并不十分全，就这样吧~

第 1 章 Java语言概述

- Java语言有哪些特点?
  答：简单易学、面向对象、平台无关性、可靠性、安全性、支持多线程、支持网络编程、编译与解释并存
- 什么是Java的虚拟机?
  答：任何一种可以运行Java字节码的软件均可看成是Java的虚拟机
- 什么是字节码？采用字节码的最大好处是什么?
  答：字节码是Java虚拟机的指令组，和CPU上的微指令很相似 。字节码最大的好处是可跨平台运行
- 什么是平台无关性？Java语言是怎样实现平台无关性的?
  答：编写的应用程序不用修改就可以在不同的软硬件平台上运行。Java语言是靠JVM在目标代码级实现平台无关性的，可以说JVM是Java平台无关的基础
- Java语言程序有几种？他们包含哪几个方面?
  答：Application应用程序和Applet小程序
- 什么是Java程序的主类?
  答：Java应用程序的主类必须包含一个定义为`public static void main(String[] args)`；Java小程序的主类必须是一个继承自系统JApplet或Applet的子类，且该类必须是public类。

第 3 章 Java语言基础

- Java语言定义了哪几种基本数据类型?
  答：8种基本数据类型。byte, short, int, long, float, double, char
- 表示整数类型数据的关键字有哪几个？他们各占用几个字节?
  答：byte, short, int, long分别占1, 2, 4, 8个字节
- 单精度浮点float和双精度浮点double的区别是什么?
  答：单精度浮点数的数据位是32位，双精度浮点数的数据位是64位，double的精度是float的两倍
- 字符型常量与字符串常量的主要区别是什么?
  答：字符串常量是用一对单引号括起来的单个字符，字符串常量是用双引号括起来的一串若干个字符（可以是0个）
- 简述Java语言对定义标识符的规定有哪些。
  答：标识符可以由字母、数字和下划线、美元符号等组合而成，标识符必须以字母、下划线或美元符号开头，不能以数字开头
- Java语言采用何种编码方案？有何特点?
  答：Unicode字符集编码方案，便于西文字符和中文字符的处理
- 什么是强制类型转换？在什么情况下需要强制类型转换?
  答：强制类型转换就是将长数据转换为短数据。如果要将较长的数据转换成较短的数据时，就要进行强制类型转换。
- 自动类型转换得前提是什么？转换是从"短"到"长"的优先级顺序是怎样的?
  答：转换前的数据类型与转换后的类型兼容，转换后的数据类型的表示范围比转换前的类型大。byte→short→char→int→long→float→double
- 数字字符串转换为数值型数据时，所使用的方法有哪些?
  答:

转换的方法	功能说明
Byte.parseByte(String s)	将数字字符串转换为字节型数据
Short.parseShort(String s)	将数字字符串转换为短整型数据
Integer.parseInteger(String s)	将数字字符串转换为整型数据
Long.parseLong(String s)	将数字字符串转换为长整型数据
Float.parseFloat(String s)	将数字字符串转换为浮点型数据
Double.parseDouble(String s)	将数字字符串转让为双精度型数据
Boolean.parseBoolean(String s)	将字符串转换为布尔型数据

- 写出由键盘输入数据的两种基本格式。
  答：在1.5版本之前，Java用BufferedReader来读取输入数据，在1.5版本之后，Java用Scanner来读取输入数据

```
import java.io.*;
public class Buffer {
    public static void main(String[] args) throws IOException {
        String s;
        BufferedReader buf = new BufferedReader(new
InputStreamReader(System.in));
        str = buf.readLine();
    }
}
```

```
import java.util.*;
public class Scan {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        double num = sc.nextDouble();
    }
}
```

- 编写程序，从键盘上输入一个浮点数，然后将该浮点数的整数部分输出。
  答：浮点数的输入用double或者float，第一种方法用BufferedReader来读，第二种方法用Scanner来读

```
import java.io.*;
public class Exercise {
    public static void main(String[] args) throws NumberFormatException,
IOException {
        BufferedReader buff = new BufferedReader(new
InputStreamReader(System.in));
        double num = Double.parseDouble(buff.readLine());
        int i = (int) num;
        System.out.println(i);
    }
}
```

```
import java.util.*;
public class Exercise {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        double t = sc.nextDouble();
        int num = (int) t;
        System.out.println(num);
    }
}
```

- 编写程序，从键盘上输入两个整数，然后计算他们相除后得到的结果并输出。
  答：这里用Scanner来读两个整数

```
import java.util.*;
public class Exercise {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int a = sc.nextInt();
        int b = sc.nextInt();
        System.out.println(a / b);
    }
}
```

- 编写程序，从键盘上输入圆柱体的底半径r和高h，然后计算其体积并输出。
  答：这里要用到Math.PI，假设题目给出的半径r和高h都是整形数值

```
import java.util.*;
public class Exercise {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int r = sc.nextInt();
        int h = sc.nextInt();
        System.out.println(Math.PI * r * r * h);
    }
}
```

- Java语言有哪些算术运算符、关系运算符、逻辑运算符、位运算符和赋值运算符?
  - 算术运行符包括+、-、*、/、%、++、--
  - 关系运算符包括>、<、>=、<=、==、!=
    - 逻辑运算符包括!、&&、||、&、|
    - 位运算符包括>>、<<、>>>、&、|、^、~
    - 赋值运算符包括=
- 逻辑运算符中“逻辑与、逻辑或”和“简洁与、简洁或”的区别是什么?
  答：非简洁运算在必须计算玩左右两个表达式之后，才取结果值；而简洁运算可能只计算左右的表达式而不计算右边的表达式
- 逻辑运算符与位运算符的区别是什么?
  答：位运算符的操作数只能为整型或字符型数据，逻辑运算符的操作数为boolean型的量
- 什么是运算符的优先级和结合性?
  答：运算符的优先级决定了表达式中不同运算执行的先后顺序，运算符的结合性决定了并列的多个同级运算符的先后执行顺序
- 写出下列表达式的值，设x=3, y = 17, yn = true。

(1) x + y * x --
3 + 17 * 3, 3 + 51, 54

(2) -x*y+y
-3 * 17 + 7, -51 + 7, -44

(3)x<y&&yn
3 < 17 && true, true

(4)x > y ||!yn
3 > 17 || !true, false

(5)y!=++x?x:y
17 != 4 ? 4 : 17, 4

(6)y++/--x
17 / 2, 8

第 4 章 流程控制

- 将学生的学习成绩按不同的分数段分为优、良、中、及格和不合格五个等级，从键盘上输入一个0~100之间的成绩，输出响应的等级。要求用switch语句实现。

```
import java.util.*;
public class Exercise {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int score = in.nextInt();
        switch (score / 10) {
        case 10:
        case 9:
            System.out.println("优");
            break;
        case 8:
            System.out.println("良");
            break;
        case 7:
            System.out.println("中");
            break;
        case 6:
            System.out.println("及格");
            break;
        default:
            System.out.println("不合格");
            break;
        }
    }
}
```

- 设学生的学习成绩按如下的分数评定为四个等级：85~100为A，70~84为B，60~69为C，0~59为D。从键盘上输入一个0~100之间的成绩，要求用switch语句根据成绩，评定并输出相应的等级。

```
import java.util.*;
public class Exercise {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int score = in.nextInt();
        switch (score / 10) {
        case 10:
        case 9:
            System.out.println("A");
            break;
        case 8:
            if (score % 10 >= 5)
                System.out.println("A");
            else
                System.out.println("B");
            break;
        case 7:
        case 6:
            System.out.println("C");
            break;
        default:
            System.out.println("D");
            break;
        }
    }
}
```

- 编写一个Java应用程序，输入1~100之间所有既可以被3整除，又可被7整除的数。

```
public class Exercise {
    public static void main(String[] args) {
        for (int i = 1; i <= 100; i++) {
            if (i % 3 == 0 && i % 7 == 0)
                System.out.println(i);
        }
    }
}
```

- 编写一个Java应用程序，在键盘上输入数n，计算并输出1!+2!+...+n!的结果

```java
import java.util.*;
public class Exercise {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        long sum = 0, temp = 1;
        for (int i = 1; i <= n; i++) {
            sum += temp * i;
            temp = i;
        }
        System.out.println(sum);
    }
}
```

- 在键盘上输入数n，编程计算sum = 1 - (1 / 2!) + (1/ 3!) -...(-1)^n-1(1/n!)。

```java
import java.util.*;
public class Exercise {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        double sum = 0;
        int sign = 1, temp = 1;
        for (int i = 1; i <= n; i++) {
            sum += sign * (1.0 / (temp * i));
            temp *= i;
            sign = -sign;
        }
        System.out.println(sum);
    }
}
```

