《Java 程序设计基础与实战（微课版）》实验指导书

课程号：_____

适用专业：_____

制定人：_____

前言

一、上机实验目的

上机实验的目的是提高学生的分析问题、解决问题的能力和动手能力，通过实践环节理解Java语言的基本结构和程序设计方法。通过亲手编程掌握Java语言编程的方法。

二、实验基本内容

为了使学生在上机实验时目标明确，本实验指导书针对课程内容编写了12个实验。学生可以在课内机时先完成指导书中给出的程序，理解所学的知识，在此基础上再编写其他应用程序。指导书中的12个实验如下：

1．Java 程序运行环境的下载、安装与配置。

2．狼人杀游戏。

3．汽车租赁系统。

5．美食烹饪大赛。

6．打卡异常提醒功能。

7．智慧农业月度实际预算统计功能。

8．模拟病毒篡改系统文件内容。

9．智慧儿童早教育儿识色系统。

10.模拟冰墩墩挂件预售功能

11.交友网站聊天程序

12.商场物品种类的新增功能

三、实验任务与时间安排

《Java程序设计基础与实战（微课版）》是一门实践性很强的课程，除了在课内安排的实验外，鼓励同学在课外用相关技术进行编程练习。《Java设计基础与实战（微课版）》上机实验时间为27学时，与教学内容同时进行上机实验。

实验一：Java 编程环境下载、安装、配置与运行

【实验目的】

1. 掌握 JDK 的安装步骤；
2. 掌握 Path 环境变量的配置
3. 掌握编译与运行 Java 程序的方法
4. 掌握 IntelliJ IDEA 软件的使用方法；
5. 为后续 java 程序的实验做准备

【实验要求】

1. 预习试验内容并写出上机报告。
2. 实验中出现的问题及实验体会。

【实验内容】

1. 下载并安装 JDK1.8

（1）使用浏览器访问 Oracle 官网进入 Java downloads 页面选择 Java8 版本进行下载。也可使用本书第 1 章资源包中提供的安装文件。

（2）双击运行下载的安装文件进行安装，在安装过程中可以设置安装路径及选择组件，建议初次安装使用默认选项。

2. 设置 Path 环境变量

（1）在系统环境变量的 Path 变量中增加值为 JDK 的安装路径。

（2）打开 cmd 命令提示符窗口，输入命令“javac”，若出现以下结果，则表示安装配置成功。

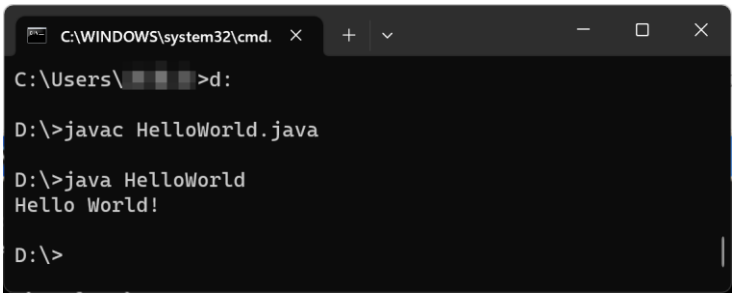
3. 使用记事本编写和运行 Java 程序

（1）使用操作系统提供的记事本作为编辑器。在电脑 D 盘新建文本文件命名为 HelloWorld，并修改后缀名为.java。在 HelloWorld.java 中编写如下代码。

（2）打开命令提示符窗口，进入 HelloWorld.java 所在路径：d:。

（3）键入编译器和要编译的源程序文件名：javac HelloWorld.java。按回车键开始编译。

（4）在命令提示符窗口键入解释器和要解释的字节码文件名：java Hello。按回车键即开始执行程序并输出结果。注意：字节码文件区分大小写。

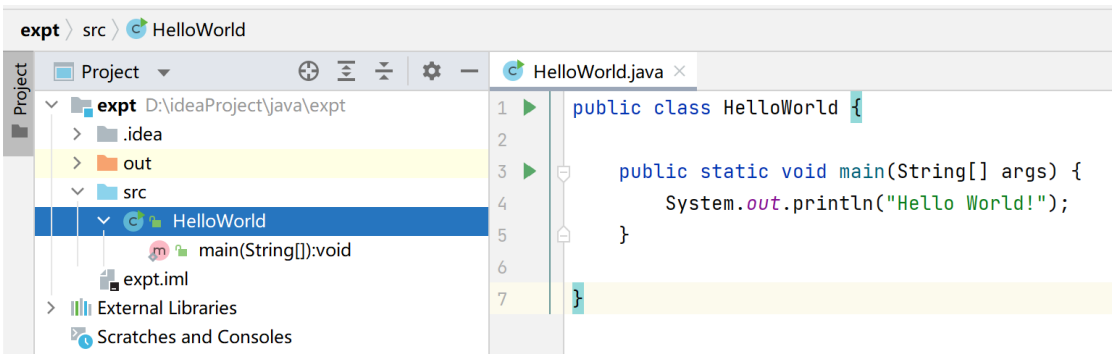


4. 安装 IntelliJ IDEA

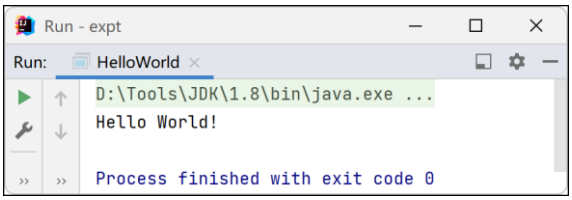
（1）使用浏览器访问 Jetbrains 官网单击 Developer Tools，选择 Intellij IDEA（以下简简称 IDEA）进行下载，也可使用本书第 1 章资源包中提供的安装文件。

