有问题请联系　李玮玮 QQ: 949412543

作业清单

序号	作业
1	熟悉 Java 程序基本语法（个人信息打印、算数运算）
2	程序执行流程（冒泡排序、乘法口诀表、计算周薪）
3	类和对象（在学生数组中查询指定学生信息、统计不及格名单）
4	封装和继承（汽车竞速）
5	抽象类和接口（形状定义及面积周长计算、空间点距离计算）
6	多态（饲养员饲养动物并根据动物类别喂食）
7	枚举（根据成绩判定学生等级）
8	异常处理（冰箱装大象）
9	容器（模拟给抢装子弹、射击）
10	流和文件（用文件在本地存储学生信息）
11	字符串、日期格式化（计算生日、解析字符串）
12	线程和多线程（工厂不同车间加工玩具）
13	网络编程（启动线程下载网络图片、使用 Socket 技术实现即时通信）
14	JDBC 数据库连接技术（数据库操作）
15	课程设计（图书管理系统）

雪梨作业

1 熟悉 Java 程序基本语法

1. 请使用 Eclipse 编写简单程序，实现如下功能:

打印个人信息（包括：姓名、学号、方向名称），打印结果如下所示：

姓名：鹿晗

学号：2016012001

方向名称：2016 级智能设备 1 班

2. 在类中定义一个算数运算的方法，要求能够根据参数（+、-、*、/）不同，实现对应的算数运算操作。

3. 在入口方法中，测试题目 2 中的方法（即调用题目 2 中的方法，验证结果的正确性）。

2 程序执行流程

1. 使用冒泡排序法实现对给定整形数组排序。

2. 实现倒着的乘法口诀表，即第 1 行是 9＊1 到 1＊1，最后一行是 9＊9，打印结果如下所示：

```
9*1=9   8*1=8   7*1=7   6*1=6   5*1=5   4*1=4   3*1=3   2*1=2   1*1=1
9*2=18  8*2=16  7*2=14  6*2=12  5*2=10  4*2=8   3*2=6   2*2=4
9*3=27  8*3=24  7*3=21  6*3=18  5*3=15  4*3=12  3*3=9
9*4=36  8*4=32  7*4=28  6*4=24  5*4=20  4*4=16
9*5=45  8*5=40  7*5=35  6*5=30  5*5=25
9*6=54  8*6=48  7*6=42  6*6=36
9*7=63  8*7=56  7*7=49
9*8=72  8*8=64
9*9=81
```

3. 编写一个 JAVA 程序，计算并显示某人根据下面规则确定的周薪：

如果工作时间小于等于 40 小时，那么每小时 30 元；否则，该职员在 1200 元基础上，对超过 40 小时的部分，再加上 45 元每小时。

要求：

程序应该根据输入工作的时间数，计算每周的薪水并显示出来。

3 类和对象

定义学生类用来存储学生的信息(学号，姓名，英语成绩、高数成绩、体育成绩)，用数组存储 10 名学生的信息。

(1) 按名字查询某位同学成绩，要求能够实现部分匹配的查找，例如：希望查找 John，可查找到所有名字包含 John 的人，例如：John Brown，John Smith 都能得到（可以使用字符串 String 的方法 contains，来判断是否包含某字符串）

(2) 查询得到所有科目不及格的人数及名单

提示：

（1）定义学生类（包括属性和方法（方法包括构造器、设置器、访问器））

（2）类中其他方法可自由发挥，比如打印各个属性的方法（如 print()或 toString()）

（3）自己定义包含 main 方法的类，类中定义 2 个方法，分别用来实现按名字查询成绩、查找不及格人数并打印不及格名单的功能。

名词解释：

java 里面的构造器是一个与类同名的特殊的方法，称为构造方法，在创建类的对象时使用，用于对象初始化。

设置器是一个针对类的某个私有属性成员属性的有特殊命名要求的方法，如对 XXX 属性，应为 setXXX。主要提供对私有属性的改变，提供一个设置变量值的途径。

访问器是一个针对类的某个私有属性成员属性的有特殊命名要求的方法，如对 XXX 属性，应为 getXXX。主要提供对私有属性的值，提供一个获得变量值的途径。

4 封装和继承

建立一个汽车 Auto 类，包括轮胎个数、汽车颜色、车身重量、时速成员变量，提供至少 2 个构造方法，汽车拥有的方法包括加速、停车。

建立一个小汽车 Car 类，继承 Auto，添加空调、CD 成员变量，覆盖加速方法。提供 2 个构造方法，其中一个含参数的构造方法能够显示指定调用父类中某个含参数的构造方法。

其他要求：

创建一个工厂 Factory 的测试类，

1. 该类中包含 main 方法，该方法中构造汽车类和小汽车类的对象，由用户输入赛车的时长，以分钟为单位（如 30 分钟），时长需大于 10 分钟，调用两车的赛车方法。

2. 包含一个赛车方法 competition()。该方法的参数包含 1 个汽车对象，1 个小汽车对象，一个赛车时间（如 30 分钟）。

赛车规则：2 辆车赛车过程中可加速 1 次，加速时间取 10 分钟内的随机值，2 车同时出发，经过 30 分钟后判断车程，行程远的获胜，打印出最终赛车的赛况（包括名次、每辆车的信息及行驶车程）

5 抽象类和接口

1.根据如下所述创建 3 个类，并抽象出一个抽象父类 Shape，定义其用于计算面积的抽象方法，实现 3 个类的功能。

（1）创建 Circle 、Triangle、Rectangle 三个类（放入一个同名的 java 文件中）将其放入 org .edu2act .figure 包中。

（2）创建名为 ShapeTest 的测试类，编写 main()方法测试这三个类。

（3）其中圆的半径,三角形的三边,方形的长与宽都为 double 类型, 都不能为负(在构造函数中验证); 三角形三边关系: 任意两边之和都大于第三边。

提示：已知三角形的三边长，利用海伦——秦九昭公式求三角形面积：

已知三角形的三边分别是 a、b、c，

先算出周长的一半 s=1/2(a+b+c)

则该三角形面积：$\sqrt{s(s-a)(s-b)(s-c)}$

2.在第 1 题的基础上完善 3 个形状类，圆形（属性半径：r）、三角形（属性三边长：a、b、c）、方形 （属性长宽：a、b）三个形状类，并增加或修改计算周长、面积的成员方法。在测试类中测试这三个类。

3.实现一个接口，接口中有抽象方法 getDistance(Object obj)，可以计算距离。

定义二维空间点类（DoubleDimensionPoint），实现上述接口，并进行距离测试（2,3）与（8,9）点的距离

定义三维空间点类（TripleDimensionPoint），实现上述接口，并进行距离测试（2,1,3）与（1,8,9）点的距离。

提示：

两点之间距离公式：设(x1，y1)，(x2，y2)分别为 A、B 两个点的坐标

- $|AB| = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

两点间距离公式的三维坐标形式：两点的坐标为 $A(x_1, y_1, z_1)$ $B(x_2, y_2, z_2)$

- $|AB| = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$

JDK 中的某些 API：

四舍五入：Math.round(float a)、Math.round(double a)

开平方：Math.sqrt(double a)

幂次方：Math.pow(double a, double b)　表示求 a 的 b 次方

6 多态

按照如下描述定义类，实现动物园内饲养员对动物的喂养活动：猫吃鱼、狗吃肉、大象吃香蕉

1.动物类：Animal

属性：名字(name)

方法：构造方法、eat 方法(参数为：　食物类型的变量)

2.猫类 继承自 动物类：Cat

属性：学生自己定义

方法：构造方法、其他方法自己定义

3.狗类 继承自 动物类：Dog

属性：学生自己定义

方法：构造方法、其他方法自己定义

4.大象类 继承自 动物类：Elephant

属性：学生自己定义

方法：构造方法、其他方法自己定义

5.食物类:Food

属性：名字(name)

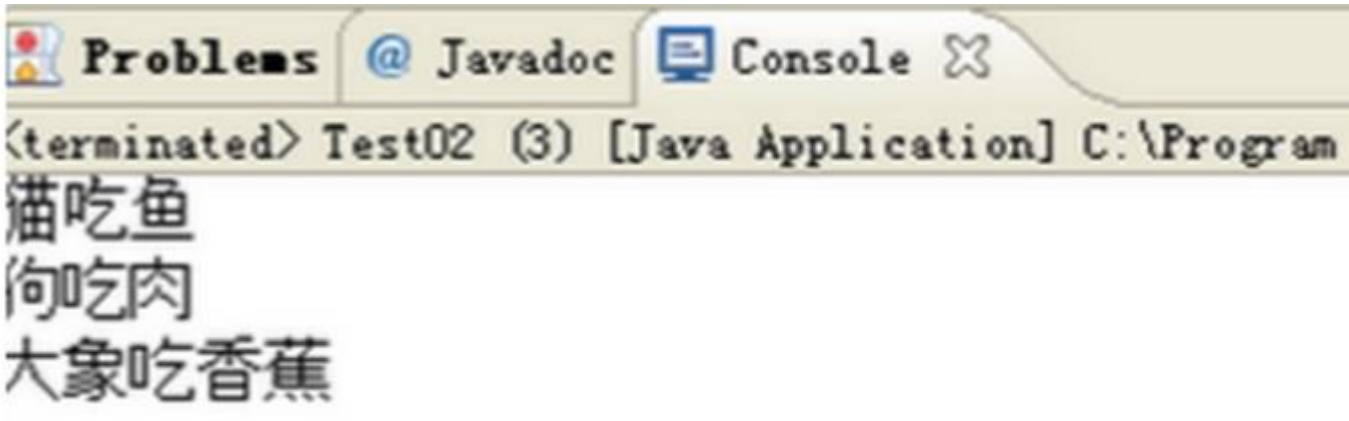
方法：构造方法

6.饲养员 Person

属性：学生自己定义

方法：喂养 feed 方法(参数为 ：动物类型 变量 食物类型 变量)

输出类似如下效果：



7 枚举

定义等级 Grade 枚举类型，其中等级可分为 ABCD 四个等级；

定义学生类，属性包括姓名、英语成绩、数学成绩、体育成绩、Grade 等级；

定义学生类的设置等级的方法：根据英语，数学，体育的平均成绩计算每个学生的等级，平均成绩大于等于 90 分为 A 级别，80-90 分为 B 级别，79-60 分为 C 级别，<60 为 D 级别。