- 水仙花数是指其个位、十位和百位三个数字的立方和等于这个三位数本身，求出所有的水仙花数。

```
public class Exercise {
    public static void main(String[] args) {
        for (int i = 100; i <= 999; i++) {
            int temp = i, sum = 0;
            while (temp != 0) {
                sum += Math.pow(temp % 10, 3);
                temp /= 10;
            }
            if (sum == i)
                System.out.println(i);
        }
    }
}
```

- 从键盘输入一个整数，判断该数是否是完全数。完全数是指其所有因数（包括1但不包括其本身）的和等于该数自身的数。例如，28=1+2+4+7+14就是一个完全数。

```
import java.util.*;
public class Exercise {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int num = in.nextInt();
        int sum = 0;
        for (int i = 1; i < num; i++) {
            if (num % i == 0)
                sum += i;
        }
        System.out.println(sum == num);
    }
}
```

- 计算并输出一个整数各位数字之和。如，5423的各位数字之和为5+4+2+3。

```java
import java.util.*;
public class Exercise {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int num = in.nextInt();
        int sum = 0;
        while (num != 0) {
            sum += num % 10;
            num /= 10;
        }
        System.out.println(sum);
    }
}
```

- 从键盘上输入一个浮点型数，然后将该浮点数的整数部分和小数部分分别输出。

```java
import java.util.*;
public class Exercise {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        double num = in.nextDouble();
        int digital = (int) num;
        System.out.println(digital + " " + (num - digital));
    }
}
```

- 设有一长为3000m的绳子，每天减去一半，问需几天时间，绳子的长度会短于5m。

```java
public class Exercise {
    public static void main(String[] args) {
        int day = 0;
        double len = 3000;
        while (len >= 5) {
            len /= 2;
            day++;
        }
        System.out.println(day);
    }
}
```

- 编程输出如下的数字图案：

```
1  3  6  10  15
2  5  9  14
4  8 13
7 12
11
```

```
public class Exercise {
      public static void main(String[] args) {
         int[][] matrix = new int[5][5];
         int k = 1;
         for (int i = 0; i < 5; i++) {
            for (int j = 0; j <= i; j++) {
               matrix[i - j][j] = k++;
            }
         }
         for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 5 - i; j++) {
               System.out.print(matrix[i - j][j] + " ");
            }
            System.out.println();
         }
      }
   }
```

第 5 章 数组与字符串

- 从键盘输入n个数，输出这些数中大于其平均值的数。

```
import java.util.*;
public class Exercise {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        int[] num = new int[n];
        double aver = 0;
        for (int i = 0; i < n; i++) {
            num[i] = in.nextInt();
            aver += num[i] * 1.0 / n;
        }
        for (int i = 0; i < n; i++) {
            if (num[i] >= aver)
                System.out.println(num[i]);
        }
    }
}
```

- 从键盘键入n个数，求这n个数中得最大数与最小数并输出。

```
import java.util.*;
public class Exercise {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        int max = Integer.MIN_VALUE, min = Integer.MAX_VALUE;
        for (int i = 0; i < n; i++) {
            int temp = in.nextInt();
            if (temp > max)
                max = temp;
            if (temp < min)
                min = temp;
        }
        System.out.println(max + " " + min);
    }
}
```

- 求一个3阶方阵的对角线上各元素之和。

```
int sum = 0;
for (int i = 0; i < 3; i++) {
    sum += matrix[i][i];
    sum += matrix[3 - i][i];
}
```

- 找出4 x 5矩阵中值最小和最大元素，并分别输出其值及所在的行号和列号。

```
int minRow = 0, minCol = 0, maxRow = 0, maxCol = 0;
for (int i = 0; i < 4; i++) {
    for (int j = 0; j < 5; j++) {
        if (matrix[minRow][minCol] > matrix[i][j]) {
            minRow = i;
            minCol = j;
        }
        if (matrix[maxRow][maxCol] < matrix[i][j]) {
            maxRow = i;
            maxCol = j;
        }
    }
}
System.out.println(minRow + " " + minCol + " " + maxRow + " " + maxCol);
```

- 产生0~ 100之间的8个随机整数，并用冒泡排序将其升序排序后输出（冒泡排序算法：每次进行相邻两数的比较，若次序不对，则交换两数的次序）。

```
import java.util.*;
public class Exercise {
    public static void main(String[] args) {
        int[] num = new int[8];
        Random random = new Random();
        for (int i = 0; i < 8; i++) {
            num[i] = random.nextInt() % 100;
        }
        for (int i = 0; i < 8; i++) {
            for (int j = 0; j < 8 - i - 1; j++) {
                if (num[j] > num[j + 1]) {
                    int temp = num[j];
                    num[j] = num[j + 1];
                    num[j + 1] = temp;
                }
            }
        }
        for (int i = 0; i < 8; i++) {
            System.out.println(num[i] + " ");
        }
    }
}
```

- 15个红球和15个绿球排成一圈，从第1个球开始数，当数到第13个球时就拿出此球，然后再从下一个球开始数，当再数到第13个球时又取出此球，如此循环进行，直到仅剩15个球为止，问怎样排法才能使每次取出的球都是红球。

```
public class Exercise {
    public static void main(String[] args) {
        int[] ball = new int[30];
        for (int j = 0, temp = 0; temp <= 15; temp++) {
            int cnt = 1, i = j;
            while (cnt < 13 || ball[i] == 1) {
                if (ball[i] != 1)
                    cnt++;
                i++;
                if (i == 30)
                    i = 0;
            }

            ball[i] = 1;
            j = i + 1;
            if (j == 30)
                j = 0;
        }
        for (int i = 0; i <= 29; i++) {
            System.out.print(ball[i] + " ");
        }
    }
}
```

- 编写Java应用程序，比较命令行中给出的两个字符串是否相等，并输出比较结果。

```
public class Exercise {
    public static void main(String[] args) {
        System.out.println(args[0].equals(args[1]));
    }
}
```

- 从键盘上输入一个字符串和子串开始位置与长度，截取该字符串的子串并输出。

```
import java.util.*;
public class Exercise {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        String s = in.next();
        int start = in.nextInt(), len = in.nextInt();
        System.out.println(s.substring(start, start + len));
    }
}
```

- 从键盘上输入一个字符串和一个字符，从该字符串中删除给定的字符。

```
import java.util.*;
public class Exercise {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        String s = in.nextLine();
        char c = in.next().charAt(0);
        for (int i = 0; i < s.length(); i++) {
            if (s.charAt(i) != c)
                System.out.print(s.charAt(i));
        }
    }
}
```

- 编程统计用户从键盘输入的字符串中所包含的字母、数字和其他字符的个数。

```
import java.util.*;
public class Exercise {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        String s = in.nextLine();
        int ch = 0, digital = 0, other = 0;
        for (int i = 0; i < s.length(); i++) {
            char c = s.charAt(i);
            if ((c >= 'A' && c <= 'Z') || (c >= 'A' && c <= 'z'))
                ch++;
            else if (c >= '0' && c <= '9')
                digital++;
            else
                other++;
        }
        System.out.println(ch + " " + digital + " " + other);
    }
}
```

- 将用户从键盘输入的每行数据都显示输出，直到输入字符串"exit"，程序运行结束。

```java
import java.util.*;
public class Exercise {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        String data = "";
        do {
            data = in.nextLine();
            if ("exit".equals(data))
                break;
            System.out.println(data);
        } while (true);
    }
}
```

第 6 章 类与对象

- 类与对象的区别是什么?
  类是对某一类食物的描述，是抽象的、概念上的定义
  对象则是实际存在的属该类食物的具体的个体
  对象是类的实例化
- 如何定义一个类？类的结构是怎样的?
  定义一个类实际上就是定义类的属性和方法

```
类的一般结构如下：
[类修饰符] class 类名称
{
  [修饰符] 数据类型 成员变量名称;
  [修饰符] 返回值的数据类型 方法名(参数1, 参数2, ……)
  {
    语句序列;
    return [表达式];
  }
}
```

- 定义一个类时所使用的修饰符有哪几个？每个修饰符的作用是什么？是否可以混用?
  定义一个类时所使用的修饰符有4个，每个修饰符的作用如下：

修饰符	含义
public	将一个类声明为公共类，它可以被任何对象方法
abstract	将一个类声明为抽象类，没有实现方法，需要子类提供方法的实现，所以不能创建该类的实例
final	将一个类声明为最终类即非继承类，表示他不能被其他类所继承
缺省	缺省修饰符时，则表示只有在相同包中的对象才能使用这样的类