（2）双击运行下载的安装文件进行安装，安装过程参考教材中步骤。

（3）在 IDEA 中新建 Project，命名为 expt01。在 expt01 项目下 src 目录下新建类 HelloWorld，在 HelloWorld.java 中编写如下代码。



运行 HelloWord.java 的结果如下图所示。


Run - expt
Run:
HelloWorld
D:\Tools\JDK\1.8\bin\java.exe ...
Hello World!
Process finished with exit code 0

实验二：狼人杀游戏

【实验目的】

1. 了解 Java 的数据类型
2. 掌握各种变量的声明方式
3. 理解运算符的优先级
4. 掌握 Java 基本数据类型。运算符与表达式、数组的使用方法
5. 理解 Java 程序语法结构，掌握顺序结构、选择结构和循环结构语法的程序设计方法

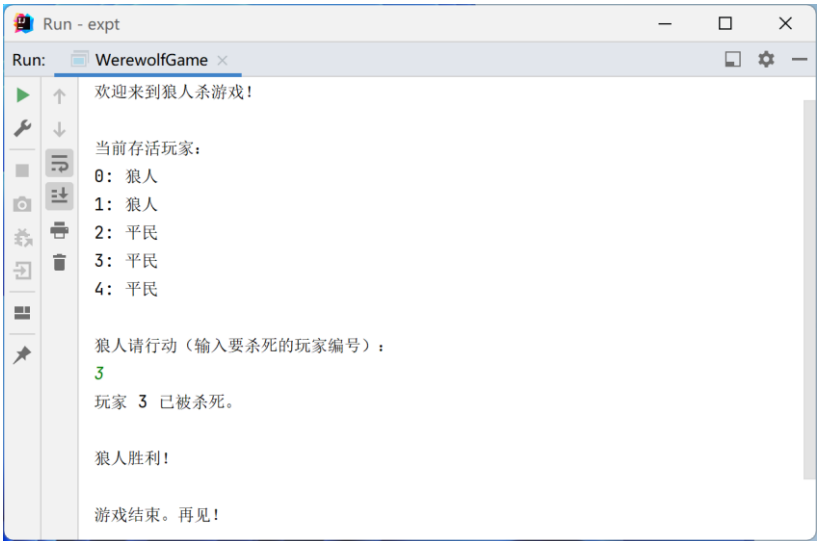
【实验要求】

1. 预习试验内容并写出上机报告。
2. 实验中出现的问题及实验体会。

【实验内容】

1. 初始化游戏角色数组，包括狼人和平民，以及存活状态数组
2. 进入游戏循环，直到游戏结束
3. 显示当前存活的玩家信息，包括玩家编号和角色
4. 狼人行动，输入要杀死的玩家编号，如果选择杀死同伴则无效，否则将该玩家的存活状态设置为死亡
5. 判断游戏是否结束，如果狼人数大于等于平民数则狼人胜利，如果狼人数为 0 则平民胜利
6. 如果游戏未结束，则进入平民行动，包括发言和投票，输入要投票的玩家编号，将该玩家的存活状态设置为死亡
7. 再次判断游戏是否结束
8. 游戏结束，输出游戏结束信息

狼人杀游戏运行结果如下图所示。



狼人杀游戏的具体代码如下

```
import java.util.Scanner;
public class WerewolfGame {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("欢迎来到狼人杀游戏！");
        // 定义游戏角色数组
        String[] roles = {"狼人", "狼人", "平民", "平民", "平民"};
        int numPlayers = roles.length;
        boolean[] isAlive = new boolean[numPlayers];
        // 初始化状态
        for (int i = 0; i < numPlayers; i++) {
            isAlive[i] = true;
        }
        // 游戏开始
        boolean gameOver = false;
        while (!gameOver) {
            // 显示当前状态
            System.out.println("\n当前存活玩家：");
            for (int i = 0; i < numPlayers; i++) {
                if (isAlive[i]) {
```

```
                System.out.println(i + ": " + roles[i]);
            }
        }
        // 狼人行动
        System.out.println("\n狼人请行动(输入要杀死的玩家编号):");
        int wolfTarget = scanner.nextInt();
        if (roles[wolfTarget].equals("狼人")) {
            System.out.println("不能杀死同伴!");
        } else {
            isAlive[wolfTarget] = false;
            System.out.println("玩家 " + wolfTarget + " 已被杀死。
");

        }
        // 判断游戏是否结束
        int numWolves = 0;
        int numVillagers = 0;
        for (int i = 0; i < numPlayers; i++) {
            if (isAlive[i]) {
                if (roles[i].equals("狼人")) {
                    numWolves++;
                } else {
                    numVillagers++;
                }
            }
        }
        if (numWolves > numVillagers) {
            System.out.println("\n狼人胜利!");
            gameOver = true;
        } else if (numWolves == 0) {
            System.out.println("\n村民胜利!");
```

```
                gameOver = true;
            }
            // 平民行动
            if (!gameOver) {
                System.out.println("\n天亮了，请发言并投票（输入要投票的玩家编号）：");
                int voteTarget = scanner.nextInt();
                if (isAlive[voteTarget]) {
                    System.out.println("玩家 " + voteTarget + " 已被投票出局。");
                    isAlive[voteTarget] = false;
                } else {
                    System.out.println("该玩家已经死亡或不存在！");
                }
                // 判断游戏是否结束
                numWolves = 0;
                numVillagers = 0;
                for (int i = 0; i < numPlayers; i++) {
                    if (isAlive[i]) {
                        if (roles[i].equals("狼人")) {
                            numWolves++;
                        } else {
                            numVillagers++;
                        }
                    }
                }
                if (numWolves >= numVillagers) {
                    System.out.println("\n狼人胜利！");
                    gameOver = true;
                } else if (numWolves == 0) {
```

```
                    System.out.println("\n 村民胜利！");
                    gameOver = true;
                }
            }
        }
        System.out.println("\n 游戏结束。再见！");
    }
}
```