在测试类中定义一个 10 个学生的数组，然后写程序实现四个不同等级中各自的学生的百分比

8 异常处理

1. 定义个大象类 Elephant，包含长、宽、高属性。
定义一个冰箱类 Fridge，包含长、宽、高 3 个属性；方法包括一个求体积的方法和一个装大象的方法。其中，装大象的方法要求，冰箱的体积以及长宽高能够容纳一只大象，否则抛出一个 Exception 异常。
定义一个测试类 StuffElephant，在 main 方法中测试用冰箱装一只大象的方法，并且捕获装大象方法可能会抛出的 Exception 异常，并使用 finally 模拟清空冰箱的操作。

2. 修改上一个题目，自定义体积过大的异常类（OutOfVolumeException），当冰箱装不下大象时，抛出此异常。

9 容器

使用容器来模拟枪射击。

1．定义子弹类：Ball,有属性 编号，型号。

默认构造方法，设编号为 1， 型号为“M54”。

带参数构造方法：设定编号和型号。

方法 1：取得编号

方法 2：取得型号。

方法 3：显示子弹的编号和型号： 编号-型号。

2.定义枪类：Gun

有属性：编号，型号，弹夹，装弹数量。

默认构造方法：创建编号为 1，型号为“M54”，装弹数量为 6。

带参数构造方法：与属性相同的参数，设定属性：编号，型号，数量的值。

装弹方法：将一个子弹对象装入弹夹，如果弹夹已满，抛出弹夹满异常。

射击方法：返回一个子弹对象，减少弹夹的一颗子弹，如果弹夹空，则抛出空异常。

显示弹夹内子弹列表信息方法：显示弹夹内的子弹信息：编号-型号

取得当前弹夹子弹数的方法。

3.编写带 main 方法的测试类，模拟枪的装弹和射击功能。

10 流和文件

1. 使用字节输入流和字节输出流读取并拷贝本地磁盘上的一张图片
2. 使用 BufferedReader 和 BufferedWriter 读取本地文本文件，并以追加的方式拷贝文件
3. 使用 Scanner 对象读取用户输入的学生信息，根据输入信息创建学生对象，并实现学生信息的序列化与反序列化操作。

具体要求如下：

（1） 定义学生类，其属性包括：姓名、年龄、成绩

（2） 定义测试类，在测试类中定义 List 集合，用于存储学生对象（至少存 3 个学生对象），分别调用 3 中方法，实现其功能，并打印（3）中第 c 个方法返回的结果。

（3） 在测试类中定义 3 个方法：

a. 初始化学生集合的方法（要求学生对象数据来源于用户在控制台的输入）

b. 序列化学生集合对象的方法（参数为带路径的文件名称、待序列化的对象）

c. 反序列化学生集合对象的方法（参数为带路径的文件名称，返回值是反序列化后的集合对象）

11 字符串、日期格式化

1.选择合适的日期类型表示现在的时间，并计算现在距离自己的生日还有多少天？

2.编写一个类，其功能是使用 SimpleDateFormate 类打印自己出生日期的两种不同格式的形式。

3. 解析字符串“I Love to Learn Java,Do you Love Too?”，要求计算字符串中单词的个数，将字符‘,’替换成英文空格，并计算出字符串中有几个字符‘o’。

12 线程和多线程

模拟玩具工厂加工玩具

创建车间类，属性包括：车间名称

创建工厂类，属性包括：车间类的对象的集合，仓库容量

工厂类的方法包括：

1. 加工玩具的方法(product)，方法的功能是各个车间同时工作。根据仓库容量平均分配给各个车间需要加工玩具的数量。

2. 建造玩具车间的方法(createWorkshop)，方法功能是模拟建造玩具生成车间，即向集合属性中添加一个新车间。

要求：

1. 使用多线程模拟加工玩具的功能，假设每隔 2 秒可完成一个玩具的加工，每个车间完成一个玩具的加工后，打印当前已加工玩具数量（给每个线程设置名称为车间的名称）

2. 创建测试类，模拟工厂加工玩具的功能。

提示：车间集合的容量即是线程的数量

13 网络编程

1. 使用 URL 与 HttpURLConnect 类下载网络上的两张图片，要求启动 2 个线程，分别下载一张图片。

提示：

1） 使用 Thread 或 Runnable 方法实现多线程，自定义的类中需包含 2 个属性，1 个是图片的地址，另一个是存储位置

2）下载图片的操作在线程中实现

2. 模拟多客户端和服务器之间通过 Socket 进行通信，服务器收到客户端请求后启动线程实现与客户端的交互。

3. 模拟单客户端和单服务器通过 Socket 频繁通信，要求如下：

1）分别模拟客户端和服务器端，完成客户端和服务器端的通信；

2）客户端和服务器端要发送的数据来自于键盘的输入；

3）接收到数据后打印数据内容；

提示 1：通过 Scanner 获取输入，把输入的数据写入输出流完成发送数据。

提示 2：可以尝试使用高级字符缓冲流：BufferReader、BufferWriter（不强制使用高级字符流，也可以使用字节流）

14 JDBC 数据库连接技术

1. 某花店需新进一批花，请帮助店主实现花的入库与展示操作，要求如下：

1）定义花类 Flower，属性包括：编号、花名、价格

2）定义操作数据库的类 DBOperate，要求至少提供 2 个方法，分别是花入库方法：addFlowerToDB(List<Flower> flowerList)和显示库中所有花的信息的方法：showFlowers()

3）定义测试类，测试 2）中方法的正确性

2.为便于实现数据库的操作，通常将数据库操作封装成一个工具类，请完成数据库操作工具类 DBUtil。

请结合上题中的花类，实现 DBUtil 工具类，并实现获取数据库连接 Connection getDBConnection(String url, String userName, String userPwd)、查询数据 ResultSet queryData(String sql)、更新数据 int updateData(String sql)、删除数据 int deleteData(String sql)、插入数据 int insertData(String sql)、关闭数据库连接 void closeDBConnection()的方法。

参考答案

1 熟悉 Java 程序基本语法

1.1

```
/**
 * 打印学生的姓名、学号、专业方向名称
 * @author  李玮玮
 * @date 创建时间：2018 年 5 月 3 日 下午 3:05:37
 */
public class Homework1_1 {

	public static void main(String[] args) {
		//实例化对象
		Homework1_1 homework1_1 = new Homework1_1();
		//调用方法实现打印功能
		homework1_1.printName("鹿晗");
		homework1_1.printNo("2016012001");
		homework1_1.printMajor("2016 级智能设备 1 班");
	}

	/**
	 * 打印学生的姓名
	 * @param name 姓名
	 */
	private void printName(String name) {
		System.out.println("姓名： " + name);
	}

	/**
	 * 打印学生的学号
	 * @param no 学号
	 */
	private void printNo(String no) {
		System.out.println("学号: " + no);
	}

	/**
	 * 打印学生的专业名称
	 * @param major 专业名称
```

```
     */
    private void printMajor(String major) {
        System.out.println("方向名称：" + major);
    }
}
```

1.2

```
/**
 * 算数运算
 * @author 李玮玮
 * @date 创建时间：2018 年 5 月 3 日 下午 3:13:56
 */
public class Homework1_2 {
    private double num1;
    private double num2;

    /**
     * 构造方法，初始化两个 double 类型的成员属性
     * @param num1
     * @param num2
     */
    public Homework1_2(double num1, double num2) {
        this.num1 = num1;
        this.num2 = num2;
    }

    /**
     * 实现算数运算的方法
     * @param operation 算数运算的类型
     * @return 算数运算的结果
     */
    public double arithmeticOperate(String operation) {
        double result = 0;
        switch (operation) {
        case "+":
            result = num1 + num2;
            break;
        case "-":
            result = num1 - num2;
            break;
        case "*":
            result = num1 * num2;
```

```
                break;
            case "/":
                //判断除数是否为 0
                if (num2 != 0) {
                    result = num1 / num2;
                } else {
                    System.out.println("除数不能为 0");
                }
                break;
            default:
                System.out.println("非法的运算符");
                break;
            }
            return result;
        }
}
```

1.3

```
/**
 * 测试算数运算类中的算数运算方法
 * @author  李玮玮
 * @date 创建时间：2018 年 5 月 3 日 下午 3:21:40
 */
public class Homework1_3 {
    public static void main(String[] args) {
        //测试算数运算类中的算数运算方法
        Homework1_2 arithmatic = new Homework1_2(10, 20);
        //测试加法运算
        System.out.println(arithmatic.arithmeticOperate("+"));
        //测试减法运算
        System.out.println(arithmatic.arithmeticOperate("-"));
        //测试乘法运算
        System.out.println(arithmatic.arithmeticOperate("*"));
        //测试除法运算
        System.out.println(arithmatic.arithmeticOperate("/"));
    }
}
```

2 程序执行流程

2.1

```
public class Homework2_1 {

	public static void main(String[] args) {
		int[] nums = {33,53,16,7,84,325,53,74,57,92};
		new Homework2_1().sort(nums);
		for (int n : nums) {
			System.out.print(n+",");
		}
	}
	/**
	 * 使用冒泡排序法实现对给定整形数组排序
	 * @param nums 待排序的数组
	 */
	private void sort(int[] nums) {
		for(int i = 0; i < nums.length; i++) {//外层循环变量控制排序的轮数，即排好序的元素个数-1
			for(int j = 0; j < nums.length-i-1; j++) {//内层循环变量控制同一轮中参与比较的两个数
				int temp = 0;
				if(nums[j] > nums[j+1]) {
					temp = nums[j];
					nums[j] = nums[j+1];
					nums[j+1] = temp;
				}
			}
		}
	}
}
```

2.2

```
public class Homework2_2 {

	public static void main(String[] args) {
		new Homework2_2().showMultiplicationTable();
	}

	/**
```

```
     * 实现倒着的乘法口诀表，即第 1 行是 9＊1 到 1＊1，最后一行是 9＊9
     */
    private void showMultiplicationTable() {
        for (int i = 1; i < 10; i++) {// 外层循环控制行数,即乘数
            for (int j = 9; j > i-1; j--) {// 内层循环控制被乘数
                System.out.print(j + "*" + i + "=" + j*i+"\t");
            }
            System.out.println();
        }
    }
}
```