- 成员变量的修饰符有哪些？各修饰符的功能是什么？是否可以混用？
  成员变量的修饰符有8个，每个修饰符的功能如下：

修饰符	含义
public	公共访问控制符。指定该变量为公共的，它可以被任何对象的方法访问
private	私有访问控制符。指定该变量只允许自己类的方法访问，其他任何类(包括子类)中的方法均不能访问此变量
protected	保护访问控制符。指定该变量只可以被它自己的类及其子类或同一包中的其他类访问，在子类中可以覆盖此变量
缺省	缺省访问控制符时，则表示在同一个包中的其他类可以访问此成员变量，而其他包中的类不能访问该成员变量
final	最终修饰符。指定此变量的值不能改变
static	静态修饰符。指定该变量被所有对象共享，即所有的实例都可以使用该变量
transient	过度修饰符。指定该变量是一个系统保留，暂无特别作用的临时性变量
volatile	易失修饰符。指定该变量可以同时被几个线程控制和修改

- 成员方法的修饰符有哪些？各修饰符的功能是什么？是否可以混用？
  成员方法的修饰符有9个，每个修饰符的功能如下：

修 饰 符	含 义
public	公共访问控制符。指定该方法为公共的，它可以被任何对象的方法访问
private	私有访问控制符。指定该方法只允许自己类的方法访问，其他任何类(包括子类)中的方法均不能访问此方法
protected	保护访问控制符。指定该方法只可以被它自己的类及其子类或同一包中的其他类访问
缺省	缺省访问控制符时，则表示在同一个包中的其他类可以访问此成员方法，而其他包中的类不能访问该成员方法
final	最终修饰符。指定该方法不能被重载
static	静态修饰符。指定不需要实例化一个对象就可以调用的方法
abstract	抽象修饰符。指定该方法只声明方法头，而没有方法体，抽象方法需在子类中被实现
synchronized	同步修饰符。在多线程程序中，该修饰符用于在运行前，对它所属的方法加锁，以防止其他线程访问，运行结束后解锁
native	本地修饰符。指定此方法的方法体是用其他语言(如C)在程序外部编写的

- 成员变量与局部变量的区别有哪些?
    - 从语法形式上看，成员变量属于类的，而局部变量是在方法中定义的变量或是方法的参数；成员变量可以被public、private、static等修饰符所修饰，而局部变量则不能被访问控制修饰符及static所修饰；成员变量和局部变量都可以被final所修饰
    - 从变量在内存中得存储方式上看，成员变量是对象的一部分，而对象是存在于堆内存的，而局部变量是存在于栈内存的。
    - 从变量在内存中的生存时间上看，成员变量是对象的一部分，它随着对象的创建而存在，而局部变量随着方法的调用而产生，随着方法调用的结束而自动消失
    - 成员变量如果没有被赋初值，则会自动以类型的默认值赋值（有一种情况例外，被final修饰但没有被static修饰的成员变量必须显示地赋值）；而局部变量则不会自动赋值，必须显式地赋值后才能使用。
- 创建一个对象使用什么运算符？对象实体与对象的引用有何不同?
  创建一个对象使用new运算符；对象实体是实在存在于对内存中的，而对象的引用是管理对象实体的句柄存在于栈内存中
- 对象的成员如何表示?
  对象的成员通过对象名.对象成员来访问
- 在成员变量或成员方法前加上关键字this表示什么含义?
  this特指成员变量
- 什么是方法的返回值？返回值在类的方法里面的作用是什么?

- 在方法调用中，使用对象作为参数进行传递时，是“传值”还是“传址”？对象作参数起到什么作用?
  传址；

- 什么叫匿名对象？一般在什么情况下使用匿名对象？
  当一个对象被创建之后，不定义对象的引用变量，而是直接调用对象的方法，这个的对象叫做匿名对象；
  使用匿名对象的情况有如下两种：
  - 如果对一个对象只需要进行一次方法调用，那么就可以使用匿名对象
  - 将匿名对象作为实参传递给一个方法调用
- 定义一个Student类，包含如下内容：
  成员变量：学号，姓名，性别，班干部否，数学，语文，外语
  成员方法：输入，总分，平均分
  编程实现这个类，并调用响应的方法输入数据，计算总分和平均分。

```
import java.util.*;
public class Student {
  private String id;
  private String name;
  private String gender;
  private String isLeader;
  private int math;
  private int chinese;
  private int english;

  public void input() {
    Scanner in = new Scanner(System.in);
    id = in.next();
    name = in.next();
    gender = in.next();
    isLeader = in.next();
    math = in.nextInt();
    chinese = in.nextInt();
    english = in.nextInt();
    in.close();
  }

  public int total() {
    return math + chinese + english;
  }

  public double aver() {
    return total() / 3.0;
  }
}
```

- 以m行n列二维数组为参数进行方法调用，分别计算二维数据各列元素之和，返回并输出所计算的结果。

```
int[] sum = new int[n];
for (int i = 0; i < n; i++) {
  for (int j = 0; j < m; j++) {
    sum[i] += matrix[j][i];
  }
  System.out.println(sum[i]);
}
```

第 7 章 Java语言类的特性

- 一个类的公共成员与私有成员有何区别?
  私有成员即只能在类的内部访问，在类的外部访问会出现错误
  公共成员则可以被其他类所访问
- 什么方法的重载?
  有一些方法含义相同，但带有不同的参数，这些方法使用相同的名字，这就叫方法的重载
- 一个类的构造方法的作用是什么？若一个类没有声明构造方法，该成员能正确执行吗？为什么?
  构造方法的作用是在对象被创建时初始化对象的成员的方法；可以正确执行，因为如果省略构造方法，Java编译器会自动为该类生成一个默认的构造方法
- 构造方法有哪些特性?
  - 构造方法的方法名与类名相同
  - 构造方法没有返回值，但不能写void
  - 构造方法的主要作用是完成对类对象的初始化工作
  - 构造方法一般不能有编程人员显式地调用，而是用new来调用
  - 在创建一个类的对象的同时，系统会自动调用该类的构造方法为新对象初始化
- 在一个构造方法内可以调用另一个构造方法吗？如果可以，如何调用?
  可以；通过this来调用，而且this关键字必须卸载构造方法内的第一行位置
- 静态变量与实例变量有哪些不同?
  静态变量是隶属于类的变量，而不是属于任何一个类的具体对象，静态变量不需要实例化就可以使用。
  实例变量是隶属于对象的变量，是属于具体的一个类的，是需要把类实例化为对象才可以使用的
- 静态方法与实例方法有哪些不同?
  静态方法实质是属于整个类的方法，而实例方法是属于某个具体对象的方法
- 在一个静态方法内调用一个非静态成员为什么是非法的?
  静态方法是属于整个类的，所以它不能 操纵和处理属于某个对象的成员，而只能处理属于整个类的成员，即静态方法只能访问静态成员变量或静态成员方法
- 对象的相等于指向它们的引用相等，两者有什么不同?

- 什么是静态初始化器？其作用是什么？静态初始化器由谁在何时执行？它与构造方法有何不同?
  静态初始化器是由关键字static修饰的一对大括号“{}”括起来的语句组；它是用来初始化工作的；
  静态初始化器有Java虚拟机在类初始化的时候一次执行；它与构造方法的不同：

- 构造方法是对每个新创建的对象初始化，而静态初始化器是对类自身进行初始化。
- 构造方法是在用new运算符创建新对象时又系统自动执行，而静态初始化器一般不能有程序来调用，它是在所属的类被加载入内存时有系统调用执行的。
- 用new运算符创建多少个新对象，构造方法就被调用多少次，但静态初始化器则在类被加载入内存时只执行一次，与创建多少个对象无关。
- 不同于构造方法，静态初始化器不是方法，因为没有方法名、返回值和参数。

第 8 章 继承、抽象类和接口

- 子类将继承父类的所有成员吗？为什么？
  不是；父类的私有成员只能是父类自己使用，子类不能继承，子类只能继承父类的所有非private成员
- 在子类中可以调用父类的构造方法吗？若可以，如何调用？
  可以；在子类的方法中通过super()来调用父类特定的构造方法
- 在调用子类的构造方法之前，会先自动调用父类中没有参数的构造方法，其目的是什么？
  目的是为了便于初始化操作
- 在子类中可以访问父类的成员吗？若可以，用什么方式访问？
  可以；通过super.变量名或super.方法名来方法父类的成员
- 用父类对象变量可以访问子类的成员吗？若可以，则只限于什么情况？
  可以；只限于“覆盖”的情况发生时，也就是说，父类与子类的方法名称、参数个数与类型必须完全相同，才可通过父类的对象调用子类的方法。
- 什么是多态机制？Java语言中是如何实现多态的？