实验三 汽车租赁系统

【实验目的】

1. 掌握面向对象设计基本步骤
2. 掌握类和对象的概念
3. 掌握构造方法及其重载
4. 了解成员变量和成员方法的特性。

【实验要求】

1. 预习试验内容并写出上机报告。
2. 实验中出现的问题及实验体会。

【实验内容】

设计一个简单的汽车租赁系统，系统包含汽车类和租赁类，要求如下：

汽车类 Car：

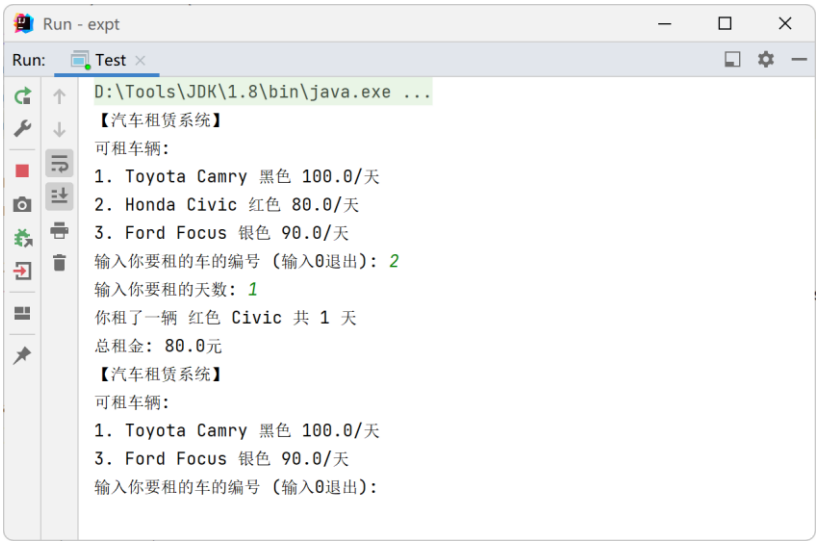
- 属性：汽车品牌、车型、颜色、租金每日价格、是否被租走
- 方法：构造方法、获取车型、获取颜色、获取租金价格、是否被租走、租赁该车

租赁类 Rental：

- 属性：租赁天数、租赁的汽车对象、总租金
- 方法：构造方法、计算租金、获取租赁天数、获取租赁汽车、获取总租金

要求用户可以查看当前可用的汽车并进行租赁，租赁时需要输入租赁天数，系统会计算总租金并将租赁的汽车标记为已被租走。还车时需要将租赁的汽车标记为未被租走。

汽车租赁系统的运行截图如下所示。



汽车类 Car.java 的代码如下：

```
public class Car {
    private String brand;
    private String model;
    private String color;
    private double rentPrice;
    private boolean rented;
    public Car(String brand, String model, String color, double rentPrice)
{
        this.brand = brand;
        this.model = model;
        this.color = color;
        this.rentPrice = rentPrice;
        this.rented = false;
    }
    public String getModel() {
        return model;
    }
    public String getColor() {
        return color;
    }
```

```
    public double getRentPrice() {
        return rentPrice;
    }
    public boolean isRented() {
        return rented;
    }
    public void rent() {
        rented = true;
    }
    public void returnCar() {
        rented = false;
    }
}
```

租赁类 Rental.java 的代码如下：

```
public class Rental {
    private int rentalDays;
    private Car car;
    private double totalPrice;
    public Rental(Car car, int rentalDays) {
        this.car = car;
        this.rentalDays = rentalDays;
        this.totalPrice = rentalDays * car.getRentPrice();
        car.rent();
    }
    public int getRentalDays() {
        return rentalDays;
    }

    public Car getCar() {
        return car;
```

```
    }
    public double getTotalPrice() {
        return totalPrice;
    }
    public void returnCar() {
        car.returnCar();
    }
}
```

测试类代码如下：

```
public class Test {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Car[] cars = new Car[3]; // 初始化 3 辆车
        cars[0] = new Car("Toyota", "Camry", "黑色", 100, false);
        cars[1] = new Car("Honda", "Civic", "红色", 80, false);
        cars[2] = new Car("Ford", "Focus", "银色", 90, false);
        while (true) {
            System.out.println("【汽车租赁系统】");
            System.out.println("可租车辆:");
            for (int i = 0; i < cars.length; i++) {
                if (!cars[i].isRented()) {
                    System.out.println(i + 1 + ". " +cars[i].getBrand() + " "+cars[i].getModel() + " " + cars[i].getColor() + " " + cars[i].getRentPrice() + "/天");
                }
            }
            System.out.print("输入你要租的车的编号 (输入 0 退出): ");
            int carIndex = scanner.nextInt();
            if (carIndex == 0) {
                break;
```

```
                }
                if (carIndex < 1 || carIndex > cars.length) {
                    System.out.println("该车可租");
                    continue;
                }
                if (cars[carIndex - 1].isRented()) {
                    System.out.println("该车已租");
                    continue;
                }
                System.out.print("输入你要租的天数: ");
                int rentalDays = scanner.nextInt();
                if (rentalDays <= 0) {
                    System.out.println("租赁天数不合法!");
                    continue;
                }
                Rental rental = new Rental(cars[carIndex - 1], rentalDays);
                double totalRent = rental.getTotalPrice();
                System.out.println("你租了一辆 " + cars[carIndex - 1].getColor() + " " + cars[carIndex - 1].getModel() + " 共 " + rentalDays + " 天");
                System.out.println("总租金: " + totalRent+"元");
                cars[carIndex - 1].rent();
            }
        }
    }
```

