

Java 第五章 高级语言特征上机报告

22009200601 汤栋文

2023 年 5 月 11 日

目录

1.	个人信息	- 2 -
2.	任务重述	- 2 -
3.	PPT 练习	- 2 -
3.1	StaticMethod on P7	- 2 -
3.2	InheritStaticInit on P21	- 3 -
3.3	BlankFinal on P31	- 3 -
3.4	AbstractInterface on P53	- 3 -
3.5	FindDups on P64	- 4 -
3.6	UseArrayList on P70	- 4 -
3.7	QueueDemo on P74	- 4 -
3.8	Freq on P79	- 4 -
3.9	UseHashMap on P81	- 5 -
3.10	TestIterator on P83	- 5 -
3.11	CoinTest on P94	- 5 -
3.12	AutoBoxingTest on P99	- 5 -
4.	日历程序修改	- 6 -
4.1	修改复数类的继承关系	- 6 -
4.2	实现 Comparable<Complex>接口	- 6 -
4.3	设计并实现静态方法 valueOf	- 6 -
5.	编写图书馆管理系统	- 7 -
5.1	整体架构	- 7 -
5.2	Book 类	- 7 -
5.3	Student 类	- 8 -
5.4	Log 类	- 8 -
5.5	Database 类	- 8 -
5.6	Main 函数和运行结果	- 9 -
6.	总结	- 9 -

1. 个人信息

姓名：汤栋文

学号：22009200601

日期：2023 年 5 月 11 日

2. 任务重述

1. 练习 PPT 中的全部小练习

(页码: 7, 21, 31, 53, 64, 70, 74, 79, 81, 83, 94, 99)

2. 编写一个程序，完善第四章的复数类功能

- 修改复数类的继承关系，使其父类为 `Java.lang.Number` 类，并实现 `Comparable<复数类>` 接口。实现全部继承的方法。
- 设计并实现静态方法 `valueOf(String str)`，效验输入的字符串，合规则返回一个复数对象实例。
- 设计并实现静态的加运算方法，实现对两个各种数值类型数据的加法计算，返回复数对象实例。