2.3

```
public class Homework2_3 {

    public static void main(String[] args) {
        int pay = new Homework2_3().calculateWeeklyPay(46);
        System.out.println(pay);
    }

    /**
     * 计算并显示某人根据下面规则确定的周薪. 如果工作时间小于等于 40 小时，那么每小时 30 元；
     * 否则，该职员在 1200 元基础上，对超过 40 小时的部分，再加上 45 元每小时。
     *
     * @param workHours
     *                  工作时长
     * @return 周薪
     */
    private int calculateWeeklyPay(int workHours) {
        if (workHours > 0) {
            if (workHours <= 40) {
                return 30 * workHours;
            } else {
                return 1200 + 45 * (workHours - 40);
            }
        }
        return 0;
    }
}
```

3 类和对象

3.1 学生类定义

```
public class Student {
	private String sName;//学生姓名
	private String sNo;//学生学号
	private float englishScore;//英语成绩
	private float mathScore;//高数成绩
	private float physicalScore;//体育成绩

	//不含参数的构造方法
	public Student() {
		super();
	}
	//含参数的构造方法

	/**
	 * 姓名属性的访问器方法
	 * @return 姓名属性值
	 */
	public String getsName() {
		return sName;
	}

	public Student(String sName, String sNo, float englishScore, float mathScore, float physicalScore) {
		super();
		this.sName = sName;
		this.sNo = sNo;
		this.englishScore = englishScore;
		this.mathScore = mathScore;
		this.physicalScore = physicalScore;
	}

	/**
	 * 姓名属性的设置器方法
	 * @param sName 姓名
	 */
	public void setsName(String sName) {
		this.sName = sName;
	}
```

```
/**
 * 学号属性的访问器方法
 * @return 学号属性值
 */
public String getsNo() {
	return sNo;
}

/**
 * 学号属性的设置器方法
 * @param sNo 学号
 */
public void setsNo(String sNo) {
	this.sNo = sNo;
}

/**
 * 英语成绩属性的访问器方法
 * @return 英语成绩属性值
 */
public float getEnglishScore() {
	return englishScore;
}

/**
 * 英语成绩的设置器方法
 * @param englishScore 英语成绩
 */
public void setEnglishScore(float englishScore) {
	this.englishScore = englishScore;
}

/**
 * 数学成绩的访问器方法
 * @return 数学成绩的属性值
 */
public float getMathScore() {
	return mathScore;
}

/**
 * 数学成绩的设置器方法
 * @param mathScore 数学成绩
 */
```

```
    public void setMathScore(float mathScore) {
        this.mathScore = mathScore;
    }

    /**
     * 体育成绩的访问器方法
     * @return 体育成绩的属性值
     */
    public float getPhysicalScore() {
        return physicalScore;
    }

    /**
     * 体育成绩的设置器方法
     * @param physicalScore 体育成绩
     */
    public void setPhysicalScore(float physicalScore) {
        this.physicalScore = physicalScore;
    }

    /**
     * 重写的 toString()方法，返回学生对象的状态（各个属性的属性值）
     * @return 各个属性值
     */
    @Override
    public String toString() {
        return "Student [sName=" + sName + ", sNo=" + sNo + ", englishScore=" + englishScore +
", mathScore="
                + mathScore + ", physicalScore=" + physicalScore + "]";
    }

}
```

3.2 包含 main 方法的类

```
public class StudentStatistic {

    public static void main(String[] args) {
        StudentStatistic statistic = new StudentStatistic();
        // 学生数组初始化
        Student[] students = statistic.initStudentInfo();
        // 查询学生信息
        statistic.queryScoreByName("3", students);
```

```
        // 查询并统计不及格学生信息
        statistic.queryFailList(students);
    }

    /**
     * 初始化学生数组
     * @return 学生数组
     */
    private Student[] initStudentInfo() {
        // 创建包含 10 个学生的数组
        Student[] students = new Student[10];
        // 为数组元素赋值
        for (int i = 0; i < 10; i++) {
            Formatter formatter1 = new Formatter();// 用于浮点数的格式化
            Formatter formatter2 = new Formatter();// 用于浮点数的格式化
            Formatter formatter3 = new Formatter();// 用于浮点数的格式化
            Student student = new Student("学生" + (i + 1), 201601100 + i + 1 + "",
                    // Math.random()生成一个 0~1 之间的随机数，double 类型
                    // format("%.2f", value)
                    // 将 double 类型的 value 数据保留 2 位小数，
                    // 其返回值是 Formatter 对象，通过 toString 将其转换成字符串
                    // Float.parseFloat(value) 将字符串类型的 value 数据转换成浮点数
                    // *100 将 0~1 之间的保留 2 位小数的浮点数转换为 0~100 之间的数
                Float.parseFloat(formatter1.format("%.2f", Math.random()).toString()) * 100,
                Float.parseFloat(formatter2.format("%.2f", Math.random()).toString()) * 100,
                Float.parseFloat(formatter3.format("%.2f", Math.random()).toString()) * 100);
            students[i] = student;
        }
        return students;
    }
    /**
     * 按照姓名实现查询其成绩功能，名字中包含给定字符串的学生成绩均能被查询出来
     *
     * @param name
     *                待查询的学生姓名
     * @param students
     *                学生数组
     */
    private void queryScoreByName(String name, Student[] students) {
        for (Student stu : students) {
            String sName = stu.getsName();
            // 模糊查询，如果姓名中包含查询的字符串时，提取该学生的成绩信息
            if (sName.contains(name)) {
                System.out.println(sName + "的各科成绩：英语成绩 = " +
```

```
stu.getEnglishScore() + "; 数学成绩 = " + stu.getMathScore()
                                + "; 体育成绩 = " + stu.getPhysicalScore());
                }
            }
        }

        /**
         * 查询不及格名单及人数
         *
         * @param students
         *                  学生数组
         */
        private void queryFailList(Student[] students) {
            int n = 0;// 用于统计不及格人数
            System.out.println("不及格名单如下：");
            for (Student stu : students) {
                //提取学生信息
                String name = stu.getsName();
                float englishScore = stu.getEnglishScore();
                float mathScore = stu.getMathScore();
                float phiscalScore = stu.getPhysicalScore();

                if(englishScore < 60 || mathScore < 60 || phiscalScore < 60) {//存在不及格科目
                    n++;
                    System.out.print("\n" + name + " : ");
                }
                // 打印不及格课程的成绩
                if (englishScore < 60) {
                    System.out.print("英语成绩 = " + englishScore + "\t");
                } else if(mathScore < 60) {
                    System.out.print("数学成绩 = " + mathScore + "\t");
                } else if(phiscalScore < 60) {
                    System.out.print("体育成绩 = " + phiscalScore);
                }
            }
        }
}
```

4 封装和继承

4.1 汽车类

```
/**
```

```
 * 汽车类
 * 包括轮胎个数、汽车颜色、车身重量、时速成员变量
 * 提供至少 2 个构造方法
 * 汽车拥有的方法包括加速、停车
 * @author 李玮玮
 *
 */
public class Auto {

	int tyreNum;//轮胎个数
	String color;//汽车颜色
	double weight;//车身重量
	double speed;//时速

	public Auto() {
		this.tyreNum = 4;
		this.color = "white";
		this.weight = 1200;
		this.speed = 0;
	}
	Auto(String color, double weight, double speed){
		this.tyreNum = 4;
		this.color = color;
		this.weight = weight;
		this.speed = speed;
	}

	/**
	 * 随机加速方法
	 * @return 加速后的速度
	 */
	public double speedUp(){
		return Math.round(speed*(Math.random()+1));
	}

	/**
	 * 停车方法
	 */
	public void stopAuto(){
		this.speed = 0;
	}
	@Override
	public String toString() {
		return "Auto [tyreNum=" + tyreNum + ", color=" + color + ", weight=" + weight + ",
```

```
speed=" + speed + "]";
	}

}
```

4.2 小汽车类

```
/**
 * 小汽车 Car 类
 * 继承 Auto，添加空调、CD 成员变量，覆盖加速方法。
 * 提供 2 个构造方法，其中一个含参数的构造方法能够显示指定调用父类中某个含参数的构造方法。
 * public Car()
 * public Car(String color, double weight, double speed, String airCondition, String vehicleCD)
 *
 * @author liweiwei
 *
 */
public class Car extends Auto {
	String airCondition;//车载空调品牌
	String vehicleCD;//车载 CD 品牌

	public Car() {
		this.airCondition = "原装";
		this.vehicleCD = "雅马哈";
	}
	public Car(String color, double weight, double speed, String airCondition, String vehicleCD){
		super(color, weight, speed);
		this.airCondition = airCondition;
		this.vehicleCD = vehicleCD;
	}

	//复写加速的方法
	@Override
	public double speedUp() {
		return Math.round(1.5*this.speed);
	}

	@Override
	public String toString() {
		return "Car [tyreNum=" + tyreNum +", color="+ color + ", weight=" + weight +", speed="
+ speed +", airCondition=" + airCondition + ", vehicleCD=" + vehicleCD + "]";
	}
```

```
}
```