实验四 智能图形设计师

【实验目的】

1. 掌握封装的概念及其使用
2. 了解类的封装方法
3. 掌握 OOP 方式进行程序设计的方法
4. 了解类的继承性和多态性的作用。

【实验要求】

1. 预习试验内容并写出上机报告。
2. 实验中出现的问题及实验体会。

【实验内容】

1. 创建一个基类 Shape，包含以下属性和方法：
    - 属性：颜色 color、是否填充 fill、边框宽度 borderWidth
    - 方法：构造方法、获取和设置属性的方法

Shape.java 的代码如下。

```
public class Shape {
    protected double area;
    public double getArea() {
        return area;
    }
}
```

2. 创建 Shape 的子类 Rectangle，包含以下属性和方法：
    - 属性：宽度 width、高度 height
    - 方法：构造方法、获取和设置属性的方法、计算面积方法 area()

Rectangle.java 的示例代码如下。

```
public class Rectangle extends Shape {
    private double length;
    private double width;

    public Rectangle(double length, double width) {
```

```
        this.length = length;
        this.width = width;
        this.area = length * width;
    }
}
```

3. 创建 Shape 的子类 Circle，包含以下属性和方法：
    - 属性：半径 radius
    - 方法：构造方法、获取和设置属性的方法、计算面积方法 area()

Circle.java 的代码如下。

```
public class Circle extends Shape{
    private double radius;
    public Circle(double radius) {
        this.radius = radius;
        this.area = Math.PI * radius * radius;
    }
}
```

4. 创建 Shape 的子类 Triangle，包含以下属性和方法：
    - 属性：三角形三边 a、b、c
    - 方法：构造方法、获取和设置属性的方法、计算面积方法 area()

```
public class Triangle extends Shape {
    private double base;
    private double height;

    public Triangle(double base, double height) {
        this.base = base;
        this.height = height;
        this.area = 0.5 * base * height;
    }
}
```

5. 创建一个图形类管理器 ShapeManager，包含以下方法：

- 添加图形方法 add(Shape shape)
- 移除图形方法 remove(Shape shape)
- 显示所有图形方法 display()
- 在主函数中，创建各种不同的图形对象，添加到图形管理器中，然后调用显示所有图形的方法。

ShapeManager.java 的代码如下。

```
public class ShapeManager {
    private Shape[] shapes;
    public ShapeManager(int size) {
        shapes = new Shape[size];
    }
    public void addShape(Shape shape, int index) {
        shapes[index] = shape;
    }
    public double getTotalArea() {
        double totalArea = 0;
        for (Shape shape : shapes) {
            if (shape != null) {
                totalArea += shape.getArea();
            }
        }
        return totalArea;
    }
}
```

实验五 美食烹饪大赛

【实验目的】

1．掌握抽象类使用。

2．掌握接口的使用。

3．掌握 Lambda 表达式用法。

【实验要求】

1．预习试验内容并写出上机报告。

2．实验中出现的问题及实验体会。

【实验内容】

1．创建一个抽象类，包含姓名和拿手菜属性及做菜的抽象方法。

2．创建一个接口，含有做菜种类的方法。

3．创建两个人张三和李四分别继承抽象类和实现做菜接口。

4．实现抽象类和接口里的方法，并成功打印如下图所示。

```
-------比赛开始---------
我是张三我做的是鱼肉
用蒸的方式做菜
==========><==========
我是李四我做的是猪肉
用炒的方式做菜
比赛结束，评委投票
张三
李四
张三
李四
张三
-------->张三胜
```

美食烹饪大赛的代码如下。

其中抽象类代码如下：

```
abstract public class People {
    private String name; //姓名
    private String kind; //拿手菜
    public People(String name, String kind) {
        this.name = name;
```

```
        this.kind = kind;
    }
    public People() {
    }
    public abstract  void knife();
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getKind() {
        return kind;
    }
    public void setKind(String kind) {
        this.kind = kind;
    }
}
```

张三实体类代码如下：

```
public class ZhangSan extends People implements DoFood{
    public ZhangSan() {
        super();
    }
    public ZhangSan(String name, String kind) {
        super(name, kind);
    }
    @Override
    public void knife() {
        System.out.println("用蒸的方式做菜");
    }
```

```
    @Override
    public void makeFood(String name,String kind) {
        System.out.println("我是" + name + "我做的是" + kind);
    }
}
```

李四实体类代码如下：

```
public class LiSi extends People implements DoFood{
    public LiSi(String name, String kind) {
        super(name, kind);
    }
    public LiSi() {
    }
    @Override
    public void makeFood(String name,String kind) {
        System.out.println("我是" + name + "我做的是" + kind);
    }
    @Override
    public void knife() {
        System.out.println("用炒的方式做菜");
    }
}
```