3. 编写一个程序，程序提供图书馆管理功能：

(1) 程序提供图书馆馆藏书籍管理功能：

书籍信息包括书名、作者、出版社、及馆藏数量。

书籍可能存在书名相同，作者、出版社不同情况

提供针对书籍书名、作者、出版社的查询功能：书名可进行模糊查询

作者及出版社查询可能返回多条信息结果

可添加书或删除已有书籍信息。

(2) 程序提供图书馆借阅管理功能。

每位同学可同时借阅三本书籍，程序可查询某位同学的借阅情况。

当同学尝试借阅第四本书籍或书籍库存不足时，借阅失败。

当同学归还书籍时，修正同学借阅数据及馆藏书籍数据。

可针对学号查询某位同学的借阅情况。

(3) 初始数据实现。

初始数据可固化在程序代码中。

初始数据可通过数据文件读入。

3. PPT 练习

3.1 StaticMethod on P7

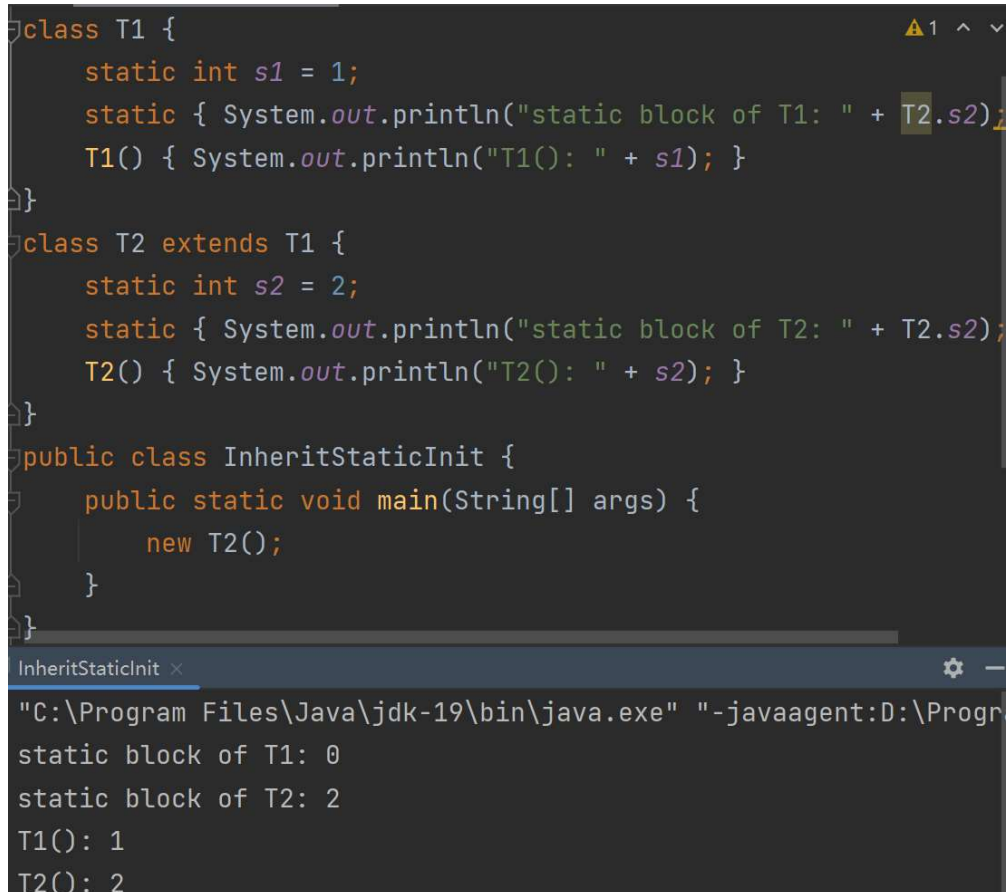
静态变量，当系统加载其所在类的时候分配空间并初始化，就好像是独立于该类以外的函数。静态变量至多初始化一次。老师给的例子故意没有把 `static` 变量放在类开头，但是这不重要，就算放在构造方法之后还是会在类加载时先初始化静态变量。

在我的图书馆管理系统的程序中也使用到了静态方法，在 `Main` 类中，用于便捷输出。

3.2 InheritStaticInit on P21

初始化一个对象会先初始化它的类，初始化一个类会先初始化它的父类。我们先初始化 T2 类，为其分配内存空间并对静态变量默认初始化后发现其有父类 T1 类，于是将 T1 初始化，注意此时 T2 只进行了默认初始化。这时我们 T1 中调用了 T2 的静态变量，T2 已经默认初始化过了所以打印出 0。而 T1 初始化完成之后，T2 初始化继续完成。

注意，此时虽然可以正常编译运行，但是 IDE 已经给我报出了一个 Warning（图中黄色高亮部分），说明这种写法是不合理的，在一个优秀的程序中，不应该出现这种情况。



```
class T1 {
    static int s1 = 1;
    static { System.out.println("static block of T1: " + T2.s2); }
    T1() { System.out.println("T1(): " + s1); }
}

class T2 extends T1 {
    static int s2 = 2;
    static { System.out.println("static block of T2: " + T2.s2); }
    T2() { System.out.println("T2(): " + s2); }
}

public class InheritStaticInit {
    public static void main(String[] args) {
        new T2();
    }
}
```

InheritStaticInit x

```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:D:\Progr
static block of T1: 0
static block of T2: 2
T1(): 1
T2(): 2
```

3.3 BlankFinal on P31

final 成员变量如果声明的时候未被初始化，则必须在构造方法中初始化。Final 修饰的变量一旦被赋值，那么它就永远无法改变。（如果是修饰引用类型，那么是引用无法改变，即无法使它指向另一个引用，但是这个引用内部的数据应该是可以修改的。）这种方式可以用来定义一些常量，一旦确定就无法修改，可以避免被用户误修改而引起一些错误。