4.3 工厂类

```
public class Factory {
    public static void main(String[] args) {
        Auto auto = new Auto("Yello", 1250, 80);
        Car car = new Car("Black", 1100, 100, "改装","惠威");
        Scanner scanner = new Scanner(System.in);
        System.out.println("请输入赛车的时间(分钟>10)：如 30");
        int competitionTime = scanner.nextInt();
        scanner.close();
        competition(auto, car, competitionTime);
    }
    /**
     * 赛车方法
     * @param auto 汽车对象
     * @param car 小汽车对象
     * @param competition 赛车时间
     */
    public static void competition(Auto auto, Car car, int competition){
        //定义两个用于统计行程的变量
        double autoDistance = 0;
        double carDistance = 0;
        //定义两个用于记录加速后速度的变量
        double autoSpeedUp = auto.speedUp();
        double carSpeedUp = car.speedUp();
        //定义两车的加速行驶时间,取 10 分钟内的随机值
        double autoSpeedTime = Math.random()*10;
        double carSpeedTime = Math.random()*10;
        //计算辆车行驶的距离
        autoDistance = Math.round(auto.speed*(competition-autoSpeedTime) +
autoSpeedUp*autoSpeedTime);
        carDistance = Math.round(car.speed*(competition-carSpeedTime) +
carSpeedUp*carSpeedTime);
        //打印出最终赛车的赛况（包括名次、每辆车的信息及行驶车程）
        if(autoDistance > carDistance){
            System.out.println("Auto 汽车获胜");
            System.out.println("No1 Auto 行驶车程：" + autoDistance);
            System.out.println(auto.toString());
            System.out.println("No2 Car 行驶车程：" + carDistance);
```

```
                System.out.println(car.toString());
            } else {
                System.out.println("Car 汽车获胜");
                System.out.println("No1 Car       行驶车程：" + carDistance);
                System.out.println(car.toString());
                System.out.println("No2 Auto      行驶车程：" + autoDistance);
                System.out.println(auto.toString());
            }
        }
}
```

5 抽象类和接口

5.1~5.2

```
/**
 * 测试类
 * @author 李玮玮
 *
 */
public class ShapeTest {
    public static void main(String[] args) {
        System.out.println(new Circle(44).area());
    }
}
/**
 * 抽象父类
 * @author 李玮玮
 *
 */
abstract class Shape{
    public abstract double area();
    /**
     * 判断给定数字是否大于 0
     * @param num 待判定的数字
     * @return true 表示大于 0，false 表示不大于 0
     */
    boolean isAboveZero(double num){
        if(num > 0){
            return true;
        } else {
            return false;
        }
```

```
    }
}

//Circle 、Triangle、Rectangle
//if((a+b>c) && (b+c>a) && (a+c>b))
/**
 * 圆
 * @author 李玮玮
 *
 */
class Circle extends Shape{
    //定义属性
    private double r;//圆的半径

    //定义方法（构造方法和普通的成员方法）
    public Circle(double r) {
        if(isAboveZero(r)){//半径大于 0
            this.r = r;
        } else {//半径不大于 0
            System.out.println("半径应该是大于 0 的数字");;
        }
    }
    public double length(){
        return Math.round(2*Math.PI*r);
    }
    public double area() {
        return Math.round(Math.PI*r*r);
    }
    public double getR() {
        return r;
    }
    public void setR(double r) {
        this.r = r;
    }

    @Override
    public String toString() {
        return "Circle [r=" + r + "]";
    }
}
/**
 * 三角形类
 * @author 李玮玮
 *
```

```
 */
class Triangle extends Shape{

	private double a;
	private double b;
	private double c;

	//构造方法，判断三角形的三边是否都大于 0，并且任意两边和都大于第 3 边
	public Triangle(double a, double b, double c) {
		//参数符合规范
		if(isAboveZero(a) && isAboveZero(b) && isAboveZero(c)){
			if((a+b>c) && (b+c>a) && (a+c>b)){
				this.a = a;
				this.b = b;
				this.c = c;
			}
		} else {
			System.out.println("三角形的参数不符合规范");
		}
	}
	//计算周长
	public double length(){
		return Math.round(a+b+c);
	}
	//计算面积
	@Override
	public double area() {
		double l = (a+b+c)*0.5;//记录三角形的半周长
		return Math.sqrt(l*(l-a)*(l-b)*(l-c));
	}

	@Override
	public String toString() {
		return "Triangle [a=" + a + ", b=" + b + ", c=" + c + "]";
	}

}
//矩形类
class Rectangle extends Shape{

	private double a;//长
	private double b;//宽
	//构造方法
	public Rectangle(double a, double b) {
```

```
            if(isAboveZero(a) && isAboveZero(b)){//参数符合规范
                this.a = a;
                this.b = b;
            } else {
                System.out.println("参数不符合规范");
            }
        }

        //计算面积
        @Override
        public double area() {
            return Math.round(a*b);
        }
        //计算周长
        public double length(){
            return Math.round(2*(a+b));
        }
        @Override
        public String toString() {
            return "Rectangle [a=" + a + ", b=" + b + "]";
        }

}
```

5.3

```
//接口
public interface Point {
    double getDistance(Object obj);
}
//测试类
public class PointsTest {
    public static void main(String[] args) {
        DoubleDimensionPoint point1 = new DoubleDimensionPoint(10, 15);
        DoubleDimensionPoint point2 = new DoubleDimensionPoint(1, 7);
        System.out.println(point1.getDistance(point2));
    }
}
/**
 * 二维空间的点类
 * @author 李玮玮
 *
 */
```

```
class DoubleDimensionPoint implements Point{

	private double x;//点的横坐标
	private double y;//点的纵坐标



	public DoubleDimensionPoint(double x, double y) {
		this.x = x;
		this.y = y;
	}

	@Override
	public double getDistance(Object obj) {
		DoubleDimensionPoint point = (DoubleDimensionPoint) obj;
		double num1 = Math.pow(this.x-point.x, 2);
		double num2 = Math.pow(this.y-point.y, 2);
		return(Math.round(Math.sqrt(num1+num2)));
	}

}
class TripleDimensionPoint implements Point{
	private double x;//点的横坐标
	private double y;//点的纵坐标
	private double z;
	@Override
	public double getDistance(Object obj) {
		TripleDimensionPoint point = (TripleDimensionPoint) obj;
		double num1 = Math.pow(this.x-point.x, 2);
		double num2 = Math.pow(this.y-point.y, 2);
		double num3 = Math.pow(this.z-point.z, 2);
		return(Math.round(Math.sqrt(num1+num2+num3)));
	}

}
```

6 多态

6.1 动物类

```
/**
 * 动物抽象类
 * @author 李玮玮
```

```
 *
 */
public abstract class Animal {
	protected String name;
	//构造方法
	public Animal(String name) {
		this.name = name;
	}

	//eat 方法
	public void eat(Food food){
		System.out.println(this.name + "吃" + food.food());
	}
}

/**
 * 猫类,继承自动物抽象类
 * @author 李玮玮
 */
class Cat extends Animal{
	public Cat(String name) {
		super(name);
	}
}
/**
 * 狗类,继承自动物抽象类
 * @author 李玮玮
 */
class Dog extends Animal{
	public Dog(String name) {
		super(name);
	}
}
/**
 * 大象类，继承自动物抽象类
 * @author 李玮玮
 */
class Elephant extends Animal{
	public Elephant(String name) {
		super(name);
	}
}
```

6.2 食物类

```
/**
 * 食物抽象类
 * @author 李玮玮
 *
 */
public abstract class Food {
	protected String name;

	public Food(String name) {
		this.name = name;
	}
	public String food(){
		return this.name;
	}

}

//鱼类继承自食物类
class Fish extends Food{
	public Fish(String food) {
		super(food);
	}
}
//肉类继承自食物类
class Meat extends Food{
	public Meat(String food) {
		super(food);
	}
}
//香蕉类继承自食物类
class Banana extends Food{
	public Banana(String food) {
		super(food);
	}
}
```

6.3 饲养员类

```
/**
 * 饲养员类
 * @author 李玮玮
```

```
 *
 */
public class Person {

	public void feed(Animal animal, Food food){
		animal.eat(food);
	}
}
```

6.4 测试类

```
public class Test {
	public static void main(String[] args) {
		Person person = new Person();
		Cat cat = new Cat("猫");
		Fish fish = new Fish("鱼");
		person.feed(cat, fish);

		Animal dog = new Dog("狗");
		Food meat = new Meat("肉");
		person.feed(dog, meat);

		Animal elephant = new Elephant("大象");
		Food banana = new Banana("香蕉");
		person.feed(elephant, banana);

	}
}
```

7 枚举

7.1 学生类

```
public class Student {
	private String sName;// 学生姓名
	private float englishScore;// 英语成绩
	private float mathScore;// 数学成绩
	private float physicalScore;// 体育成绩
	private Grade grade;

	// 不含参数的构造方法
	public Student() {
```

```
        super();
    }

    // 含参数的构造方法
    public Student(String sName, float englishScore, float mathScore, float physicalScore) {
        super();
        this.sName = sName;
        this.englishScore = englishScore;
        this.mathScore = mathScore;
        this.physicalScore = physicalScore;
        setGrade();//确定该学生等级
    }

    /**
     * 姓名属性的访问器方法
     *
     * @return 姓名属性值
     */
    public String getsName() {
        return sName;
    }

    /**
     * 姓名属性的设置器方法
     *
     * @param sName
     *                  姓名
     */
    public void setsName(String sName) {
        this.sName = sName;
    }

    /**
     * 英语成绩属性的访问器方法
     *
     * @return 英语成绩属性值
     */
    public float getEnglishScore() {
        return englishScore;
    }

    /**
     * 英语成绩的设置器方法
     *
```

```
     * @param englishScore
     *                  英语成绩
     */
    public void setEnglishScore(float englishScore) {
        this.englishScore = englishScore;
    }

    /**
     * 数学成绩的访问器方法
     *
     * @return 数学成绩的属性值
     */
    public float getMathScore() {
        return mathScore;
    }

    /**
     * 数学成绩的设置器方法
     *
     * @param mathScore
     *                  数学成绩
     */
    public void setMathScore(float mathScore) {
        this.mathScore = mathScore;
    }

    /**
     * 体育成绩的访问器方法
     *
     * @return 体育成绩的属性值
     */
    public float getPhysicalScore() {
        return physicalScore;
    }

    /**
     * 体育成绩的设置器方法
     *
     * @param physicalScore
     *                  体育成绩
     */
    public void setPhysicalScore(float physicalScore) {
        this.physicalScore = physicalScore;
    }
```

```
    /**
     * 等级属性的访问器方法
     *
     * @return
     */
    public Grade getGrade() {
        return grade;
    }

    /**
     * 等级属性的设置器方法，根据英语、数学、体育三科成绩确定等级
     */
    public void setGrade() {
        // 计算三科成绩的平均分
        float avarageScore = (englishScore + mathScore + physicalScore) / 3;
        if(avarageScore > 90) {
            this.grade = Grade.A;
        } else if(avarageScore > 80) {
            this.grade = Grade.B;
        } else if(avarageScore > 60) {
            this.grade = Grade.C;
        } else {
            this.grade = Grade.D;
        }
    }

    /**
     * 重写的 toString()方法，返回学生对象的状态（各个属性的属性值）
     *
     * @return 各个属性值
     */
    @Override
    public String toString() {
        return "Student [sName=" + sName + ", englishScore=" + englishScore + ", mathScore="
+ mathScore
                + ", physicalScore=" + physicalScore + ", grade=" + grade + "]";
    }

}
```