接口类代码如下：

```
public interface DoFood {
    //做饭过程
    void  makeFood(String name,String kind);
    public static void main(String[] args) {
        ZhangSan zhangSan = new ZhangSan();
        LiSi liSi = new LiSi();
```

```
        System.out.println("-------比赛开始---------");
        zhangSan.makeFood("张三","鱼肉");
        zhangSan.knife();
        System.out.println("==========><==========");
        liSi.makeFood("李四","猪肉");
        liSi.knife();
        System.out.println("比赛结束，评委投票");
        List<String> vote = new ArrayList<>();
        vote.add("张三");
        vote.add("李四");
        vote.add("张三");
        vote.add("李四");
        vote.add("张三");
        vote.stream().forEach(System.out::println);
        System.out.println("-------->张三胜");
    }
}
```

实验六 打卡异常提醒功能

【实验目的】

1．理解异常的概念。

2．理解异常的类型。

3．掌握异常处理的方式。

4．了解自定义异常类。

5．掌握常用类的使用方法。

6．掌握日期操作类的使用方法。

【实验要求】

1．预习试验内容并写出上机报告。

2．实验中出现的问题及实验体会。

【实验内容】

1．自定义一个异常类并继承 RuntimeException。

2．使用日期格式化类获得当前时间。

3．若当前时间晚于截止打卡时间，抛出自定义异常，如下图所示。

```
当前时间：11:52:56
超过早9点不允许打卡
```

打卡异常提醒功能代码如下。

```
public class MyException extends RuntimeException {
    public MyException() {
    }
    public MyException(String s) {
        super(s);
    }
    public static void main(String[] args) {
        SimpleDateFormat simpleDateFormat = new
SimpleDateFormat("HH:mm:ss");//日期格式化类
        String format = simpleDateFormat.format(new Date());//当前时间转成
自定义字符串类型
```

```
        Date parse = null;
        try {
            parse = simpleDateFormat.parse("09:00:00");//截止打卡时间日期
        } catch (ParseException e) {
            e.printStackTrace();
        }
        System.out.println("当前时间：" + format);
        if (new Date().after(parse)) {//比较当前日期是否大于早 9 点
            try {
                throw new MyException("超过早 9 点不允许打卡");//抛出自定义
异常
            } catch (MyException e) {
                System.out.println(e.getMessage());
            }
        }
    }
}
```

实验七 智慧农业月度实际预算统计功能

【实验目的】

1．掌握集合的使用。

2．掌握泛型的使用。

3．掌握集合遍历的方法。

【实验要求】

1．预习试验内容并写出上机报告。

2．实验中出现的问题及实验体会。

【实验内容】

1．创建水稻，小麦，玉米三个部门的 List 预算集合。

2．收集各部门员工的实际报销金额。

3．使用迭代器计算各个部门的实际报销金额。

4．汇总所有部门实际花费金额和一共花费金额，并输出打印如下图所示。

```
----------本季度总费用----------
玉米部门本季度一共报销了：140.57元
水稻部门本季度一共报销了：100.1元
小麦部门本季度一共报销了：89.9元

------------------------------
合计：本年度农业部门一共花费330.57元
```

智慧农业实际预算统计管理代码如下。

```
public class Agricultural {
    public static void main(String args[]) {
        List<Double> rice = new ArrayList<>(); // 水稻部门的实际总预算集
合
        Double riceMoney = 0.0; //水稻一共花费金额初始变量
        rice.add(50.20);//小李本季度报销 50.2 元。
        rice.add(29.58);//小王本季度报销 29.58 元。
        rice.add(20.32);//小张本季度报销 20.32 元。
        Iterator<Double> riceIterator = rice.iterator();
        while (riceIterator.hasNext()){//使用迭代器计算水稻部门总花费
            riceMoney += riceIterator.next();
```

```
        }
        List<Double> wheat = new ArrayList<>(); // 小麦部门的实际总预算
集合
        Double wheatMoney = 0.0;//小麦一共花费金额初始变量
        wheat.add(13.12);//小赵本季度报销 13.12 元。
        wheat.add(66.66);//小钱本季度报销 66.66 元。
        wheat.add(10.12);//小孙本季度报销 10.12 元。
        Iterator<Double> wheatIterator = wheat.iterator();
        while (wheatIterator.hasNext()){//使用迭代器计算小麦部门总花费
            wheatMoney += wheatIterator.next();
        }
        List<Double> corn = new ArrayList<>(); // 玉米部门的实际总预算集
合
        Double cornMoney = 0.0;//玉米一共花费金额初始变量
        corn.add(30.32);//小赵本季度报销 30.32 元。
        corn.add(95.32);//小钱本季度报销 95.32 元。
        corn.add(14.93);//小孙本季度报销 14.93 元。
        Iterator<Double> cornIterator = corn.iterator();
        while (cornIterator.hasNext()){//使用迭代器计算玉米部门总花费
            cornMoney += cornIterator.next();
        }
        // 各部门预算统计
        Map<String,Double>  budget = new HashMap<>();
        budget.put("水稻",riceMoney);
        budget.put("小麦",wheatMoney);
        budget.put("玉米",cornMoney);
        Iterator<String> budgetIterator =budget.keySet().iterator();
        Double total = 0.0;//所有部门一共实际花费总金额初始变量
        System.out.println("_________本季度总费用__________");
        while (budgetIterator.hasNext()){//迭代器判下一个元素是否有值
```

```
            String key = budgetIterator.next();//拿到下一个元素的key
            System.out.println(key  +  "部门本季度一共报销了："+
budget.get(key) + "元");
            total +=  budget.get(key);//拿到该Key对应的Map具体值
        }
        System.out.println("_____________________________");
        System.out.println("合计：本年度农业部门一共花费" + total + "元
");
    }
}
```