- 方法的覆盖与方法的重载有何不同？
  重载是指在同一个类内定义多个名称相同，但参数个数或类型不同的方法
  覆盖是指在子类中，定义名称、参数个数与类型均与父类中完全相同的方法，用于实现重写父类中同名方法的功能。
- this和super分别有什么特殊的含义？
  super是从子类调用父类的成员，而this是调用同一类中得其它成员
- 什么是最终类与最终方法？他们的作用是什么？
  如果用final修饰成员方法，则该成员方法不能再被子类所覆盖，即该方法为最终方法
  如果用final修饰一个类，则该类不能被继承，即该类是最终类
- 什么是抽象类和抽象方法？使用时应注意哪些问题？
  抽象类是以修饰符abstract修饰的类，抽象方法是以abstract关键字开头的方法，此方法只声明返回值的数据类型、方法名称与所需的参数，但没有方法体；注意：
  - 抽象类是需要被继承的，所以abstract类不能用final来修饰。
  - abstract不能与private、static、final或native并列修饰同一个方法。
- 什么是接口？为什么要定义接口？
  接口的数据成员都是静态的且必须初始化，接口中的方法必须全部都声明为abstract的，接口就是一种特俗的抽象类；接口的主要作用是帮助实现类的多继承
- 如何定义接口？接口与抽象类有哪些异同？
  定义接口的语法格式如下：

```
[public] interface 接口名称 [extends 父接口名列表]
{
  [public][static][final] 数据类型 成员变量名 = 常量;
  [public][static] 返回值的数据类型 方法名(参数列表);
}
```

接口与抽象类不同：

- 接口的数据成员都是静态的且必须初始化
- 接口中的方法必须全部都声明为abstract的，也就是说，接口不能像抽象类一样有一般方法，而必须全部是抽象方法

- 内部类的类型有哪几种？分别在什么情况下使用？它们所起的作用有哪些？

- 怎样使用匿名内部类对象？

- 什么是包？它的作用是什么？如何创建包？如何引用包中的类？
  包是类的组织方式；包得作用是提供区别类名空间的机制；使用package可以创建一个包，创建语法`package 包名1[.包名2[.包名3]...]`；使用import导入包
- Java语言中怎样清除对象？能否控制Java系统中垃圾的回收时间？
  Java语言垃圾回收清楚对象；不能控制Java系统中垃圾的回收时间

第 9 章 异常处理

- 什么是异常？简述Java语言的异常处理机制。
  异常是指在程序运行中又代码产生的一种错误；异常，抛出异常，捕获异常
- Throwable类的两个直接子类Error和Exception的功能各是什么？用户可以捕获的异常是哪个类的异常？
  Error类及其子类的对象，代表了程序运行时Java系统内部的错误
  Exception子类则是提应用程序使用的，它是用户程序能够捕捉到得异常情况。
  用户可以捕获的异常是Exception异常
- Exception类有何作用？每个Exception类的对象代表了什么？
  Exception类对象是Java程序抛出和处理的对象，它由各种不同的子类分别对应于各种不同类型的异常。
- 什么是运行时异常？什么是非运行时异常？
  运行时异常是程序运行时自动地对某些错误做出反应而产生的；非运行时异常是在程序运行过程中又环境原因造成的异常
- 抛出异常有哪两种方式？
  - 系统自动抛出的异常
  - 指定方法抛出异常
- 在捕获异常时，为什么要在catch()括号内设一个变量e？

- 在异常处理机制中，用catch()括号内的变量e接收异常类对象的步骤有哪些？

- 在什么情况下，方法的头部必须列出可能抛出的异常?

- 若try语句结构中有多个catch()子句，这些子句的排列顺序与程序执行效果是否有关？为什么?
  相关；当try块抛出一个异常时，程序的流程首先转向第一个catch块，并审查当前异常对象可否被这个catch块所接受。能接收是指异常
- 什么是抛出异常？系统定义的异常如何抛出？用户自定义的异常又如何抛出?

- 系统定义的异常与用户自定义的异常有何不同？如何使用这两类异常?

第 10 章 Java语言的输入输出与文件处理

- 什么是文件的输入与输出?
  文件的输入与输出即以文件作为数据输入流或数据输出流
- 什么是流？Java语言中分为哪两种流？这两种流有何差异?
  流是指计算机各部件之间的数据流动；字节流和字符流；
- InputStream、OutputStream、Reader和Writer四个在在功能上有何异同?
  InputStream和OutputStream是用来处理以位为单位的流
- 利用基本输入输出流实现从键盘上读入一个字符，然后显示了屏幕上。

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

public class Exercise {
    public static void main(String[] args) throws IOException {
        byte[] c = new byte[2];
        InputStream in = new DataInputStream(System.in);
        in.read(c);
        OutputStream out = new DataOutputStream(System.out);
        out.write(c);
        in.close();
        out.close();
    }
}
```

- 顺序输入输出流与管道输入输出流的区别是什么?
  顺序输入流其功能是将多个输入流顺序连接在一起，形成单一的输入数据流，没有对应的输出数据流存在。
  管道流用来将一个程序或线程的输出连接到另外一个程序或线程作为输入，从而可以实现程序内部线程间的通信或不同程序间的通信

- Java语言中定义的三个标准输入输出流是什么？他们对应什么设备？
  标准输入流、标准输出流、标准错误流；
  标准输入流对应的设备是键盘
  标准输出流和标准错误流对应的设备是显示器
- 利用文件输出流创建一个文件file1.txt，写入字符“文件已被成功创建！”，然后用记事本打开该文件，看一下是否正确写入。

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;

public class Exercise {
    public static void main(String[] args) throws IOException {
        File file = new File("file1.txt");
        FileOutputStream out = new FileOutputStream(file);
        String s = new String("文件已被创建成功!");
        byte[] bytes = s.getBytes("UTF-8");
        out.write(bytes);
        out.flush();
        out.close();
    }
}
```

- 利用文件输入流打开第7题中创建的file1.txt，读出其内容并显示在屏幕上。

```
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;

public class Exercise {
    public static void main(String[] args) throws IOException {
        File file = new File("file1.txt");
        FileInputStream in = new FileInputStream(file);
        byte[] bytes = new byte[1024];
        while (in.read(bytes) != -1) {
            System.out.println(new String(bytes));
        }
        in.close();
    }
}
```

- 利用文件输入输出流打开第7题创建的文件file1.txt，然后在文件的末尾追加一行字符串“又添加了一行文字！”。

```java
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;

public class Exercise {
    public static void main(String[] args) throws IOException {
        File file = new File("file1.txt");
        FileOutputStream out = new FileOutputStream(file, true);
        String s = "又添加了一行文字！";
        byte[] bytes = s.getBytes("UTF-8");
        out.write(bytes);
        out.close();
    }
}
```

- 产生15个20~9999之间的随机整数，然后利用BufferedWriter类将其写入文件file2.txt中；之后再读取该文件中得数据并将他们以升序排序

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Arrays;

public class Exercise {
    public static void main(String[] args) throws IOException {
        File file = new File("file2.txt");
        BufferedWriter writer = new BufferedWriter(new
FileWriter(file));
        for (int i = 0; i < 15; i++) {
            String s = "" + ((int)(Math.random() * 9979) + 20) + "\n";
            writer.write(s);
        }
        writer.flush();
        writer.close();

        BufferedReader reader = new BufferedReader(new
FileReader(file));
        int[] nums = new int[15];
        for (int i = 0; i < 15; i++) {
            nums[i] = Integer.parseInt(reader.readLine());
        }
        reader.close();
        Arrays.sort(nums);
        writer = new BufferedWriter(new FileWriter(file));
        for (int i = 0; i < 15; i++) {
            String s = "" + (nums[i]) + "\n";
            writer.write(s);
        }
        writer.flush();
        writer.close();