7.2 枚举定义

```
public enum Grade {
    A, B, C, D;
}
```

7.3 测试类 统计等级比例

```
/**
 * 统计学生等级比例
 *
 * @author 李玮玮
 * @date 创建时间：2018 年 5 月 3 日 下午 4:20:49
 */
public class Test {

    public static void main(String[] args) {
        Test test = new Test();
        // 学生数组初始化
        Student[] students = test.initStudentInfo();
        // 统计学生等级比例
        test.statistic(students);
    }

    /**
     * 初始化学生数组
     *
     * @return 学生数组
     */
    private Student[] initStudentInfo() {
        // 创建包含 10 个学生的数组
        Student[] students = new Student[10];
        // 为数组元素赋值
        for (int i = 0; i < 10; i++) {
            Formatter formatter1 = new Formatter();// 用于浮点数的格式化
            Formatter formatter2 = new Formatter();// 用于浮点数的格式化
            Formatter formatter3 = new Formatter();// 用于浮点数的格式化
            Student student = new Student("学生" + (i + 1),
                    // Math.random()生成一个 0~1 之间的随机数，double 类型
                    // format("%.2f", value)
                    // 将 double 类型的 value 数据保留 2 位小数，其返回值是 Formatter
对象，通过 toString 将其转换成字符串
                    // Float.parseFloat(value) 将字符串类型的 value 数据转换成浮点数
```

```
                        // *100 将 0~1 之间的保留 2 位小数的浮点数转换为 0~100 之间的数
                        Float.parseFloat(formatter1.format("%.2f", Math.random()).toString()) *
100,
                        Float.parseFloat(formatter2.format("%.2f", Math.random()).toString()) *
100,
                        Float.parseFloat(formatter3.format("%.2f", Math.random()).toString()) *
100);
                students[i] = student;
            }
            return students;
        }

        /**
         * 统计各个等级学生比例
         *
         * @param students
         *                  学生数组
         */
        private void statistic(Student[] students) {
            int nA = 0;// A 级学生人数
            int nB = 0;// B 级学生人数
            int nC = 0;// C 级学生人数
            int nD = 0;// D 级学生人数
            for (Student stu : students) {
                switch (stu.getGrade()) {
                case A:
                    nA++;
                    break;
                case B:
                    nB++;
                    break;
                case C:
                    nC++;
                    break;
                case D:
                    nD++;
                    break;
                }
            }
            System.out.println("A 级学生人数：" + nA + ";B 级学生人数：" + nB + ";C 级学生人
数：" + nC + ";D 级学生人数：" + nD);
            // 计算等级比例
            System.out.println("A 级学生比例：" + nA * 100 / students.length + "%");
            System.out.println("B 级学生比例：" + nB * 100 / students.length + "%");
```

```
            System.out.println("C 级学生比例：" + nC * 100 / students.length + "%");
            System.out.println("D 级学生比例：" + nD * 100 / students.length + "%");
        }

}
```

8 异常处理.1

8.1 大象类

```
/**
 * 大象类
 * @author 李玮玮
 *
 */
public class Elephant {
    private int length;
    private int width;
    private int height;

    public Elephant(int length, int width, int height) {
        this.length = length;
        this.width = width;
        this.height = height;
    }

    public int getLength() {
        return length;
    }

    public int getWidth() {
        return width;
    }

    public int getHeight() {
        return height;
    }

}
```

8.2 冰箱类

```
/**
 * 冰箱类
 * 包含 3 个属性：长、宽、高
 * 包含 2 个方法：求体积，装大象
 * @author 李玮玮
 *
 */
public class Fridge {
	private int length;
	private int width;
	private int height;

	public Fridge(int length, int width, int height) {
		this.length = length;
		this.width = width;
		this.height = height;
	}

	//求体积的方法
	public int volume(){
		return this.length*this.width*this.height;
	}

	//装大象的方法
	public void store(Elephant elephant) throws OutOfVolumeException{
		//判断是否能装下大象,如果装不下，则抛出一个 Exception 异常
		if(this.length < elephant.getLength() || this.width < elephant.getWidth()
				|| this.height < elephant.getHeight()){
			//System.out.println("冰箱装不下这只大象");
			throw new OutOfVolumeException();
		} else {
			System.out.println("大象已经装入冰箱");
		}
	}
}
```

8.3 异常类

```
public class OutOfVolumeException extends Exception {
	private String MESSAGE = "异常：大象体积过大";
	@Override
```

```
    public String getMessage() {
        // TODO Auto-generated method stub
        return MESSAGE;
    }

    @Override
    public String toString() {
        // TODO Auto-generated method stub
        return MESSAGE + "," + this.getClass().getName();
    }

    @Override
    public void printStackTrace() {
        // TODO Auto-generated method stub
        System.out.println(MESSAGE + "," + this.getClass().getName());
    }
}
```

8.4 装大象

```
public class StuffElephant {
    public static void main(String[] args) {
        //  定义大象和冰箱类
        Elephant elephant1 = new Elephant(20, 10, 30);
        Elephant elephant2 = new Elephant(40, 20, 60);
        Fridge fridge = new Fridge(30, 30, 30);
        //模拟用冰箱装大象
        try {
            fridge.store(elephant1);
            fridge.store(elephant2);
        } catch (OutOfVolumeException error){
            error.printStackTrace();
        } catch (Exception e) {
            System.out.println("大象太大，冰箱装不下");
        } finally {
            System.out.println("大象已经移出冰箱");
        }
    }
}
```

9 容器

9.1 子弹类

```
/**
 * 1．定义子弹类：Ball,有属性 编号，型号。
 * 默认构造方法，设编号为 1， 型号为“M54”。
 * 带参数构造方法：设定编号和型号。
 * 方法 1：取得编号
 * 方法 2：取得型号。
 * 方法 3：显示子弹的编号和型号： 编号-型号。
 * @author 李玮玮
 *
 */
public class Ball {
	private int ballNo;//编号
	private String model;//型号

	//默认构造方法，设编号为 1， 型号为“M54”
	public Ball() {
		this.ballNo = 1;
		this.model = "M54";
	}
	//带参数构造方法：设定编号和型号
	public Ball(int ballNo, String model) {
		this.ballNo = ballNo;
		this.model = model;
	}
	//方法 1：取得编号
	public int getBallNo() {
		return ballNo;
	}
	//法 2：取得型号
	public String getModel() {
		return model;
	}
	//方法 3：显示子弹的编号和型号： 编号-型号
	public void showInfo(){
		System.out.println(this.ballNo + " - " + this.model);
	}
}
```

9.2 枪类

```
/**
 * 2.定义枪类：Gun
 * 有属性：编号，型号，弹夹，装弹数量。
 * 默认构造方法：创建编号为 1，型号为“M54”，装弹数量为 6。
 * 带参数构造方法：与属性相同的参数，设定属性：编号，型号，数量的值。
 * 装弹方法：将一个子弹对象装入弹夹，如果弹夹已满，抛出弹夹满异常。
 * 射击方法：返回一个子弹对象，减少弹夹的一颗子弹，如果弹夹空，则抛出空异常。
 * 显示弹夹内子弹列表信息方法：显示弹夹内的子弹信息：编号-型号
 * 取得当前弹夹子弹数的方法。
 * @author 李玮玮
 *
 */
public class Gun {
	private int gunNo;//编号
	private String model;//型号
	private List<Ball> clip;//弹夹
	private int ballCount;//装弹数量

	//默认构造方法：创建编号为 1，型号为“M54”，装弹数量为 6
	public Gun() {
		this.gunNo = 1;
		this.model = "M54";
		this.ballCount = 6;
		clip = new ArrayList<Ball>();
	}
	//带参数构造方法：与属性相同的参数，设定属性：编号，型号，数量的值

	public Gun(int gunNo, String model, int ballCount) {
		this.gunNo = gunNo;
		this.model = model;
		this.ballCount = ballCount;
		clip = new ArrayList<Ball>();
	}
	//装弹方法：将一个子弹对象装入弹夹，如果弹夹已满，抛出弹夹满异常
	public void addBall(Ball ball) throws Exception{
		if(clip.size() == ballCount){//弹夹已满
			System.out.println("弹夹已满");
			throw new Exception();
		}else{
			clip.add(ball);
			System.out.println("装入一个子弹");
```

```
        }
    }
    //射击方法：返回一个子弹对象，减少弹夹的一颗子弹，如果弹夹空，则抛出空异常
    public Ball shoot() throws Exception{
        Ball ball = null;
        if(clip.size() < 1){//弹夹空
            System.out.println("弹夹空");
            throw new Exception();
        }else{
            ball = clip.get(clip.size()-1);
            clip.remove(clip.size()-1);
            System.out.println("射击");
        }
        return ball;
    }
    //显示弹夹内子弹列表信息方法：显示弹夹内的子弹信息：编号-型号
    public void showClipInfo(){
        for (Ball ball : clip) {
            ball.showInfo();
        }
    }
    // 取得当前弹夹子弹数的方法。
    public int getCurrentBallCount(){
        return clip.size();
    }
}
```

9.3 测试类

```
/**
 * 3.编写带 main 方法的测试类，模拟枪的装弹和射击功能。
 * @author 李玮玮
 *
 */
public class ShootTest {

    public static void main(String[] args) throws Exception {
        //创建枪对象
        Gun gun = new Gun();
        //创建若干个子弹对象
        Ball ball1 = new Ball();
        Ball ball2 = new Ball();
        Ball ball3 = new Ball();
```

```
            Ball ball4 = new Ball();
            Ball ball5 = new Ball();
            Ball ball6 = new Ball();
            //模拟装弹
            gun.addBall(ball1);
            gun.addBall(ball2);
            gun.addBall(ball3);
            gun.addBall(ball4);
            gun.addBall(ball5);
            gun.addBall(ball6);
            //模拟射击
            gun.shoot();
            System.out.println(gun.getCurrentBallCount());//显示剩余子弹数量
            gun.showClipInfo();//显示剩余子弹信息
            gun.shoot();
      }
}
```