实验八 模仿病毒篡改系统文件内容

【实验目的】

1．掌握使用字节流和字符流读写文件。

2．了解其他 IO 流。

3．熟练掌握 File 类及其用法。

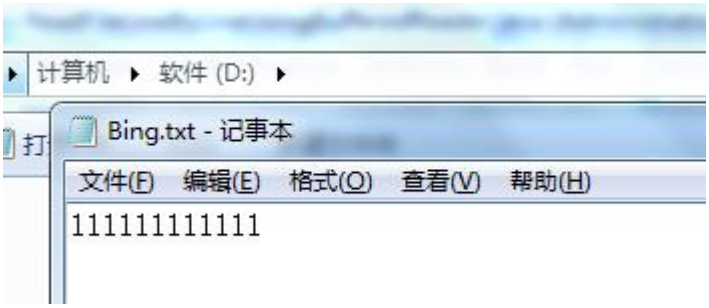
【实验要求】

1．预习试验内容并写出上机报告。

2．实验中出现的问题及实验体会。

【实验内容】

1．在 D 盘目录下，新建一个名为 Bing 的 txt 文件。输入内容 111111111111，如下图所示。

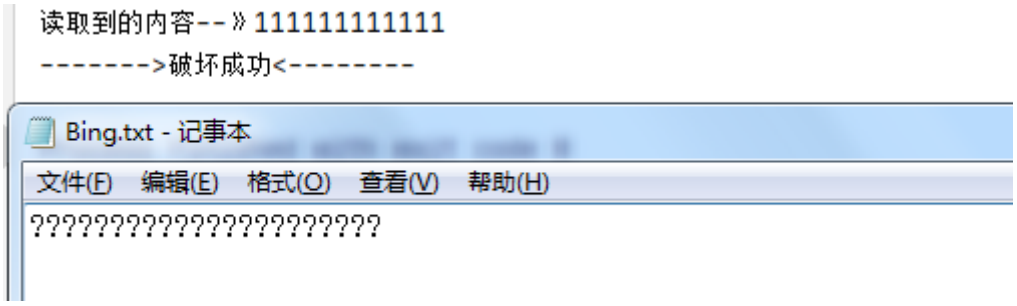


2．使用 File 类判断当前目录下是否含有此文件，有则破坏文件内容。

3．创建带缓冲的读取字符流读取文件的第一行内容，并打印出来。

4．创建写入字符流，并写入“??????????????????????”内容。

5．查看控制台和 Bing 文本发现内容确实被篡改破坏，如下图所示。



模仿病毒篡改系统文件的代码如下。

```
public class ReadFileLineByLineUsingBufferedReader {
    public static void main(String[] args) {
        BufferedReader reader; //创建读取文件缓冲流全局变量
```

```
        File file = new File("D:/bing.txt");
        if(file.exists()){//判断文件是否存在
            try {
                reader = new BufferedReader(new FileReader(
                        "D:/bing.txt"));//实例化读取文件缓冲流
                String line = reader.readLine();//读到下一行数据
                while (line != null) {//不为空继续读下一行
                    System.out.println("读取到的内容-->>" + line);//打印每一行内容
                    line = reader.readLine();//继续下一行
                }
                reader.close();//关闭缓冲流
            } catch (IOException e) {
                e.printStackTrace();
            }
            Writer writer;//定义写入字符流全局变量
            {
                try {
                    writer = new FileWriter("D:/bing.txt");//实例化字节写入流
                    writer.write("??????????????????????");//写入内容
                    System.out.println("------->破坏成功<--------");
                    writer.close();//关闭字节流
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
```

```
            }
        }
    }
```