    }
}
```

- Java语言中使用什么类来对文件与文件夹进行管理?

  java.io.File

第 11 章 多线程

- 简述线程的基本概念，程序、进程、线程的关系是什么。
  线程与进程相似，但线程是一个比进程更小的执行单位。
  程序是静态的代码，进程是程序的一次执行过程，是系统运行程序的基本单位；线程是进程的更小的划分
- 什么是多线程？为什么程序的多线程功能是必要的?
  多线程就是同事执行一个以上的线程，一个线程的执行不必等待另一个线程执行完后才执行，所有线程都可以发生在同一时刻；单一的进程在执行任务会出现资源空闲，采用多线程可以让CPU在同一个时间之内执行一个程序中得好几个程序段来完成工作
- 多线程与多任务的差异是什么?
  多任务是针对操作系统而言的，表示操作系统可以同事运行多个应用程序，而多线程是对一个进程而言的，表示在一个进程内部可以同时执行多个线程。
- 线程有哪些基本状态？这些状态是如何定义的?
  新建状态、就绪状态、阻塞状态、运行状态、消亡状态。
  新建状态：线程对象已经被分配了内存空间和其他资源，并已被初始化，但是该线程尚未被调度。
  就绪状态：处于新建状态的线程被启动后，将进入线程队列排队等待CPU时间片，此时它以具备了运行的条件。
  运行状态：线程正在运行，并且已经拥有了对CPU的控制权
  阻塞状态：正在执行的线程如果在某些特殊情况下，将让出CPU并暂时中止自己的执行
  消亡状态：线程不再具有继续运行的能力
- Java程序实现多线程有哪两个途径?
  继承Thread类，实现Runnable接口
- 在什么情况下必须以类实现Runnable接口来创建线程?
  当类已经继承了一个类的时候，由于Java是单继承的，所以必须实现Runnable接口才能创建线程
- 什么是线程的同步？程序中为什么要实现线程的同步？是如何实现同步的?
  当一个线程对共享的数据进行操作时， 应使之成为一个"原子操作"，即在没有完成相关操作之前，不允许其他线程打断它，否则，就会破坏数据的完整性，必然会得到错误的处理结果；线程间的数据共享会导致多个线程同时处理数据而使数据出现不一致，所以要实现线程的同步；通过线程间互斥访问临界资源来实现同步。
- 假设某家银行可接受顾客的汇款，每进行一次汇款，便可计算出汇款的总额。现有两名顾客，每人分三次、每次100元将钱汇入。试编程来模拟顾客的汇款操作。

```
public class Exercise {
    public static void main(String[] args) {
        Customer customer1 = new Customer();
        Customer customer2 = new Customer();
        customer1.start();
        customer2.start();
    }
}

class Customer extends Thread {

    @Override
    public void run() {
        for (int i = 0; i < 3; i++) {
            Bank.deposit(100);
        }
    }
}

class Bank {
    public static int balance = 0;
    public synchronized static void deposit(int b) {
        int temp = balance;
        temp += b;
        try {
            Thread.sleep((int) (Math.random() * 1000));
        } catch (InterruptedException e) {

        }
        balance = temp;
        System.out.println("汇款之后" + balance);
    }
}
```

第 12 章 泛型与容器类

- 什么是泛型的类型参数？泛型的主要优点是什么？在什么情况下使用泛型方法？泛型类与泛型方法的主要区别是什么？
  泛型所操作的数据类型被指定为一个参数，这个参数称为类型参数；泛型的主要优点是能够在编译时而不是在运行时检测出错误； 返回值类型和至少一个参数类型是泛型的情况下使用泛型方法；对于泛型方法，不需要把实际的类型传递给泛型方法，但泛型类却恰恰相反，即必须把实际的类型参数传递给泛型类
- 已知Integer是Number的子类，GeneralType是General的子类吗？Generla是General的父类吗？
  不是；不是

- 在泛型中，类型通配符的主要作用是什么？
  一是用在泛型类创建泛型对象时；二是用在方法的参数中
- 分别件数LinkedList和ArrayList、HashSet与HashMap和TreeMap有何异同。
  相同点：
  - LinkedList和ArrayList都实现了List接口
  - HashSet和TreeSet都实现了Set接口

  不同点：
  - LinkedList采用链表结构保存对象，使用双链表实现List。这种结构向链表中任意位置插入、删除元素时不需要移动其他元素，链表的的大小可以动态增长或减小的，但不具有随机存取特性
  - ArrayList数组列表类使用一维数组实现List，该类实现的是可变数组，允许所有元素，包括null。具有随机存取特性，插入、删除元素是需要移动其他元素，当元素很多时，插入、删除的速度较慢。在向ArrayList中添加元素时，其容量会自动增大，但不能自动缩小
  - HasetSet是根据哈希码来存取集合中的元素，集合中的元素是无序的
  - TreeSet还实现了SortedSet接口，集合中的元素都是有序的
- 将1~10之间的整数存放到一个线性表LinkedList的对象中，然后将其下标为4的元素从列表中删除。

```
import java.util.LinkedList;

public class Exercise {
    public static void main(String[] args) {
        LinkedList<Integer> linkedList = new LinkedList<>();
        for (int i = 1; i <= 10; i++) {
            linkedList.add(i);
        }
        linkedList.remove(4);
    }
}
```

- 利用ArrayList类创建一个对象，并向其添加若干个字符串型元素，然后随机选一个元素输出。

```
import java.util.ArrayList;

public class Exercise {
    public static void main(String[] args) {
        ArrayList<String> arrayList = new ArrayList<>();
        arrayList.add("Hello");
        arrayList.add("World");
        arrayList.add("Java");
        System.out.println(arrayList.get((int)(Math.random() *
(arrayList.size() - 1))));
    }
}
```

- 集合A={1,2,3,4}和B={1,3,5,7,9,11}，编程求A与B的交集、并集和差集。

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

public class Exercise {
    public static void main(String[] args) {
        Set<Integer> a = new HashSet<>();
        Set<Integer> b = new HashSet<>();
        for (int i = 1; i <= 4; i++) {
            a.add(i);
        }
        for (int i = 1; i <= 11; i += 2) {
            b.add(i);
        }
        a.retainAll(b);
        a.addAll(b);
        a.remove(b);
    }
}
```

- 利用随机函数生成10个随机数，并将他们有序地存入到一个TreeSet对象中，然后利用迭代器有序地输出。

```
import java.util.Iterator;
import java.util.TreeSet;

public class Exercise {
    public static void main(String[] args) {
        TreeSet<Double> treeSet = new TreeSet<>();
        for (int i = 0; i < 10; i++) {
            treeSet.add(Math.random());
        }
        Iterator<Double> it = treeSet.iterator();
        while (it.hasNext()) {
            System.out.println(it.next());
        }
    }
}
```

- 利用HashMap类对象存储公司电话号码薄，其中包含公司的电话号码和公司名称，然后删除一个公司和查询一个公司的操作。

```java
import java.util.HashMap;

public class Exercise {
    public static void main(String[] args) {
        HashMap<String, String> number = new HashMap<>();
        number.put("A", "11111111");
        number.put("B", "22222222");
        number.put("C", "33333333");
        number.put("D", "44444444");
        number.remove("A");
        number.containsKey("C");
    }
}
```