10 流和文件

10.1

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

/**
 * @author  李玮玮
 * @date 创建时间：2018 年 5 月 24 日 下午 2:44:19
 */
public class Homework10_1 {

      public static void main(String[] args) {
            new Homework10_1().copyImgFile();
      }
      //拷贝文件的方法
      public void copyImgFile() {
            try {
                  //创建文件输入流，用于从本地读取图片文件
                  InputStream in = new FileInputStream("E:/img.jpg");
```

```
                //创建文件输出流，用于拷贝从输入流读取的图片文件，从而实现转存
                OutputStream out = new FileOutputStream("E:imgCopy.jpg");
                //循环读写文件
                int n = 0;
                while((n = in.read()) != -1) {
                        out.write(n);
                        out.flush();
                }
                System.out.println("文件拷贝成功");
                //文件读写完毕，关闭流
                in.close();
                out.close();
            } catch (FileNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
}
```

10.2

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;

/**
 * @author  李玮玮
 * @date 创建时间：2018 年 5 月 24 日 下午 2:52:11
 */
public class Homework10_2 {
    public static void main(String[] args) {
        Homework10_2 homework = new Homework10_2();
        //使用第 1 种方法拷贝文件
```

```
//          homework.copyFileMethod1();
            //使用第 2 种方法拷贝文件
            homework.copyFileMethod2();
        }
        //第 1 种方法，拷贝后的文件是乱码
        public void copyFileMethod1() {
            try {
                //创建 BufferedReader 对象
                BufferedReader reader = new BufferedReader(new FileReader("E:/test.txt"));
                //创建 BufferedWriter 对象
                BufferedWriter writer = new BufferedWriter(new FileWriter("E:/testCopy.txt",
true));
                //循环读写文件
                String str = "";
                while((str = reader.readLine()) != null) {
                    writer.write(str);
                    writer.flush();
                }
                //打印空行，方便下次追加
                writer.newLine();
                System.out.println("文件拷贝成功");
                //关闭流
                reader.close();
                writer.close();
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
        //第 2 种方法，通过指定字符编码方式，解决拷贝后的文件是乱码的问题
        public void copyFileMethod2() {
            try {
                //创建 BufferedReader 对象
                BufferedReader reader = new BufferedReader(new InputStreamReader(new
FileInputStream("E:/test.txt"), "utf-8"));
                //创建 BufferedWriter 对象
                BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("E:/testCopy.txt", true),"utf-8"));
                //循环读写文件
                String str = "";
                while((str = reader.readLine()) != null) {
                    writer.write(str);
```

```
                writer.flush();
            }
            //打印空行，方便下次追加
            writer.newLine();
            System.out.println("文件拷贝成功");
            //关闭流
            reader.close();
            writer.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

10.3 学生类

```
import java.io.Serializable;

/**
 * @author  李玮玮
 * @date 创建时间：2018 年 5 月 24 日 下午 3:10:48
 */
public class Student implements Serializable{
    //定义属性
    private String name;     //姓名属性
    private int age;         //年龄属性
    private float score; //成绩属性

    //定义属性对应的访问器和设置器方法
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
```

```
	}
	public float getScore() {
		return score;
	}
	public void setScore(float score) {
		this.score = score;
	}

	//重写 toString()方法
	@Override
	public String toString() {
		return "Student [name=" + name + ", age=" + age + ", score=" + score + "]";
	}
}
```

10.4 测试类

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

/**
 * @author 李玮玮
 * @date 创建时间：2018 年 5 月 24 日 下午 3:10:10
 */
public class Homework10_3 {
	// 定义学生集合
	private List<Student> stus;

	public Homework10_3() {
		// 初始化学生集合对象
		stus = new ArrayList<>();
	}

	public List<Student> getStus() {
		return stus;
	}
```

```
public static void main(String[] args) {
    // 初始化 Homework10_3 对象
    Homework10_3 work = new Homework10_3();
    //初始化学生信息
    work.initStudents();
    try {
        //序列化学生信息
        work.storeStus("E:/stuInfo.txt", work.getStus());
        //反序列化学生信息
        List<Student> list = work.readStusFromFile("E:/stuInfo.txt");
        //显示反序列化的学生信息
        System.out.println(list);
    } catch (FileNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

// 初始化学生对象
public void initStudents() {
    // 通过控制台输入，初始化 3 个学生对象
    // 1. 初始化用于接收控制台输入的 Scanner 对象
    Scanner scanner = new Scanner(System.in);
    for (int i = 0; i < 3; i++) {
        System.out.println("请输入第" + (i + 1) + "个学生的姓名：");
        String name = scanner.next();
        System.out.println("请输入第" + (i + 1) + "个学生的年龄：");
        int age = scanner.nextInt();
        System.out.println("请输入第" + (i + 1) + "个学生的成绩：");
        float score = scanner.nextFloat();
        // 2. 根据用户的输入创建学生对象
        Student stu = new Student();
        stu.setName(name);
        stu.setAge(age);
        stu.setScore(score);
        // 3. 将创建好的学生对象添加到学生集合列表中
        stus.add(stu);
```

```
            }
            //4. 关闭 Scanner 对象流
            scanner.close();
        }
        /**
         * 序列化方法
         * @param filePath 序列化的文件名称
         * @param list 待存储的集合数据
         * @throws FileNotFoundException 可能抛出的序列化的文件未找到异常
         * @throws IOException  可能抛出的 IO 异常
         */
        public void storeStus(String filePath, List<Student> list) throws FileNotFoundException, IOException {
            //创建序列化对象
            ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(filePath));
            //存储学生数据
            out.writeObject(list);
            out.flush();
            //序列化完成，关闭序列化对象流
            out.close();
            System.out.println("序列化操作完成，学生信息已保存");
        }
        /**
         * 反序列化方法
         * @param filePath 反序列化读取的文件
         * @return 反序列化得到的学生数据
         * @throws FileNotFoundException 可能抛出的反序列化文件不存在异常
         * @throws IOException 可能抛出的 IO 异常
         * @throws ClassNotFoundException 可能抛出的类型转换异常
         */
        public List<Student> readStusFromFile(String filePath) throws FileNotFoundException, IOException, ClassNotFoundException{
            //创建反序列化对象
            ObjectInputStream in = new ObjectInputStream(new FileInputStream(filePath));
            //读取学生信息
            List<Student> list = (List<Student>) in.readObject();
            //反序列化完成，关闭反序列化对象流
            in.close();
            return list;
        }
}
```

11 字符串、日期格式化

11.1 有两种方法实现

```
import java.time.LocalDate;
import java.time.MonthDay;
import java.time.Period;
import java.time.temporal.ChronoUnit;
import java.util.Calendar;
import java.util.Date;

import javax.naming.BinaryRefAddr;

/**
 * 选择合适的日期类型表示现在的时间，并计算现在距离自己的生日还有多少天
 *
 * @author 李玮玮
 * @date 创建时间：2018 年 5 月 30 日 上午 9:52:38
 */
public class Homework11_1 {

	public static void main(String[] args) {
		//方法 1
		Calendar calendar = Calendar.getInstance();
		//设置生日 Calendar.DAY_OF_MONTH 和 Calendar.DATE 是等价的
		//假设出生日期是 1994-7-15(月份从 0 开始）
		calendar.set(Calendar.YEAR, 1994);
		calendar.set(Calendar.MONTH, 6);
		calendar.set(Calendar.DATE, 15);
		//转换成 Date
		Date date = calendar.getTime();
		//计算距离生日的天数
		DaysToBirthday(date);

		//方法 2
		//LocalDate 中的月份从 1 开始
		LocalDate birth = LocalDate.of(1994, 6, 1);
		DaysToBirthday(birth);
	}

	/**
	 * 方法 1： 通过传统日期形式获取距离生日的天数
	 *
```

```
     * @param birthDay  出生日期
     * @return
     */
    public static int DaysToBirthday(Date birthDay) {
        int days = -1;//用于记录距离生日的天数
        //创建 Calendar 的对象
        Calendar calendar = Calendar.getInstance();
        System.out.println("----"+calendar.getTime());
        //获取当前月份和日信息
        int monthNow = calendar.get(Calendar.MONTH);
        int dayNow = calendar.get(Calendar.DAY_OF_MONTH);
        //设置生日给 Calender 对象
        calendar.setTime(birthDay);
        System.out.println("----"+calendar.getTime());
        //获取生日的月份和日信息,月份从 0 开始
        int monthBirth =    calendar.get(Calendar.MONTH);
        int dayBirth = calendar.get(Calendar.DAY_OF_MONTH);
        //定义 2018 每月的天数数组
        int[] monthDays = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
        if(monthNow > monthBirth || (monthNow == monthBirth && dayNow > dayBirth)) {
            System.out.println("今年生日已过，期待明年生日吧");
            return days;
        } else if(monthNow == monthBirth && dayNow <= dayBirth){
            days = dayBirth - dayNow;
        } else {
            days = 0;
            for(int i = monthNow+1; i < monthBirth; i++) {
                days += monthDays[i];
            }
            days = days + monthDays[monthNow] - dayNow + dayBirth;
        }
        System.out.print("距离生日还有"+days+"天");
        return days;

    }

    /**
     *  方法 2：  通过 JDK8 中的日期特性获取距离生日的天数
     * @param birthDay  出生日期
     * @return
     */
    public static int DaysToBirthday(LocalDate birthDay) {
        //获取当前日期
        LocalDate now = LocalDate.now();
```

```
            //设置今年过生日时的日期
            LocalDate birth = birthDay.of(now.getYear(), birthDay.getMonth(),
birthDay.getDayOfMonth());
            long days = ChronoUnit.DAYS.between(now, birth);
            if(days < 0) {
                System.out.println("今年生日已过，期待明年生日吧");
            } else {
                System.out.print("距离生日还有"+days+"天");
            }
            return 0;

        }
}
```