实验九 智慧儿童早教育儿识色系统

【实验目的】

1．掌握常用的 Swing 组件的使用办法。

2．了解常用的窗体和布局管理器。

3．熟练掌握事件监听器的使用方法。

【实验要求】

1．预习试验内容并写出上机报告。

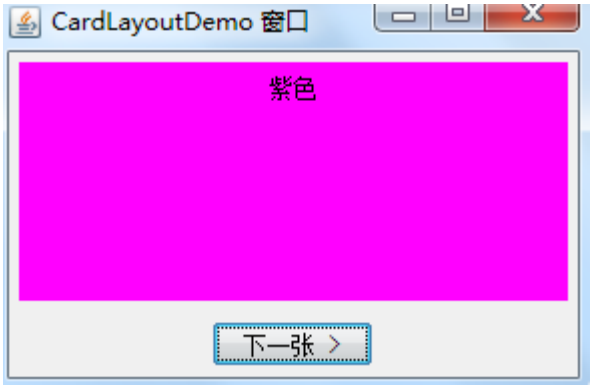
2．实验中出现的问题及实验体会。

【实验内容】

1．创建图形用户界面，在 Applet 容器中添加组件标签、按钮，并使用网格布局管理器排列组件在容器中的位置。

2．编写顶层容器 Container 和中间容器 Jpane 的 CardLayout(共设置六种颜色)。

3．要求给点击按钮注册监听器，点击按钮时可切换容器中的下一个颜色。效果如下图所示。



智慧儿童早教育儿识色功能代码如下。

```
public class CardLayoutDemo extends JFrame {
private JPanel cardPanel = null; // 主要的 JPanel，该 JPanel 的布局管理将被设置成 CardLayout
private JPanel controlPanel = null; // 放按钮的 JPanel
private CardLayout card = null; // CardLayout 布局管理器
private JButton btnNext = null; // 下一步
private JPanel p_1 = null, p_2 = null, p_3 = null,p_4 = null, p_5 = null, p_6 = null; // 要切换的三个 JPanel
```

```
    public CardLayoutDemo() {
        super("CardLayoutDemo 窗口");
        try {
            // 将 LookAndFeel 设置成 Windows 样式

UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFee
l");
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        /**创建一个具有指定的水平和垂直间隙的新卡片布局*/
        card = new CardLayout(5, 5);
        cardPanel = new JPanel(card); // JPanel 的布局管理将被设置成
CardLayout
        controlPanel = new JPanel(); // 构造放按钮的 JPanel
        btnNext = new JButton("下一张 >");
        controlPanel.add(btnNext);
        p_1 = new JPanel();
        p_2 = new JPanel();
        p_3 = new JPanel();
        p_4 = new JPanel();
        p_5 = new JPanel();
        p_6 = new JPanel();
        //设置每页的颜色
        p_1.setBackground(Color.GREEN);
        p_2.setBackground(Color.MAGENTA);
        p_3.setBackground(Color.WHITE);
        p_4.setBackground(Color.PINK);
        p_5.setBackground(Color.CYAN);
```

```
        p_6.setBackground(Color.ORANGE);
        //设置每页的标题名字
        p_1.add(new JLabel("绿色"));
        p_2.add(new JLabel("紫色"));
        p_3.add(new JLabel("白色"));
        p_4.add(new JLabel("粉色"));
        p_5.add(new JLabel("绿色"));
        p_6.add(new JLabel("橙色"));
        cardPanel.add(p_1, "p1");
        cardPanel.add(p_2, "p2");
        cardPanel.add(p_3, "p3");
        cardPanel.add(p_4, "p1");
        cardPanel.add(p_5, "p2");
        cardPanel.add(p_6, "p3");
        /**下面是翻转到卡片布局的某个组件，可参考 API 中的文档*/
        btnNext.addActionListener(new ActionListener(){ // 下一步的按钮动作
            public void actionPerformed(ActionEvent e) {
                card.next(cardPanel);
            }
        });
        this.getContentPane().add(cardPanel);
        this.getContentPane().add(controlPanel, BorderLayout.SOUTH);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setSize(300, 200);
        this.setVisible(true);
    }
    public static void main(String[] args) {
        new CardLayoutDemo();
    }
```

}

实验十 模拟冰墩墩挂件预售功能

【实验目的】

1．掌握创建线程的方式。

2．了解线程的生命周期及状态转换。

3．掌握多线程的同步机制。

4．理解进程和线程的区别。

【实验要求】

1．预习试验内容并写出上机报告。

2．实验中出现的问题及实验体会。

【实验内容】

1．创建平台实体类并继承 Thread 类。

2．自定义本次预售数量和同步锁的 key。

3．重写 Thread 类中的 run 方法，定义预售规则。

4．模拟三个平台同时预售冰墩墩挂件，未出现超卖现象。如下图所示。

```
官网卖出了第10个冰墩墩
直营店卖出了第9个冰墩墩
授权店卖出了第8个冰墩墩
直营店卖出了第7个冰墩墩
授权店卖出了第6个冰墩墩
官网卖出了第5个冰墩墩
直营店卖出了第4个冰墩墩
授权店卖出了第3个冰墩墩
官网卖出了第2个冰墩墩
直营店卖出了第1个冰墩墩
冰墩墩卖完了
```

模拟冰墩墩挂机预售代码如下。

```
public class Panda extends Thread {
    // 通过构造方法给线程名字赋值
    public Panda(String name) {
        super(name);// 给线程名字赋值
    }
    // 为了保持冰墩墩数的一致，冰墩墩数要静态变量
```

```
    static int tick = 10;
    // 创建一个静态钥匙
    static Object ob = "aa";//值可以是任意的
    // 重写 run 方法，实现买冰墩墩挂件操作
    @Override
    public void run() {
        while (tick > 0) {
            synchronized (ob) {// 这个很重要，必须使用一个锁，
                // 进去的人会把钥匙拿在手上，出来后才把钥匙拿让出来
                if (tick > 0) {
                    System.out.println(getName() + "卖出了第" + tick + "
个冰墩墩");
                    tick--;
                } else {
                    System.out.println("冰墩墩卖完了");
                }
            }
            try {
                sleep(1000);//休息一秒
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
    public static void main(String[] args) {
        /**
         * java 多线程同步锁的使用
         * 示例：三个平台同时出售冰墩墩
         */
        //实例化冰墩墩对象，并为每一个平台取名字
```

```
        Panda panda1 =new Panda("官网");
        Panda panda2 =new Panda("直营店");
        Panda panda3 =new Panda("授权店");
        // 让每一个站台对象各自开始工作
        panda1.start();
        panda2.start();
        panda3.start();
    }
}
```

实验十一 交友网站聊天程序

【实验目的】

1．了解网络通信协议。

2．熟练掌握 UDP 和 TCP 通信。

3．熟练掌握网络程序的开发方法。

【实验要求】

1．预习试验内容并写出上机报告。

2．实验中出现的问题及实验体会。

【实验内容】

1．新建一个客户端类。

2．创建一个与服务端连接的管道，一个线程负责客户端的消息读取和一个字节输入流管道。

3．通过 Socket 把数据发送出去，然后刷新流。

4．创建一个服务端实体类，进行服务端口注册。

5．使用 Socket 监听客户端发送来的消息、上线和离线相应。

6．使用客户端给服务端发送消息，成功发出，服务端并成功收到。

交友网站聊天程序的运行效果如下图所示。

```
客户端启动
请输入：
很高兴认识你
请输入：
收到了消息很高兴认识你
交个朋友
请输入：
收到了消息交个朋友
```

```
服务端启动成功
/192.168.1.145:63070上线了
/192.168.1.145:63070说很高兴认识你
/192.168.1.145:63070说交个朋友
```