第 13 章 图形界面设计

- Swing组件分为哪三类?
  容器类（container class）、辅助类(hepler class)和组件类(component class)
- Swing的顶层容器包含哪些窗格？大部分的可见组件都放在哪个窗格中?
  JFrame、JApplet和JDialog；JFrame
- 颜色类Color中提供了哪些表示颜色的变量?
  BLACK、BLUE、CYAN、DARK_GRAY、GRAY、GREEN、LIGHT_GRAY、MAGENTA、ORANGE、PINK、RED、WHITE、YELLOW
- 字体类Font的主要用途有哪些?
  设置组件所用字体的样式、大小与字形等属性。
- 图标类ImageIcon的主要作用是什么？Java语言当前支持哪三种图像格式?
  装饰组件；GIF、JPEG、PNG
- Swing主要用来处理文字输入的组件有几个？这几组件有何不同?
  JTextField、JPasswordField、JTextArea
  单行文本编辑组件JTextField、密码文本行组件JPasswordField、多行文本编辑组件JTextArea
- 在将组件添加到窗口中时，为什么需将组件设置为不透明状态才能将其底色显示出来?
  因为JFrame的框架一旦被创建，其中就已经包含了一个内容窗格，设置的JFrame背景颜色，仍然会被内容窗格遮盖，由框架内容窗格委托特性可知，在向JFrame中添加组件时，组件都加在了内容窗格中
- 如何设置才能将窗口的底色显示出来?
  通过调用JFrame的成员方法getContentPane()获得，然后再设置内容窗格的背景色，这样才能显示出窗口的背景色
- 设计一个窗口，内含一个文本框、三个复选框、两个单选框、一个标签和一个按钮。各组件的位置、大小和其上的文字由用户自己设定。

```java
import java.awt.FlowLayout;

import javax.swing.ButtonGroup;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JRadioButton;
import javax.swing.JTextField;

public class Exercise {
    public static void main(String[] args) {
        JFrame window = new JFrame("A Windows");
        window.setLayout(new FlowLayout());

        JTextField textField = new JTextField(20);
        window.add(textField);

        JCheckBox checkBox1 = new JCheckBox("CheckBox1");
        JCheckBox checkBox2 = new JCheckBox("CheckBox2");
        JCheckBox checkBox3 = new JCheckBox("CheckBox3");
        window.add(checkBox1);
        window.add(checkBox2);
        window.add(checkBox3);

        JRadioButton radioButton1 = new
JRadioButton("RadioButton1");
        JRadioButton radioButton2 = new
JRadioButton("RadioButton2");
        ButtonGroup bg = new ButtonGroup();
        bg.add(radioButton1);
        bg.add(radioButton2);

        window.add(radioButton1);
        window.add(radioButton2);

        JLabel label = new JLabel("This is a label");
        window.add(label);

        JButton button = new JButton("Button");
        window.add(button);

        window.pack();
        window.setVisible(true);
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

- Java语言中常用的布局管理器有哪几个？它们各有什么特点?
  awt五种布局管理器，swing3种（其中两种不介绍）
  流式布局管理器FlowLayout：流式页面设计，组件按照加入容器的先后顺序从左向右排列，一行排满之后就自动地转到下一行继续从左向右排列，每一行中得组件默认设置为剧中排列
  边界式布局管理器BorderLayout：将显示区域分为东、西、南、北、中5个区域，在将组件加入容器时，指定把组件加入到哪个区域中，默认为中间区域
  网格式布局管理器GridLayout：将容器的空间划分为若干行与列的网格形式，每个网格称为单元格，在容器上添加组件时，它们会按从左向右、从上到下的顺序在网格中排列。
  卡片式布局管理器CardLayout：把容器中得所有组件如同堆叠起来的一副“扑克牌”，每次只能显示最上面的一张
  网格包布局管理器GridBagLayout：是在网格布局管理器的基础上发展而来的，但GridBagLayout布局管理器中允许组件占用不同行或者不同咧的多个单元格，这些被占用的单元格称为组件的显示区域
  盒式布局管理器BoxLayout：在一行或者一列中摆放组件

第 14 章 事件处理

- 什么是事件？简述Java语言的委托事件模型。
  所谓事件，就是用户使用鼠标或键盘对窗口中得组件进行交互时所发生的事情；委托事件模型就是组件将事件处理委托给外部的处理实体
- 若要处理事件，就必须要有事件监听者，通常哪有对象可以担任监听者?
  包含“事件源”的容器对象、内部类对象和匿名内部类都可以担任监听者
- Java语言中处理事件的具体方法是什么?
  - 确认触发的事件，取得事件类的名字，如ActionEvent，去掉其中的“Event”字样，在剩下的部分加入“Listener”，这就是在类里需要实现的事件监听者接口
  - 实现上述的接口，针对想要捕获的事件编写方法代码。如要捕获单击按钮事件，就要为ActionListener接口中的actionPerformed()方法编写代码
  - 为事件监听者创建一个对象，让组件调用方法完成对它的注册，方法是在监听者接口的名字中加入一个前缀“add"，如addActionListener()
- 在进行事件处理时，可以使用实现多个监听者接口的方式，也可以尽可能地使用继承适配器类的方式。使用适配器类的好处是什么?
  当需要对某件事件进行处理时，只需让事件处理类继承事件所对应的适配器类，这样只需要覆盖本次操作用到的事件处理方法即可，而不必实现无关的事件处理方法
- 设计一个窗口，在窗口内放置一个按钮，当不断地单击该按钮时，在其上显示它被单击的次数。

```
import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;

public class Exercise {
    public static void main(String[] args) {
        new Window();
    }
}

class Window extends JFrame{
    private static final long serialVersionUID =
-2929177477951330739L;
    private JButton button;
    private int time;
    private JLabel label;
    public Window() {
        button = new JButton("Click Me");
        label = new JLabel("0");
        label.setHorizontalAlignment(JLabel.CENTER);
        time = 0;
        button.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                time++;
                label.setText("" + time);
            }
        });

        add(button, BorderLayout.NORTH);
        add(label, BorderLayout.SOUTH);
        pack();
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

- 设计一个窗口，在该窗口中添加一个JList组件，该组件中有5门课程名称的选项。然后再在窗口中添加一个文本区，当选择JList组件中的某个选项后，文本区中显示对该课程的介绍。

```
import java.awt.BorderLayout;

import javax.swing.JFrame;
```

```
import javax.swing.JList;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;

public class Exercise {
    public static void main(String[] args) {
        new Window();
    }
}

class Window extends JFrame {

    private static final long serialVersionUID =
-2929177477951330739L;
    private JList<String> courses;

    public Window() {
        String[] model = { "English", "Chinese", "C Programming
Language", "Java Programming Language",
                "Operation System" };
        courses = new JList<>(model);

        JScrollPane scrollPane = new JScrollPane(courses);
        add(scrollPane, BorderLayout.CENTER);

        JTextArea textArea = new JTextArea();
        textArea.setEditable(false);
        add(textArea, BorderLayout.SOUTH);

        courses.addListSelectionListener(new ListSelectionListener()
{
            @Override
            public void valueChanged(ListSelectionEvent e) {
                String s = courses.getSelectedValue();
                switch (s) {
                case "English":
                    textArea.setText("This is English");
                    break;
                case "Chinese":
                    textArea.setText("This is Chinese");
                    break;
                case "C Programming Language":
                    textArea.setText("This is C Programming
Language");
                    break;
                case "Java Programming Language":
```

```
                    textArea.setText("This is Java Programming
Language");
                    break;
                case "Operation System":
                    textArea.setText("This is Operation System");
                    break;
                }
            }
        });

        pack();
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

- 设计一个窗口，其中包含一个文本区、两个复选框和三个横向滑动条，其中滑动条用来调整R、G、B三色的分量从0~255的变化；两个复选框分别用于设定把滑动条调出的颜色应用于文本区的前景色还是背景色。

```
import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.BoxLayout;
import javax.swing.JCheckBox;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JSlider;
import javax.swing.JTextArea;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

public class Exercise {
    public static void main(String[] args) {
        new Window();
    }
}

class Window extends JFrame {

    private static final long serialVersionUID =
-4682287151007612144L;
    JCheckBox frontGroundColor = new JCheckBox("frontGroundColor");
    JCheckBox backGroundColor = new JCheckBox("backGroundColor");
    JTextArea textArea = new JTextArea("选择前景色和背景色，滑动滑动条查
看颜色变化", 20, 30);
    JSlider rJSlider = new JSlider(JSlider.HORIZONTAL, 0, 255, 0);
```

```
    JSlider gJSlider = new JSlider(JSlider.HORIZONTAL, 0, 255, 0);
    JSlider bJSlider = new JSlider(JSlider.HORIZONTAL, 0, 255, 0);

    public Window() {
        setLayout(new BoxLayout(getContentPane(),
BoxLayout.Y_AXIS));

        add(textArea);

        JPanel select = new JPanel();
        select.setLayout(new FlowLayout());
        select.add(frontGroundColor);
        select.add(backGroundColor);
        add(select);

        JPanel colorPanel = new JPanel();
        colorPanel.setLayout(new BoxLayout(colorPanel,
BoxLayout.Y_AXIS));
        colorPanel.add(rJSlider);
        colorPanel.add(gJSlider);
        colorPanel.add(bJSlider);
        rJSlider.setMajorTickSpacing(30);
        rJSlider.setMinorTickSpacing(3);
        rJSlider.setPaintTicks(true);
        rJSlider.setPaintLabels(true);
        gJSlider.setMajorTickSpacing(30);
        gJSlider.setMinorTickSpacing(3);
        gJSlider.setPaintTicks(true);
        gJSlider.setPaintLabels(true);
        bJSlider.setMajorTickSpacing(30);
        bJSlider.setMinorTickSpacing(3);
        bJSlider.setPaintTicks(true);
        bJSlider.setPaintLabels(true);
        add(colorPanel);

        backGroundColor.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                if (backGroundColor.isSelected()) {
                textArea.setBackground(new
Color(rJSlider.getValue(), gJSlider.getValue(),
bJSlider.getValue()));
                } else {
                    textArea.setBackground(new Color(255, 255,
255));
                }
            }
        });
```

