        }
        return true;
    }
}
```

图 1 手写代码图片（一）

（1-3）手写代码完成时间：

时间段：15:35 ~ 15:42

共用时：7 分钟

（1-4）机器输入程序代码：

Main.java　demo1.java

```
1  public class demo1 {
2      public static void main(String[] args) {
3          boolean flag = true;
4          for (int i = 1; i <= 1024 ; i++) {
5              if(!isKalaz(i)){
6                  System.out.println("1024以内不满足卡拉兹猜想的数: " + i);
7                  flag = false;
8              }
9          }
10         if (flag) System.out.println("1024以内所有数都满足卡拉兹猜想");
11     }
       1 usage
12     public boolean isKalaz(int n){
13         while (n!=1){
14             if (n % 2 == 0){
15                 n /= 2;
16             }else {
17                 n = (3 * n + 1) / 2;
18             }
19         }
20         return true;
21     }
22 }
```

图 2 机器输入程序截图（一）

（1-5）手写代码的编译运行情况：

图 3 编译运行结果（一）

由上图可见，手写的代码程序存在错误，错误之处在于 isKalaz 方法没有定义成 static 静态方法，导致 main 函数无法正常调用 isKalaz 方法。

这是因为，在 Java 中，static 方法只能直接调用其他 static 方法或访问 static 变量。因为 static 方法属于类本身，而不是类的某个特定实例。因此，static 方法无法访问非 static（即实例级别）成员，因为它们不依赖于特定对象的实例。

（1-6）程序修改与正确运行结果：

程序修改：

将 isKalaz(int n)函数定义为 static 修饰的方法即可，如下图所示：

```
public class demo1 {
    public static void main(String[] args) {
        boolean flag = true;
        for (int i = 1; i <= 1024 ; i++) {
            if(!isKalaz(i)){
                System.out.println("1024以内不满足卡拉兹猜想的数：" + i);
                flag = false;
            }
        }
        if (flag) System.out.println("1024以内所有数都满足卡拉兹猜想");
    }
    1 usage
    public static boolean isKalaz(int n){
        while (n!=1){
            if (n % 2 == 0){
                n /= 2;
            }else {
                n = (3 * n + 1) / 2;
            }
        }
        return true;
    }
}
```

图 4 程序修改细节（一）

运行结果：（1024 以内所有数都满足卡拉兹猜想）

图 5 正确运行结果（一）

（2）题 2 编程实现如下：

（2-1）编程思路：

首先，定义两个实现类 NumberPrinter 和 LetterPrinter，分别用于打印数字和字母。NumberPrinter 类在其 run()方法中使用一个 for 循环输出从 1 到 13 的数字，而 LetterPrinter 类的 run()方法使用一个 for 循环输出从'A'到'Z'的字母，以两个字母为一组。

然后，在 demo2 测试类中的 main 方法中，通过 Thread 类创建了两个线程 t1 和 t2，分别与 NumberPrinter 和 LetterPrinter 关联。随后调用 start()方法启动这两个线程，使它们并行运行。

事实证明，此思路没有考虑到题目中交替打印数字和字母的要求，这将在后面进行思路以及代码的修改。

（2-2）手写代码：

```
class NumberPrinter implements Runnable{
    @Override
    public void run( ){
        for (int i =1; i <= 13; i++){
            System.out.println(i);
        }
    }
}
class LetterPrinter implements Runnable {
    @Override
    public void run( ){
        for (int i = 65, i <=90; i++){
            System.out.print ((char) i++);
            System.out.print ( (char) i++);
        }
    }
}

class
public class demo2 {
    public static void main (String[] args){
        Thread t1 = new NumberPrinter( );
        Thread t2 = new LetterPrinter( );
        t1.start( );
        t2.start( );
    }
}
```

图 6 手写代码图片（二）

（2-3）手写代码完成时间：

时间段：20:51 ~ 21:03

总用时：12 分钟

（2-4）机器输入程序代码：

```
class NumberPrinter implements Runnable {
    @Override
    public void run() {
        for (int i = 1; i <= 13; i++) {
            System.out.print(i);
        }
    }
}
1 usage
class LetterPrinter implements Runnable {
    @Override
    public void run() {
        for (int i = 65; i <= 90;) {
            System.out.print((char)i++);
            System.out.print((char)i++);
        }
    }
}
public class demo2 {
    public static void main(String[] args) {
        Thread t1 = new NumberPrinter();
        Thread t2 = new LetterPrinter();
        t1.start();
        t2.start();
    }
}
```

图 7　IDEA 输入程序截图（二）

（2-5）手写代码的编译运行情况：

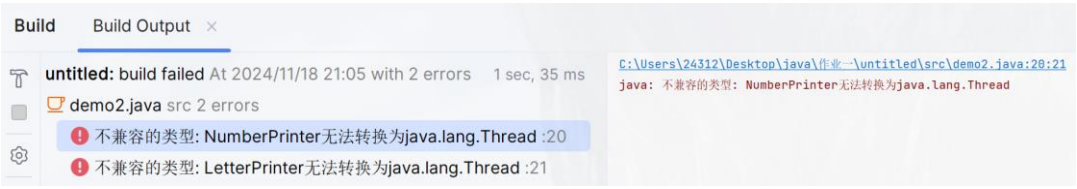

图 8 手写代码运行情况（二）

由图 8 可见，手写的代码犯了一个低级错误：将一个 Runnable 的实现类直接当作一个线程类使用，赋值给了 Thread 类型的变量，导致程序报错。

对于通过实现 Runnable 接口来创建线程的情况而言，必需在创建 Thread 对象时传入 Runnable 实例作为参数，而不能直接将 new 出来的 Runnable 的实现类赋值给 Thread 变量。例如：Thread thread = new Thread(new MyRunnable())。

（2-6）程序修改与正确运行结果：

程序修改一：

将创建的实现 Runnable 接口的实例对象作为参数传递给 new Thread ( )方法，从而创建一个 Thread 类型的线程对象