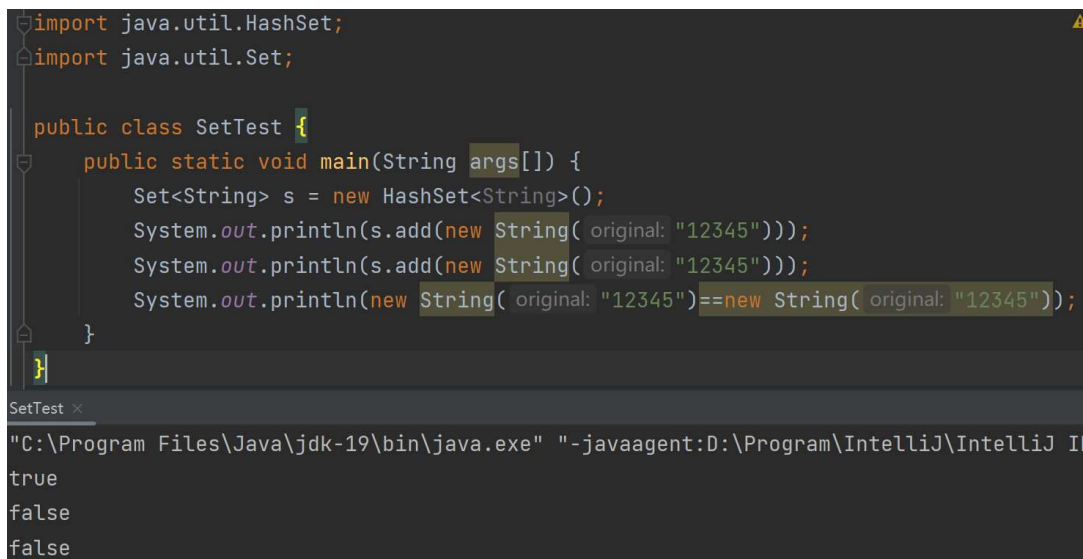
3.4 AbstractInterface on P53

(1)(3) 错误。定义了方法，相当于定义了个什么也不执行的 print 函数，接口不能定义方法体。
(2)(4) 正确。但是在 Java 中，接口本身就是一种抽象类型，不需要使用 abstract 关键字来声明。(2) 中修饰了也不会出问题，但确实没有这么做的必要。

3.5 FindDups on P64

Set 集合，同数学上的集合一样，无序（无序储存可以在一定程度上加快运行时速度）、不允许有相同元素，程序会把命令行参数中重复的字符串筛选出来并输出，最后输出 Set，即不重复的字符串。

但这里存在一个问题。如果是引用类型，set 在判断相等的时候比较的是什么？通过下面这个例子可以看出，比较的时候，其实比较的是值，也就是使用 equals 方法。（实验的时候是这么想的，但是报告写到最后我突然想，这个地方会不会比较的是哈希值或者其他的什么，毕竟我的实验只是排除了比较引用，这个可以再试一下。）



```
import java.util.HashSet;
import java.util.Set;

public class SetTest {
    public static void main(String args[]) {
        Set<String> s = new HashSet<String>();
        System.out.println(s.add(new String( original: "12345")));
        System.out.println(s.add(new String( original: "12345")));
        System.out.println(new String( original: "12345")==new String( original: "12345"));
    }
}
```

SetTest x

```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:D:\Program\IntelliJ\IntelliJ I
true
false
false
```

3.6 UseArrayList on P70

1. 添加元素默认添加到末尾。
2. ArrayList 仍是以 0 为索引开始。
3. 插入则由指定位置元素依次后移，然后插入到指定位置。
4. 这个和 python 里的列表比较像，提供了很多好用的方法。

3.7 QueueDemo on P74

虽然用了随机数，但是给定了种子，多次试验都是这一个结果，说明了 Random 是伪随机，种子相同的情况下生成的序列相同。

Queue 是一个先进先出的容器，没啥好说的。老师这里利用了 queue.peek() != null 来判断队列是否为空，也可以用 queue.size() > 0，或者直接 !queue.isEmpty() 就可以了。

3.8 Freq on P79

词频统计，Map 定义了一种映射关系。遍历给定的单词表，获取词频，然后进行更改。这里可以看出 put 时，如果键不存在，则会新建键值对，如果存在，则会更改键所对应的值。输出结果为: 8 distinct words detected: be=1, delegate=1, if=1, is=2, it=2, me=1, to=3, up=1。这可以与 python 里的字典类别，是一种很灵活好用的存储方式。