```
        frontGroundColor.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                if (frontGroundColor.isSelected()) {
                    textArea.setForeground(new
Color(rJSlider.getValue(), gJSlider.getValue(),
bJSlider.getValue()));
                } else {
                    textArea.setForeground(new Color(0, 0, 0));
                }
            }
        });

        rJSlider.addChangeListener(new ColorListen());
        gJSlider.addChangeListener(new ColorListen());
        bJSlider.addChangeListener(new ColorListen());

        pack();
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    class ColorListen implements ChangeListener {
        @Override
        public void stateChanged(ChangeEvent e) {
            if (frontGroundColor.isSelected()) {
                textArea.setForeground(new
Color(rJSlider.getValue(), gJSlider.getValue(),
bJSlider.getValue()));
            }
            if (backGroundColor.isSelected()) {
                textArea.setBackground(new
Color(rJSlider.getValue(), gJSlider.getValue(),
bJSlider.getValue()));
            }
        }
    }
}
```

- 编写一应用程序，在其窗口内包含一个菜单栏和一个文本区。菜单栏包括“设置”和“操作”两个菜单。“操作”菜单包括“退出”菜单项，当用户选择“退出”菜单项时，则关闭窗口退出整个应用程序的运行；“设置”菜单包括“字体”和“风格”两个菜单项和一个“只读”复选菜单项。“字体”菜单项包括“宋体”、“楷体”和“黑体”3个单选子菜单项，“风格”菜单项包括“普通”、“黑体”、“斜体”3个复选子菜单项。当“只读”菜单项未被选中时，用户可以了文本区内输入字符；当“只读”菜单项被选中时，用户不能在文本区内输入字符。当用户选择其它菜单项时，文本区内的文字随之变化

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.BoxLayout;
import javax.swing.ButtonGroup;
import javax.swing.JCheckBoxMenuItem;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JRadioButtonMenuItem;
import javax.swing.JTextArea;

public class Exercise {
    public static void main(String[] args) {
        new Window();
    }
}

class Window extends JFrame {

    private static final long serialVersionUID =
-4682287151007612144L;
    JTextArea textArea = new JTextArea(20, 30);

    public Window() {
        setLayout(new BoxLayout(getContentPane(),
BoxLayout.Y_AXIS));

        JMenuBar menuBar = new JMenuBar();
        JMenu setting = new JMenu("Setting");
        JMenu operating = new JMenu("Operating");
        menuBar.add(setting);
        menuBar.add(operating);

        JMenuItem exit = new JMenuItem("Exit");
        exit.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                System.exit(0);
            }
        });
        operating.add(exit);

        JMenu style = new JMenu("Style");
        JMenu font = new JMenu("Font");
        ButtonGroup buttonGroup = new ButtonGroup();
        JRadioButtonMenuItem ssti = new JRadioButtonMenuItem("宋
体");
```

```
        JRadioButtonMenuItem klti = new JRadioButtonMenuItem("楷
体");
        JRadioButtonMenuItem hzti = new JRadioButtonMenuItem("黑
体");
        buttonGroup.add(ssti);
        buttonGroup.add(klti);
        buttonGroup.add(hzti);
        font.add(ssti);
        font.add(klti);
        font.add(hzti);
        JCheckBoxMenuItem normal = new JCheckBoxMenuItem("正常");
        JCheckBoxMenuItem black = new JCheckBoxMenuItem("黑体");
        JCheckBoxMenuItem till = new JCheckBoxMenuItem("斜体");
        style.add(normal);
        style.add(black);
        style.add(till);
        JRadioButtonMenuItem readOnly = new
JRadioButtonMenuItem("Read-Only");
        setting.add(style);
        setting.add(font);
        setting.add(readOnly);

        add(menuBar);

        add(textArea);

        readOnly.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                if (readOnly.isSelected()) {
                    textArea.setEditable(false);
                } else {
                    textArea.setEditable(true);
                }
            }
        });

        addText(normal);
        addText(black);
        addText(till);
        addText(ssti);
        addText(klti);
        addText(hzti);
        addText(readOnly);

        setVisible(true);
        pack();
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
```

```
    private void addText(JMenuItem item) {
        item.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                textArea.append(item.getText() + "\n");
            }
        });
    }
}
```

- 在上一题的基础上增加如下的功能：每当用户选中“只读”菜单项时，都将“字体”和“风格”两个菜单变成灰色，使之不能被选中；而每当“只读”菜单项未被选中时，再将“字体”和“风格”两个菜单项恢复成可选状态。

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.BoxLayout;
import javax.swing.ButtonGroup;
import javax.swing.JCheckBoxMenuItem;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JRadioButtonMenuItem;
import javax.swing.JTextArea;

public class Exercise {
    public static void main(String[] args) {
        new Window();
    }
}

class Window extends JFrame {

    private static final long serialVersionUID =
-4682287151007612144L;
    JTextArea textArea = new JTextArea(20, 30);

    public Window() {
        setLayout(new BoxLayout(getContentPane(),
BoxLayout.Y_AXIS));

        JMenuBar menuBar = new JMenuBar();
        JMenu setting = new JMenu("Setting");
```

```
        JMenu operating = new JMenu("Operating");
        menuBar.add(setting);
        menuBar.add(operating);

        JMenuItem exit = new JMenuItem("Exit");
        exit.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                System.exit(0);
            }
        });
        operating.add(exit);

        JMenu style = new JMenu("Style");
        JMenu font = new JMenu("Font");
        ButtonGroup buttonGroup = new ButtonGroup();
        JRadioButtonMenuItem ssti = new JRadioButtonMenuItem("宋
体");
        JRadioButtonMenuItem klti = new JRadioButtonMenuItem("楷
体");
        JRadioButtonMenuItem hzti = new JRadioButtonMenuItem("黑
体");
        buttonGroup.add(ssti);
        buttonGroup.add(klti);
        buttonGroup.add(hzti);
        font.add(ssti);
        font.add(klti);
        font.add(hzti);
        JCheckBoxMenuItem normal = new JCheckBoxMenuItem("正常");
        JCheckBoxMenuItem black = new JCheckBoxMenuItem("黑体");
        JCheckBoxMenuItem till = new JCheckBoxMenuItem("斜体");
        style.add(normal);
        style.add(black);
        style.add(till);
        JRadioButtonMenuItem readOnly = new
JRadioButtonMenuItem("Read-Only");
        setting.add(style);
        setting.add(font);
        setting.add(readOnly);

        add(menuBar);

        add(textArea);

        readOnly.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                if (readOnly.isSelected()) {
                    textArea.setEditable(false);
```

```
                    style.setEnabled(false);
                    font.setEnabled(false);
                } else {
                    textArea.setEditable(true);
                    style.setEnabled(true);
                    font.setEnabled(true);
                }
            }
        });

        addText(normal);
        addText(black);
        addText(till);
        addText(ssti);
        addText(klti);
        addText(hzti);
        addText(readOnly);

        setVisible(true);
        pack();
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    private void addText(JMenuItem item) {
        item.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                textArea.append(item.getText() + "\n");
            }
        });
    }
}
```

第 16 章 小程序设计

- Java小程序的基本工作原理是什么?
  将编译好的小程序字节码文件，即.class文件保存在特定的WWW服务器上，在同一个或另一个WWW服务器上保存着嵌入有该字节码文件名的HTML文件。当某一个浏览器向服务器请求下载嵌入了小程序的HTML文件时，该文件中得信息以一定的格式显示在用户屏幕上。当浏览器遇到HTML文件中嵌有小程序的标志时，浏览器会根据这个小程序的名字和位置自动把字节码文件从WWW服务器上下载到本地，并利用浏览器本身拥有的Java解释器直接执行该字节码文件。

- init()、start()、stop()、destory()是小程序中非常重要的4个方法，请问它们各自的调用时机和功能是什么?
  init()在小程序被创建时第一个调用的方法，它只运行一次，主要是用来对小程序设置初值用start()调用完init()方法之后，就立即调用start()方法。只要小程序画面每出现一次，start()方法就会被调用一次。
  stop()方法类似于start()方法的逆操作，当浏览器窗口失去焦点变为不活动状态、切换到其他网页浏览或是关闭浏览器时，需要停止小程序的运行，此时系统会自动调用stop()方法以暂停小程序的运行，所以stop()方法也可以被重复调用。
  destory()当用户退出浏览器时，浏览器运行的小程序也将停止运行，释放内存。此时浏览器会自动调用小程序对象的destory()方法来完成一些释放资源、关闭连接之类的操作等。
- 如何向小程序传递参数?
  由HTML文件中得一个专门标记来完成的
- 将Java的应用程序转换成小程序的主要步骤有哪些?
  - 制作一个HTML网页文件，带有响应的标记，从而能够下载小程序的代码
  - 声明一个类，使其继承JApplet类，并使其成为public类型
  - 在应用程序中去掉main()方法。
  - 在应用程序中，设置窗口的大小通过调用setSize()方法来实现
  - 小程序在浏览器退出时，它会终止
  - 如果在应用程序中使用setTitle()为窗口设置标题，那么在转换成小程序时此方法不能使用
  - 用init()方法替换构造方法，在浏览器创建这个小程序类的一个对象时，它调用了init()方法
- 编写小程序，用paintComponent()方法显示一行字符串，小程序包含“方法”和“缩小”两个按钮。当单击“放大”按钮时，显示的字符串的字号方法一号；单击“缩小”按钮时，显示的字符串字号缩小一号。