```
public class demo2 {
    public static void main(String[] args) {
        Thread t1 = new Thread(new NumberPrinter());
        Thread t2 = new Thread(new LetterPrinter());
        t1.start();
        t2.start();
    }
}
```

图 9 程序修改（2.1）

运行结果一：

图 10 程序运行结果（2.1）

由图可见，程序先输出了字母 A~Z，再输出了数字 1~13，而不是 1 个数字与 2 个字母交替出现，这与题目要求不符，这也是我没有充分理解题意的后果。

因此，我们想要进行第二次修改，使得 1 个数字与 2 个字母交替地被输出，最终得到的输出顺序应为 1AB2CD3EF...13YZ。

程序修改二：

为了让 1 个数字与 2 个字母交替地被输出，那么我们需要使用到线程间通信技术及同步机制，简单来说，就是使用 synchronized 关键字来控制线程的同步，并且，使用 wait()、notify()方法来进行线程间的通信。

其中，两个类的同步锁 lock 都使用本 java 文件的 class 即可，这样既保证了这个锁对象的唯一性，又不用考虑如何将一个全局共享的对象作为锁对象并传入两个类中。

```
class NumberPrinter implements Runnable {
    @Override
    public void run() {
        Object lock = demo2.class;
        synchronized (lock) {
            for (int i = 1; i <= 13; i++) {
                System.out.print(i);
                lock.notify();
                try {
                    lock.wait();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
            lock.notify();
        }
    }
}
```

图 11 NumberPrinter 类的修改点

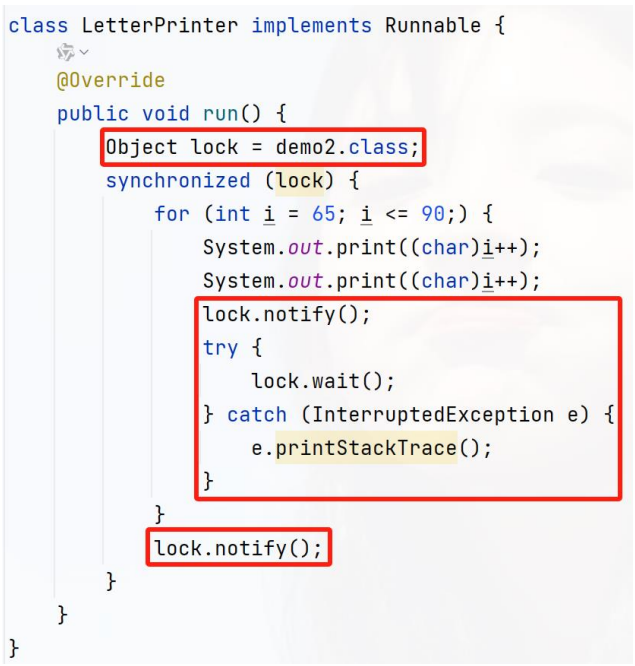

图 12　LetterPrinter 类的修改点

修改后的代码：

```
class NumberPrinter implements Runnable {
    @Override
    public void run() {
        Object lock = demo2.class;
        synchronized (lock) {
            for (int i = 1; i <= 13; i++) {
                System.out.print(i);
                lock.notify();
                try {
                    lock.wait();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
            lock.notify();
        }
    }
}
class LetterPrinter implements Runnable {
    @Override
    public void run() {
        Object lock = demo2.class;
        synchronized (lock) {
            for (int i = 65; i <= 90;) {
                System.out.print((char)i++);
```

```
                System.out.print((char)i++);
                lock.notify();
                try {
                    lock.wait();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
            lock.notify();
        }
    }
}

public class demo2 {
    public static void main(String[] args) {
        Thread t1 = new Thread(new NumberPrinter());
        Thread t2 = new Thread(new LetterPrinter());
        t1.start();
        t2.start();
    }
}
```

修改后的编程思路：

首先，定义两个实现类 NumberPrinter 和 LetterPrinter，我们通过实现 Runnable 接口来定义线程要执行的任务，分别用于打印数字和字母。NumberPrinter 类在其 run()方法中使用一个 for 循环输出从 1 到 13 的数字，而 LetterPrinter 类的 run()方法使用一个 for 循环输出从'A'到'Z'的字母，以两个字母为一组。

为了使数字和字母交替输出，我们在每个线程的执行过程中使用了**同步块**。具体来说，我们通过一个 lock 对象来同步这两个线程。每次一个线程执行完任务后，调用 lock.notify()来通知另一个线程开始执行。紧接着，当前线程通过 lock.wait()进入等待状态，直到另一个线程通过 lock.notify()再次唤醒它。这样，数字线程和字母线程就能按照顺序交替输出。

然后，在 demo2 测试类中的 main 方法中，通过 Thread 类创建了两个线程 t1 和 t2，分别与 NumberPrinter 和 LetterPrinter 关联。

最后，调用 start()方法启动这两个线程，使它们并行运行。

正确运行结果：

图 13 正确运行结果（二）