交友网站聊天程序代码如下。

客户端代码：

```
public class Client {
    public static void main(String[] args) throws Exception {
        System.out.println("客户端启动");
        //1.创建与服务端连接的管道
        Socket s = new Socket(InetAddress.getLocalHost(), 9966);
        //2.创建一个线程负责客户端的消息读取
        new ClientReaderThread(s).start();
        //3.创建一个字节输入流管道
        OutputStream o = s.getOutputStream();
        PrintStream p = new PrintStream(o);//升级流
        //4.客户端输入数据
        Scanner sc = new Scanner(System.in);
        while (true){
            System.out.println("请输入：");
            String s1 = sc.nextLine();
            p.println(s1);//发送数据出去
            p.flush();//刷新流
        }
    }
}
class ClientReaderThread extends Thread{
    private Socket socket;
    public ClientReaderThread(Socket socket){
        this.socket = socket;
    }
    @Override
    public void run(){
        try {
            //把字节输入流包装成字符输入流
            InputStream i = socket.getInputStream();
```

```
            BufferedReader b = new BufferedReader(new
InputStreamReader(i));
            String s1;
            while (true){
                if ((s1=b.readLine())!=null){
                    System.out.println("收到了消息"+s1);
                }
            }
        } catch (IOException e) {
            System.out.println("服务器把你提出群聊");
        }
    }
}
```

服务端代码如下：

```
public class Server {
    //1.定义一个静态变量储存全部管道
    public static List<Socket> all_Sockets = new ArrayList<>();
    public static void main(String[] args) throws IOException {
        System.out.println("服务端启动成功");
        //2.服务端口注册
        ServerSocket ss = new ServerSocket(9966);
        //3.管道死循环设置
        while (true){
            Socket s = ss.accept();
            System.out.println(s.getRemoteSocketAddress()+"上线了");
            all_Sockets.add(s);
            new ServerThread(s).start();
        }
    }
}
```

```
class ServerThread extends Thread{
    private Socket socket;
    public ServerThread(Socket socket){
        this.socket=socket;
    }
    @Override
    public void run(){
        try {
            InputStream i = socket.getInputStream();
            BufferedReader b = new BufferedReader(new
InputStreamReader(i));
            String s1;
            while ((s1=b.readLine())!=null){
                System.out.println(socket.getRemoteSocketAddress()+" 说
"+s1);
                sendMessage(s1);
            }
        } catch (IOException e) {
            System.out.println(socket.getRemoteSocketAddress()+"离线了
");
            Server.all_Sockets.remove(socket);
        }
    }
    private void sendMessage (String s1) throws IOException {
        for (Socket s:Server.all_Sockets){
            OutputStream o = s.getOutputStream();
            PrintStream p = new PrintStream(o);
            p.println(s1);
            p.flush();
        }
```

```
        }

    }
```

实验十二 商场物品种类的新增功能

【实验目的】

1. 了解 JDBC。

2. 掌握 JDBC 的常用类和接口。

3. 掌握使用 JDBC 操作数据库。

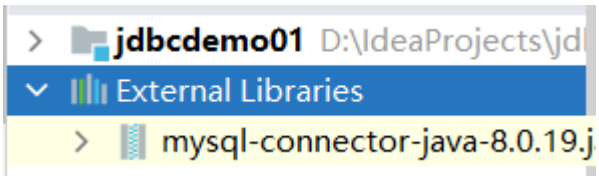
【实验要求】

1. 预习试验内容并写出上机报告。

2. 实验中出现的问题及实验体会。

【实验内容】

1. 下载 MySQL8 驱动 jar 包放在 IDEA 软件的 External Library 目录下，如下图所示。



2. 配置正确的数据库地址、账号、密码和端口号。

3. 获取数据库的 Connection 连接。

4. 使用 Statement 或 Prestatement 执行自定义的 Sql 语句。

5. 对结果集进行进一步处理，完成商品种类的新增，如下图所示。

id	name
1	手机电脑
2	家纺
4	家具
5	汽车

6. 关闭所有连接流。

商场物品种类新增功能的代码如下。

```
public class jdbcDemo {
    public static void main(String[] args) throws SQLException {
        //1.加载驱动
        try {
```

```
            Class.forName("com.mysql.cj.jdbc.Driver");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
        //2.获取链接,驱动管理器
        //数据库地址
        String
url="jdbc:mysql://localhost:3306/ishop?useUnicode=true&characterEncodin
g=utf-8&useSSL=false&serverTimezone=GMT%2B8";
        String user="root";//数据库账号
        String password="root";//数据库密码
        Connection connection = null;//数据库连接
        Statement statement = null;//执行 Sql 命令类
        try {
            connection    =    DriverManager.getConnection(url,    user,
password);
        } catch (SQLException e) {
            e.printStackTrace();
        }
        try {
            statement = connection.createStatement();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        //自定义新增 Sql，添加商品种类为汽车
        String sql="insert into tbl_commoditytype (id,name) values (5,'
汽车')";
        int i = 0;
        //executeUpdate 是做增删改的
        //4.得到结果集并处理
```

```
        try {
            i = statement.executeUpdate(sql);
            if (i>1){
                System.out.println("新增商品种类成功");
            }else {
                System.out.println("新增失败");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }finally {
            //5.关闭资源
            statement.close();
            connection.close();
        }
    }
}
```