3.9 UseHashMap on P81

常规的 Map 用法，增加、修改、删除，就没什么好说的。

结果为：按字符串输出：李二=98，海飞=99，张一=86，

修改并删除之后，按字符串输出：李二=77，海飞=99。

3.10 TestIterator on P83

迭代器的用法。用迭代器迭代遍历一般主要用到两个方法 hasNext() 和 next() 用于判断边界和获得元素并迭代。注意 remove 移除的是 next 已经产生的最后一个元素，所以如果在第一次使用 next 之前就使用了 remove，程序会报错，如图：

```
Exception in thread "main" java.lang.IllegalStateException
    at java.base/java.util.ArrayList$Itr.remove(ArrayList.java:976)
    at chapter5.TestIterator.main(TestIterator.java:19)
```

3.11 CoinTest on P94

Java 枚举是一个特殊的类，一般表示一组常量。它就像是自己定义了一个集合，但是这个集合里的每一个元素并不是任何一种类型，仅仅理解为是一种标识，是枚举类型的特使实现方式。每个枚举都是通过 Class 在内部实现的，且所有的枚举值都是 public static final 的。

以下两个形式其实是基本等价的。

```
enum Color
{
    RED, GREEN, BLUE;
}
```

```
class Color
{
    public static final Color RED = new Color();
    public static final Color BLUE = new Color();
    public static final Color GREEN = new Color();
}
```

3.12 AutoBoxingTest on P99

封装类，为了基本类型可以方便地使用类的特性，JAVA 内置了封装类。封装类在使用中和基本类型基本无异，可以直接使用 +, -, =, /, * 等运算符号，但本质上并不是运算符重载，还是方法的调用。封装类可以直接赋值，而且形如 Integer(int) 的构造方法已被弃用。

自动装箱 (autoboxing): 在应该使用对象的地方使用基本类型的数据时，编译器自动将该数据包装为对应的 Wrapper 类对象。

自动拆箱 (autounboxing): 在应该使用基本类型数据的地方使用 Wrapper 类的对象时，编译器自动从 Wrapper 类对象中取出所包含的基本类型数据。

既拥有像 C 一样对基本数据类型操作的便利性，一定程度上提升运行速度 (python 是一个反例，虽然一切皆对象，但这也导致一个简单的加减乘除都需要特别复杂的一连串函数调用。) 而且还能够有面向对象所需要的封装的便捷性。个人认为这一点是 Java 可以兼顾程序员的头发和运行时的效率的关键。

4. 日历程序修改

4.1 修改复数类的继承关系

这项任务其实在上次完成复数类任务时已经这么做了，继承自 `Number` 类在和其他的数字类型做运算或比较时会方便很多。

```
public final class Complex extends Number implements Comparable<Complex>{
```

4.2 实现 `Comparable<Complex>` 接口

`Comparable<T>` 的继承方法只有一个就是 `compareTo`，众所周知数学上的复数是无法比较大小的，这里如果硬要比较大小，我选择了比较复数的模长。获取模长的方法在上次作业已经完成了，所以只需要获取模长，然后作为 `Double` 类型来比较，而 `Double` 类型的比较在 `Double` 类里面已经写好了，可以直接以静态方法调用，可是说是非常方便了。

```
@Override
public int compareTo(Complex o) {
    return Double.compare(this.getModulus(), o.getModulus());
}
```

4.3 设计并实现静态方法 `valueOf`

这个方法写起来比想象中复杂的多。大概就是，要把用户当成傻子，他们可能给出各种各样的输入，这个函数在测试的时候就不断发现有某种形式的字符串会被错误的转换。废话就少说了，直接上代码吧。（即使测试了很多次，我还是不敢保证它这个函数没有 Bug）

```
public static Complex valueOf(String str) {
    // Verify input
    if (str == null) { throw new NullPointerException("Input string is null"); }
    str = str.trim(); // Remove leading and trailing white spaces
    if (str.isEmpty()) { throw new IllegalArgumentException("Input string is empty"); }
    if (!str.matches(regex: "-?\\d+(\\.\\d+)?([+\\-]\\d+(\\.\\d+)?i)?")) {
        throw new IllegalArgumentException("Invalid input string: " + str);
    }
    // Remove all white spaces and split string by "+" or "-"
    String[] parts = str.replaceAll(regex: "\\s", replacement: "").split(regex: "(?=[+\\-])");
    double real = 0;
    double imag = 0;
    for (String part : parts) {
        if (part.endsWith("i")) {
            // imaginary part
            part = part.substring(0, part.length() - 1); // remove "i" suffix
            if (part.equals("+") || part.equals("-")) { part += "1"; }
            imag += Double.parseDouble(part);
        } else { // real part
            real += Double.parseDouble(part);
        }
    }
    return new Complex(real, imag);
}
```