```
import java.awt.Container;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JApplet;
import javax.swing.JButton;
import javax.swing.JTextArea;

public class Exercise extends JApplet {

    private static final long serialVersionUID =
-6178049653652743932L;
    private String s;
    private JButton big;
    private JButton smaller;
    private int size = 16;
    private JTextArea textArea;
    @Override
    public void init() {
```

```
        s = "This is a string";
        big = new JButton("放大");
        smaller = new JButton("缩小");
        textArea = new JTextArea(s);
        textArea.setFont(new Font(s, Font.ITALIC, size));

        Container c = getContentPane();
        c.setLayout(new FlowLayout());
        c.add(textArea);
        c.add(big);
        c.add(smaller);

        big.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                size++;
                textArea.setFont(new Font(s, Font.ITALIC, size));
            }
        });
        smaller.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                size--;
                textArea.setFont(new Font(s, Font.ITALIC, size));
            }
        });
    }
}
```

第 17 章 Java数据库程序设计

- 写出在数据库StudentScore的Student表中查找所有年龄大于等于19的同学的SQL语句
  SELECT * FROM StudentScore.Student WHERE age >= 19;
- 写出姓名为“刘韵”的学生所学课程名称及成绩的SQL语句
  SELECT cName, grade FROM Course, Score WHERE Course.cNo = Score.cNo AND sNo IN (SELECT sNo FROM Student WHERE sName = "刘韵")
- 描述JDBC中Driver、Connection、Statement和ResultSet接口的功能

类或接口	功能说明
Driver	驱动程序
Connection	数据库连接，负责与数据库键通讯，SQL执行以及食物处理都是在某个特定Connection环境下进行的。可以产生用以执行SQL的Statement对象
Statement	用以执行不含参数的静态SQL查询和更新，并返回执行结果
ResultSet	用以获得SQL查询结果

- 使用Statement接口和PreparedStatement接口有什么区别?
  Statement接口用于执行不带参数的静态SQL语句；PreparedStatement是处理预编译语句的一个接口
- 归纳一下使用JDBC进行数据库连接的完整过程
  加载驱动程序、建立与数据库的连接、创建执行方式语句、执行SQL语句、处理返回结果和关闭创建的各种对象
- 如何在结果集中返回字段的数据？如何在结果集中返回字段名?
  使用ResultSet对象的getXXX()方法对结果集中得数据进行访问；
- 编写一个应用程序，使其可以从StudentScore数据库的某个表中查询一个字段的所有信息

```
import java.sql.*;

public class Exercise {
    public static void main(String[] args) {
        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/Student",
"user", "password");
            String sql = "SELECT sNo FROM StudentScore.Student";
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery(sql);
            while (resultSet.next()) {
                String sno = resultSet.getString("sNo");
                System.out.println(sno);
            }
            resultSet.close();
            statement.close();
            connection.close();

        } catch (ClassNotFoundException e) {
            System.out.println("The Drive not found.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

- 创建一个名为Books的数据库，并在其中建立一个名为Book的表，字段包括书名、作者、出版社、出版时间和ISBN。编写一个应用程序，运用JDBC在该数据库中实现增加、删除和修改数据的功能。

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class Exercise {
    public static void addRecord (Connection connection) {

        String bookName = "h";
        String author = "j";
        String publisher = "j";
        String publisherTime = "j";
        String isbn = "k";

        String sql = "INSERT INTO Books.Book(bookName, author,
publisher, publisherTime, isbn) " +
                "VALUES (?, ?, ?, ?, ?)";
```

```
        PreparedStatement preparedStatement = null;
        try {
            connection.prepareStatement(sql);
            preparedStatement.setString(1, bookName);
            preparedStatement.setString(2, author);
            preparedStatement.setString(3,publisher);
            preparedStatement.setString(4, publisherTime);
            preparedStatement.setString(5, isbn);
            preparedStatement.execute();
            preparedStatement.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (NullPointerException e) {
            e.printStackTrace();
        }
    }

    public static void removeRecord (Connection connection) {
        String bookName = "j";
        String sql = "DROP FROM Books.Book WHERE bookName = ?";
        PreparedStatement preparedStatement = null;
        try {
            preparedStatement = connection.prepareStatement(sql);
            preparedStatement.setString(1, bookName);
            preparedStatement.execute();
            preparedStatement.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public static void updateRecode (Connection connection) {
        String bookName = "j";
        String publisher = "k";
        PreparedStatement preparedStatement = null;
        String sql = "UPDATE Books.Book SET publisher = ? WHERE
bookName = ?";
        try {
            preparedStatement = connection.prepareStatement(sql);
            preparedStatement.setString(1, publisher);
            preparedStatement.setString(2, bookName);
            preparedStatement.execute();
            preparedStatement.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

- 假设在StudentScore数据库的Studnet表中，存在多个姓氏相同的人，根据这种情况建立查询，要求提供一个合适的图形界面，用户可以滚动查看查询记录

第 18 章 Java网络编程

- 什么是URL？URL地址由哪几部分组成?
  URL是统一资源定位符的英文缩写，它表示Internet上某一资源的地址；URL地址由5个部分组成，格式如下：
  传输协议://主机名:端口号/文件名#引用
- 什么是Socket？它与TCP/IP协议有何关系?
  Socket是实现 客户与服务器模式的通信方式，它首先需要建立稳定的连接，然后以流的方式传输数据，实现网络通信；Socket在TCP/IP中定义，针对一个特定的连接
- 简述流式Socket的通信机制。它的最大特点是什么？为什么可以实现无差错通信?
  建立两台计算机的连接，有一个程序向另一台计算机上的程序发送请求，建立通信；
- 什么是端口号？服务器端和客户端分别如何使用端口号?
  端口号是区分一台主机中得不同应用程序的一个数据标识符；服务器端和客户端必须为每个程序分配一个唯一的端口考，通过端口号指定要连接的那个程序
- 什么是套接字？其作用是什么?
  套接字是实现客户与服务器模式的通信模式，它首先需要建立稳定的连接，然后以流的方式传输数据，实现网络通信；套接字是网络上运行的两个程序间双向通信的一端，既可以接受请求，也可以发送请求，利用它可以较方便地进行网络上的数据传输。
- 编写Java程序，使用InetAddress类实现根据域名自动到DNS（域名服务器）上查找IP地址的功能。

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.InetAddress;
import java.net.UnknownHostException;

public class Exercise {
    public static void main(String[] args) throws IOException {
        BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(System.in));
        String dns = bufferedReader.readLine();
        bufferedReader.close();
        InetAddress inetAddress = null;
        try {
            inetAddress = InetAddress.getByName(dns);
            System.out.println(inetAddress.getHostAddress());
        } catch (UnknownHostException e) {
            System.out.println("Your Input Domain Can't Parse.");
        }
    }
}
```

- 用Java程序实现流式Socket通信，需要使用那两个类？他们是如何定义的？应怎样使用？
  Socket类和ServerSocket类；Socket类用在客户端，用户通过创建一个Socket对象来建立与服务器的连接，ServerSocket类的作用是实现客户机/服务器模式的通信方法下服务器的套接字；客户端的程序使用Socket类建立一服务器的套接字连接，服务器端的程序使用ServerSocket类建立接受客户套接字的服务器套接字；
- 与流式Socket相比，数据报通信有何特点？
  数据报通信是无序的，也不能确保绝对安全可靠，但它非常简单，也具有比较高的效率