11.2

```
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

/**
 * @author  李玮玮
 * @date 创建时间：2018 年 5 月 30 日 下午 3:19:22
 */
public class Homework11_2 {

    public static void main(String[] args) {
        Calendar calendar = Calendar.getInstance();
        //设置生日
        calendar.set(Calendar.YEAR, 1994);
        calendar.set(Calendar.MONTH, 7);
        calendar.set(Calendar.DAY_OF_MONTH, 1);
        //转换成 Date 类型
        Date birthday = calendar.getTime();
        //打印生日
        printDate(birthday);
    }
    /**
     * 打印自己出生日期的两种不同格式的形式
     * @param birthday 生日
     */
    public static void printDate(Date birthday) {
        //设置两种日期格式
```

```
        SimpleDateFormat simple1 = new SimpleDateFormat("yyyy-MM-dd");// HH:mm:ss
        SimpleDateFormat simple2 = new SimpleDateFormat("yyyy 年 M 月 d 日");// H 点 m 分
s 秒

        String today1 = simple1.format(birthday);
        String today2 = simple2.format(birthday);
        System.out.println("生日是：" + today1);
        System.out.println("生日是：" + today2);
    }
}
```

11.3

```
/**
 * @author  李玮玮
 * @date 创建时间：2018 年 5 月 30 日 下午 3:34:25
 */
public class Homework11_3 {

    public static void main(String[] args) {
        String str = "I Love to Learn Java,Do you Love Too?";
        analyzeString(str);
    }

    public static void analyzeString(String str) {
        //将字符串中的“,”改成“ ”
        String strNew = str.replace(",", " ");
        //获取单词个数
        String[] words = strNew.split(" ");
        int wordsNum = words.length;
        System.out.println(str + "中单词个数为" + wordsNum);
        int oNum = 0;//用于统计字符‘o’的个数
        for(String s : words) {
            //判断当前单词中是否包含字符‘o’
            if(s.contains("o")) {//计算当前单词中字符 o 的个数
                String[] subStr = s.split("o");
                oNum += subStr.length;
                if(!s.startsWith("o") && !s.endsWith("o")) {
                    oNum--;
                }
            }
        }
        System.out.println("字符 o 的出现次数为：" + oNum);
```

```
    }
}
```

12 线程和多线程

12.1 车间类的定义

```
/**
 * @author  李玮玮
 * @date 创建时间：2018 年 6 月 5 日 上午 9:18:12
 */
public class Workshop {
    private String name;

    public Workshop(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return "Workshop [name=" + name + "]";
    }

}
```

12.2 工厂类

```
/**
 * @author  李玮玮
 * @date 创建时间：2018 年 6 月 5 日 上午 9:19:24
 */

import java.util.ArrayList;
import java.util.List;
```

```
public class Factory {
	private int capacity; //容量
	private List<Workshop> workshops; //车间集合

	public Factory(int capacity) {
		this.capacity = capacity;
		workshops = new ArrayList<>();
	}

	/**
	 * 加工玩具的方法(product)，方法的功能是各个车间同时工作。
	 * 根据仓库容量平均分配给各个车间需要加工玩具的数量
	 */
	public void product() {
		//计算每个工厂加工玩具的数量
		int total = Math.abs(capacity / workshops.size());
		//创建线程让每个工厂开始制作玩具
		for(Workshop shop : workshops) {
			ProductToysThread thread = new ProductToysThread(total, shop.getName());
			thread.start();
		}
	}
	/**
	 * 建造玩具车间的方法(createWorkshop)，方法功能是模拟建造玩具生成车间，
	 * 即向集合属性中添加一个新车间
	 * @param workshopName 车间名称
	 */
	public void createWorkshop(String workshopName) {
		Workshop shopName = new Workshop(workshopName);
		workshops.add(shopName);
		System.out.println(workshopName+"车间建造完成");
	}
}
```

12.3 线程类

```
/**
 * @author  李玮玮
 * @date 创建时间：2018 年 6 月 5 日 上午 9:37:20
 */
public class ProductToysThread extends Thread{
	private int total; // 需要加工玩具的数量
```

```
    private String name; //线程名称
    public ProductToysThread(int total, String name) {
        this.total = total;
        this.name = name;
    }

    @Override
    public void run() {
        //设置线程名称
        this.setName(name);
        //模拟加工玩具
        for(int i = 1; i <= total; i++) {
            //模拟每隔 2 秒加工一个玩具并打印当前线程已加工玩具数量
            System.out.println("车间"+ this.getName()+"已经加工玩具数量： " + i);
            try {
                //间隔时间 2s
                this.sleep(2000);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
        System.out.println(this.getName()+"加工玩具完毕");
    }
}
```

12.4 测试类

```
/**
 * @author  李玮玮
 * @date 创建时间：2018 年 6 月 5 日 上午 9:45:46
 */
public class Test {

    public static void main(String[] args) {
        //创建工厂类的对象
        Factory factory = new Factory(50);
        //建造车间
        factory.createWorkshop("BearWorkshop");
        factory.createWorkshop("DogWorkshop");
        factory.createWorkshop("MonkeyWorkshop");
        factory.createWorkshop("CatWorkshop");
        factory.createWorkshop("PigWorkshop");
```

```
            //开始制作玩具
            factory.product();
        }
}
```

13 网络编程

13.1

```
public class DownloadImg extends Thread{

    private String imgUrl;
    private String savePath;

    public DownloadImg(String imgUrl, String savePath) {
        this.imgUrl = imgUrl;
        this.savePath = savePath;
    }

    @Override
    public void run() {
        try {
            URL url = new URL(imgUrl);
            HttpURLConnection connection = (HttpURLConnection) url.openConnection();
            InputStream in = connection.getInputStream();
            OutputStream out = new FileOutputStream(savePath);
            int b = -1;
            while((in.read()) != -1) {
                out.write(b);
                out.flush();
            }
            in.close();
            out.close();
            System.out.println("图片下载完成");
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }
}

public class Test {
```

```
    public static void main(String[] args) {
        String img1 = "http://pic15.nipic.com/20110813/1993003_205156492136_2.jpg";
        String img2 = "http://pic31.nipic.com/20130722/12473387_075510736124_2.jpg";
        String savePath1 = "E:/flower1.jpg";
        String savePath2 = "E:/flower2.jpg";
        DownloadImg down1 = new DownloadImg(img1, savePath1);
        DownloadImg down2 = new DownloadImg(img2, savePath2);
        down1.start();
        down2.start();
    }

}
```

13.2

客户端

```
/**
 * 模拟多客户端与服务端交互，客户端功能相同
 * 本客户端简单模拟一次收发操作，
 * 即向服务端请求，并接收响应
 * @author 李玮玮
 *
 */
public class ClientTest {

    public static void main(String[] args) {
        try {
            //创建 Socket 客户端对象
            Socket client = new Socket("127.0.0.1", 8888);
            //获取输出流，用于向服务器提交请求
            OutputStream out = client.getOutputStream();
            //狗造高级流
        BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(out, "utf-8"));
            //模拟向服务器发送一条数据
            writer.write("你好服务器，客户端前来报到");
            writer.newLine();
            writer.flush();

            //获取输入流，用于接收服务器返回的消息
            InputStream in = client.getInputStream();
```

```
            //构造高级流
        BufferedReader reader = new BufferedReader(new InputStreamReader(in, "utf-8"));
            //接收服务器的返回数据
            String responseStr = reader.readLine();
            System.out.println("收到服务端返回数据：" + responseStr);
            //关闭流
            in.close();
            out.close();
            reader.close();
            writer.close();
            client.close();

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

}
```

服务端

```
/**
 * 模拟多客户端与服务端交互
 * 本服务端模拟可以重复接收客户端请求
 * @author 李玮玮
 *
 */
public class ServerTest {

    public static void main(String[] args) {
        try {
            // 创建服务端 ServerSocket 对象
            ServerSocket server = new ServerSocket(8888);
            while (true) {
                // 开始监听是否有客户端连接
                System.out.println("等待客户端连接");
                Socket client = server.accept();
                System.out.println("收到客户端连接请求");
                // 启动线程处理客户端请求
                ClientOperation operation = new ClientOperation(client);
                operation.start();
            }
        } catch (IOException e) {
```

```
                e.printStackTrace();
            }
        }

}

class ClientOperation extends Thread {
    private Socket client;

    public ClientOperation(Socket client) {
        this.client = client;
    }

    @Override
    public void run() {
        try {
            // 获取输入流，用于接收客户端请求
            InputStream in = client.getInputStream();
            // 构造字符缓冲流
        BufferedReader reader = new BufferedReader(new InputStreamReader(in, "utf-8"));
            //读取用户请求数据
            String requestStr = reader.readLine();
            System.out.println("收到客户端请求数据：" + requestStr);

            // 获取输出流，用于给客户端返回数据
            OutputStream out = client.getOutputStream();
        BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(out, "utf-8"));
            //向客户端返回数据
            writer.write("你好客户端，信息已收到");
            writer.newLine();
            writer.flush();
            //关闭
            in.close();
            out.close();
            reader.close();
            writer.close();
            client.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

12.3

服务端

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;

/**
 * 模拟服务端功能 服务端首先接收客户端提交的数据， 然后通过读取控制台输入信息，
 * 向客户端返回数据 当控制台输入 exit 时，表示本次输入信息结束
 *
 * 当服务端本次输入结束后，要告知客户端传输信息完成， 向客户端发送“end”字符串
 *
 * @author 李玮玮
 *
 */
public class ServerTest {

    public static void main(String[] args) {

        Socket client = null;
        InputStream in = null;
        OutputStream out = null;
        BufferedWriter writer = null;
        BufferedReader reader = null;
        Scanner scanner = null;

        // 用于标识是否继续接收客户端的请求，默认为接收(flase:不提交；true:提交)
        boolean isContinue = true;
        try {
            // 创建服务端 Socket 对象
            ServerSocket server;
            server = new ServerSocket(8888);
            System.out.println("服务器开始等待客户端连接");
```

```
                // 得到客户端 Socket 对象
                client = server.accept();
                System.out.println("收到客户端连接请求");
                // 获取输入流对象用于接收客户端发送的数据
                in = client.getInputStream();
                // 构造输入缓冲流
                reader = new BufferedReader(new InputStreamReader(in, "UTF-8"));
                // 获取输出流对象，用于向服务端发送数据
                out = client.getOutputStream();
                // 构造输出缓冲流
                writer = new BufferedWriter(new OutputStreamWriter(out, "UTF-8"));
                scanner = new Scanner(System.in);
            } catch (IOException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
            }
            while (isContinue) {
                try {
                    // 创建用于接收客户端数据的临时字符串变量
                    String strTemp = null;
                    // 每次接收服务端的一行数据，当接收到的数据为“end”时，表示服务器返回数据传输结束
                    System.out.println("收到来自客户端的数据：");
                    while (!"end".equals(strTemp = reader.readLine())) {
                        System.out.println(strTemp);
                    }

                    // 创建接收控制台输入时保存字符串的临时变量
                    String str = null;
                    // 每次读取控制台一行字符，并保存在 str 临时变量中,判定是否结束输入
                    System.out.println("请在此输入向客户端返回的数据");
                    while (!"exit".equals(str = scanner.nextLine())) {
                        // 将读取的一行字符发送给服务器
                        writer.write(str);
                        // 由于缓冲流不会读取控制台的换行符，因此每次发送一行数据后还需再发送一个空行，表示换行
                        writer.newLine();
                        writer.flush();
                    }
                    // 控制台输入结束，发送 end 告知服务器数据传输结束
                    writer.write("end");
                    writer.newLine();
                    writer.flush();
```

```
                    System.out.println("向客户端返回数据完成");

                    // 判断是否客户端还有请求
                    strTemp = reader.readLine();
                    if ("Y".equals(strTemp)) {// 还有新的请求
                        isContinue = true;
                        System.out.println("客户端还有新的请求");
                    } else {
                        isContinue = false;
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
            try {
                // 与服务器连接结束
                in.close();
                out.close();
                reader.close();
                writer.close();
                client.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

}
```