4.5 设计并实现静态的加运算方法

在上一次作业中已经实现了。这里再贴出来一次。

```
// add
private static Complex complexAddComplex(Complex _complex1, Complex _complex2){
    return new Complex( real: _complex1.real+ _complex2.real, imag: _complex1.imag+ _complex2.imag);
}
public static Complex add(Number x1, Number x2){
    if(x1 instanceof Complex && x2 instanceof Complex){
        return Complex.complexAddComplex(((Complex)x1),((Complex)x2));
    } else if(x1 instanceof Complex) {
        return Complex.complexAddComplex((Complex)x1,new Complex(x2.doubleValue(), imag: 0D));
    } else if(x2 instanceof Complex) {
        return Complex.complexAddComplex(new Complex(x1.doubleValue(), imag: 0D), (Complex)x2);
    } else return new Complex( real: x1.doubleValue()+x2.doubleValue() , imag: 0D);
}
```

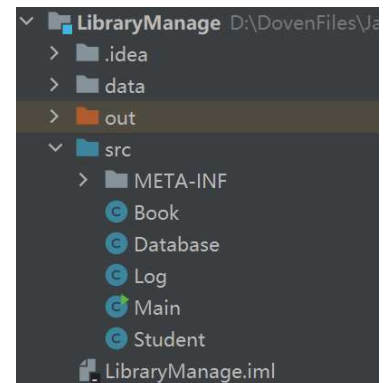
5. 编写图书馆管理系统

5.1 整体架构

共有五个类组成。分别是 Book、Database、Log、Student、Main。Book 和 Student 用来存储书籍和学生信息，和相应的操作。Log 则用来记录一切操作过程。

Database 定义了借书还书添加书籍等等一系列的操作，包括读取数据和存储数据。他建立在 Book 和 Student 和 Log 之上，去对这三者协调统一。

Main 主要是面向用户的输入输出过程。



5.2 Book 类

这里面一个 Book 类储存一种书，我指的是，《Java 程序设计》这本书有 10 本，那么会存储到一起，他们之间不会有区分。（但其实这是不好的，这样不利于每一本书行程的追踪，但是也不是大问题）。

这里主要实现了书籍信息存储，数量的记录，在 Book 类的对象中，会记录下这本书的所有借阅记录。以及实现了 equals 方法用来判断是否是同一本书。以及 Similarity 方法，用来进行模糊查找，其中字符串相似度使用了 Levenshtein distance 这是在 NLP（Natural language process）里面一种常用的评价字符串相似度的算法。

值得注意的是，这里面给书籍添加的 Id 属性，用于准确的查找每本书，这个 Id 是由 Database 负责生成且保证唯一的。可以认为是给每本书一个不同的随机值（其实不随机）。这个东西在实际应用中可能是书籍上贴着一个二维码。让学生借书还书的过程更简单且不容易出错。

（代码不再贴出，详情看代码）。

5.3 Student 类

和 book 类相似，就是储存学生信息，和以学生的角度来储存 Log 的。也就是说。一次借阅记录。会在对应 Student 实例和对应的 Book 实例储量两份相同的记录，这样我们既可以查询哪个学生借了哪本书，也可以查询哪本书借给了哪个学生。

(代码不再贴出，详情看代码)。

5.4 Log 类

存储记录，一份记录中包含，时间，人员，书籍，借阅还是归还，三个信息。就是单纯的用来方便的储存和调用信息，没什么特别的。

5.5 Database 类

这个类相对复杂得多。它集成了所有的系统内部操作。

各种函数如下，而且为了信息存储。这里直接采用序列化的方法把信息存储到了一个文件里，下次运行程序再从文件中读取信息。

而且在此定义了各种类型的查找，基于编号或名字的精确查找，还有基于书名作者出版社的模糊查找，都在这里定义（很多调用了 Book 类或者 Student 类的函数）。

(代码里的注释没整理过，因为部分复制粘贴，部分注释有点小问题)

```
public class Database implements Serializable{

    @Serial private static final long serialVersionUID = 1024L;
    private HashMap<Integer,Book> books = new HashMap<>();
    private HashMap<Long,Student> students = new HashMap<>();
    private ArrayList<Log> log = new ArrayList<>();
    private int countbook=0;
    private int countstudent=0;

    public Database() { return; } // create a new database
    public Database(String path) throws IOException, ClassNotFoundException {...} // Construct function
    public void save(String path) throws IOException {...} // save

    public int addbook(int number, String name, String publisher, String[] writers){...} // Construct function
    public void delbook(Book book) { this.books.remove(book.getId()); } // Construct function
    public long addstudent(long id, String name){...} // Construct function
    public void delstudent(Student student){...} // Construct function

    public ArrayList<Book> search_book(String str) {...} // search_book by name
    public Book search_book(int id) {...} // search_book by id
    public Student search_student(String str) {...} // search_student by name
    public Student search_student(long id) {...} // search_student by id
    public ArrayList<Object> search_all(String str) {...} // search anything

    public HashMap<Integer,Book> getBooks(){ return books; }
    public HashMap<Long,Student> getStudents(){ return students; }
    public ArrayList<Log> getLog(){ return this.log; }
    public int getCountbook() { return countbook; }
    public int getCountstudent() { return countstudent; }

    @Override
    public String toString() {...} // get information
```


5.6 Main 函数和运行结果

这里主要是输入和输出的部分。代码没什么好说的。来看一下最终的功能。

```
图书馆管理系统 [版本 0.1] (c) MTDoven.
-----图书馆管理系统启动-----
数据加载自:.\database.data
以下是所有图书和人员信息:
ID:22009200605 Name:熊二
ID:22009200606 Name:老六
ID:22009200601 Name:汤栋文
ID:22009200603 Name:李四
ID:22009200699 Name:???
ID:3 Number:5 Name:Java程序设计 Publisher:西安电子科技大学出版社 Writers:[王煦]
ID:4 Number:5 Name:C语言程序设计 Publisher:西安电子科技大学出版社 Writers:[牛毅, 王煦]
ID:6 Number:4 Name:正经名字 Publisher:正经出版社 Writers:[正经作者]
ID:7 Number:9 Name:深入浅出Pytorch Publisher:Deeplearning.com Writers:[Andraw Ng]
ID:8 Number:1 Name:高等数学 Publisher:西安电子科技大学 Writers:[杨有龙]
ID:9 Number:0 Name:路人女主的养成方法 Publisher:BiliBili Writers:[啦啦啦, 哈哈]
ID:10 Number:100 Name:大学物理 Publisher:西安电子科技大学出版社 Writers:[不知道]
数据加载完成, 按任意键继续...
```

这里有一个小 Bug 应该是编码问题, 在 cmd 里运行时有些汉字就会识别不出来, 但在 IDE 里面就没有问题, 这个有待解决。

```
-----图书馆管理系统-----
0. 保存数据并退出程序
1. 管理员 添加/删除 书籍/人员
2. 学生 借书/还书
3. 查询学生借阅情况
4. 查询书籍借阅情况
5. 任意信息搜索
6. 获取更多帮助
请输入数字:|
```

“5 任意信息搜索”的查询可以查询任何信息非常方便

```
-----查询界面-----
请输入书籍 名称/作者/出版社/姓名/学号: 王煦
ID:3 Number:5 Name:Java程序设计 Publisher:西安电子科技大学出版社 Writers:[王煦]
ID:4 Number:5 Name:C语言程序设计 Publisher:西安电子科技大学出版社 Writers:[牛毅, 王煦]
ID:6 Number:4 Name:正经名字 Publisher:正经出版社 Writers:[正经作者]
ID:7 Number:9 Name:深入浅出Pytorch Publisher:Deeplearning.com Writers:[Andraw Ng]
ID:8 Number:1 Name:高等数学 Publisher:西安电子科技大学 Writers:[杨有龙]
操作完成, 按任意键返回主界面...
```

退出时一定要记得输入 0 退出, 这样才能保证数据被保存而不丢失。(后面可以改进一下, 没当数据发生变化就自动保存)

```
请输入数字:0
所有数据已保存到:.\database.data
-----图书馆管理系统退出-----
```

其他部分留给老师探索了, 这里就不再贴出了。

6. 总结

要学的东西还很多, 要走的路还很长;
革命尚未成功, 同志仍须努力。