客户端

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.Scanner;

/**
 * 模拟客户端功能 客户端通过读取控制台输入信息，向服务器发送数据 当控制台输入
```

```
exit 时，表示本次输入信息结束
 *
 * 当用户本次输入结束后，要告知服务器客户端传输信息完成， 向服务端发送“end”字符串
 *
 * @author 李玮玮
 *
 */
public class ClientTest {

    public static void main(String[] args) {
        Socket client = null;
        InputStream in = null;
        OutputStream out = null;
        BufferedWriter writer = null;
        BufferedReader reader = null;
        Scanner scanner = null;

        // 用于标识是否继续向服务器提交新的请求，默认为提交(flase:不提交；true:提交)
        boolean isContinue = true;
        try {
            // 创建客户端 Socket 对象
            client = new Socket("127.0.0.1", 8888);
            // 获取输出流对象，用于向服务端发送数据
            out = client.getOutputStream();
            // 构造输出缓冲流
            writer = new BufferedWriter(new OutputStreamWriter(out, "UTF-8"));
            scanner = new Scanner(System.in);
            // 获取输入流对象用于接收服务端返回的数据
            in = client.getInputStream();
            // 构造输入缓冲流
            reader = new BufferedReader(new InputStreamReader(in, "UTF-8"));
        } catch (UnknownHostException e1) {
            e1.printStackTrace();
        } catch (IOException e1) {
            e1.printStackTrace();
        }
        while (isContinue) {
            try {
                // 创建接收控制台输入时保存字符串的临时变量
                String str = null;
                // 每次读取控制台一行字符，并保存在 str 临时变量中,判定是否结束输入
                System.out.println("请在此输入向服务端提交的数据");
```

```
                while (!"exit".equals(str = scanner.nextLine())) {
                    // 将读取的一行字符发送给服务器
                    writer.write(str);
                    // 由于缓冲流不会读取控制台的换行符，因此每次发送一行数据后还需再发送一个空行，表示换行
                    writer.newLine();
                    writer.flush();
                }
                // 控制台输入结束，发送 end 告知服务器数据传输结束
                writer.write("end");
                writer.newLine();
                writer.flush();
                System.out.println("向服务器发送数据完成");

                // 创建用于接收服务端返回数据的临时字符串变量
                String strTemp = null;
                // 每次接收服务端的一行数据，当接收到的数据为“end”时，表示服务器返回数据传输结束
                System.out.println("收到来自服务器的数据：");
                while (!"end".equals(strTemp = reader.readLine())) {
                    System.out.println(strTemp);
                }
                // 询问是否继续与服务器通信
                System.out.println("是否继续向服务器提交请求(Y/N)：");
                String flag = scanner.nextLine();
                if ("Y".equals(flag) || "y".equals(flag)) {
                    isContinue = true;
                    // 告知服务器，客户端还有新的请求
                    writer.write("Y");
                    writer.newLine();
                    writer.flush();
                } else {
                    isContinue = false;
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        try {
            // 与服务器连接结束
            in.close();
            out.close();
            reader.close();
            writer.close();
```

```
                client.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
}
```

14 JDBC 数据库连接技术

14.1 花类

```
public class Flower {
    private int fId;
    private String fName;
    private float fPrice;
    public int getfId() {
        return fId;
    }
    public void setfId(int fId) {
        this.fId = fId;
    }
    public String getfName() {
        return fName;
    }
    public void setfName(String fName) {
        this.fName = fName;
    }
    public float getfPrice() {
        return fPrice;
    }
    public void setfPrice(float fPrice) {
        this.fPrice = fPrice;
    }
    @Override
    public String toString() {
        return "Flower [fId=" + fId + ", fName=" + fName + ", fPrice=" + fPrice + "]";
    }
}
```

14.2 数据库操作类

```
public class DBOperate {
```

```
    //数据库驱动字符串
    private final static String DRIVER_STR = "com.mysql.jdbc.Driver";
    //数据库连接字符串
    private final static String URL_STR = "jdbc:mysql://127.0.0.1:3306/flower_db";
    //登陆数据库的用户名
    private final static String USER = "root";
    //登陆数据库的密码
    private final static String PWD = "";
    //数据库连接对象
    private static Connection conn = null;

    public void addFlowerToDB(List<Flower> flowerList) throws ClassNotFoundException, SQLException {
        Class.forName(DRIVER_STR);
        conn = DriverManager.getConnection(URL_STR, USER, PWD);
        PreparedStatement preStatement = conn.prepareStatement("insert into flower(id, name, price) values (?,?,?)");
        for(Flower flower : flowerList) {
            preStatement.setInt(1, flower.getfId());
            preStatement.setString(2, flower.getfName());
            preStatement.setFloat(3, flower.getfPrice());
            preStatement.execute();
        }
        conn.close();
        System.out.println("入库成功");
    }

    public List showFlowers() throws ClassNotFoundException, SQLException {
        List<Flower> list = new ArrayList<>();
        Class.forName(DRIVER_STR);
        conn = DriverManager.getConnection(URL_STR, USER, PWD);
        Statement statement = conn.createStatement();
        ResultSet rs = statement.executeQuery("select * from flower");
        while(rs.next()) {
            Flower flower = new Flower();
            flower.setfId(rs.getInt("id"));
            flower.setfName(rs.getString("name"));
            flower.setfPrice(rs.getFloat("price"));
            list.add(flower);
        }
        conn.close();
        System.out.println(list);
        return list;
    }
```

```
}
```

14.3 测试类

```
public class Test14 {

	public static void main(String[] args) {
		List<Flower> flowerList = new ArrayList<>();
		Flower rose = new Flower();
		rose.setfId(1);
		rose.setfName("玫瑰");
		rose.setfPrice(100);
		flowerList.add(rose);

		Flower azalea = new Flower();
		rose.setfId(2);
		rose.setfName("杜鹃");
		rose.setfPrice(90);
		flowerList.add(azalea);

		DBOperate dbOperate = new DBOperate();
		try {
			dbOperate.addFlowerToDB(flowerList);
			dbOperate.showFlowers();
		} catch (ClassNotFoundException | SQLException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}

}
```

14.4 数据库操作工具类

```
public class DBUtil {
	//数据库驱动字符串
	private final static String DRIVER_STR = "com.mysql.jdbc.Driver";
	//数据库连接字符串
	private final static String URL_STR = "jdbc:mysql://127.0.0.1:3306/flower_db";
	//登陆数据库的用户名
	private final static String USER = "root";
	//登陆数据库的密码
	private final static String PWD = "";
```

```
//数据库连接对象
private static Connection conn = null;

// 获取数据库连接的方法
public static Connection getConnection() throws SQLException, ClassNotFoundException {
    //判断连接对象是否存在且有效
    if(conn != null && !conn.isClosed()) {
        return conn;
    } else {
        Class.forName(DRIVER_STR);
        conn = DriverManager.getConnection(URL_STR, USER, PWD);
        return conn;
    }
}
// 关闭数据库连接的方法
public static void closeDBConnection() throws SQLException {
    //判断连接对象是否存在且有效
    if(conn != null && !conn.isClosed()) {
        conn.close();
    }
}
// 查询数据库的方法
public static ResultSet queryDate(String sql) throws ClassNotFoundException, SQLException {
    if(conn == null || conn.isClosed()) {
        getConnection();
    }
    Statement statement = conn.createStatement();
    return statement.executeQuery(sql);
}
// 更新数据库的方法
public static int updateData(String sql) throws ClassNotFoundException, SQLException {
    if(conn == null || conn.isClosed()) {
        getConnection();
    }
    Statement statement = conn.createStatement();
    return statement.executeUpdate(sql);
}
// 删除数据库的方法
public static int deleteData(String sql) throws ClassNotFoundException, SQLException {
    if(conn == null || conn.isClosed()) {
        getConnection();
    }
    Statement statement = conn.createStatement();
    return statement.executeUpdate(sql);
```

```
    }
    // 插入数据的方法
    public static int insertData(String sql) throws ClassNotFoundException, SQLException {
        if(conn == null || conn.isClosed()) {
            getConnection();
        }
        Statement statement = conn.createStatement();
        return statement.executeUpdate(sql);
    }
}
```