厦門大學

信息学院软件工程系

《JAVA 程序设计》实验报告

实验 6

姓名：黄勖

学号：22920212204392

学院：信息学院

专业：软件工程

完成时间：2023.4.4

一、实验目的及要求

- 熟悉继承
- 熟悉多态
- 学习异常处理
- 掌握根据需求设计类的方法

二、实验题目及实现过程

实验环境：Windows 10 21H2、jdk17、utf-8 编码

（一） 实验题目

- 2019 上机考试模拟
- 某车队有若干小汽车和卡车，现写一个程序实现新增车辆、查询车辆和列出车辆信息等功能。
- 其中，小汽车有商标、颜色、载客量(人)、出厂年、车厢数(2 厢或 3 厢)等属性，卡车有商标、颜色、载重量(吨)、出厂年等属性。

（二） 实现过程

思路：主要需要设计的是一个简单的 Java 控制台程序编程题目，实现车辆信息的增加、查询和展示。主要包含以下几个类：

- Main 类：程序入口，包含 main 方法，展示程序主菜单，根据用户选择调用其他类的方法来实现对车辆信息的操作。
- Vehicle 类：车辆类，是其他具体车辆类的父类，包含车辆的品牌、颜色、出厂年份等基本信息，以及展示车辆信息的方法。
- Car 类：小汽车类，继承自 Vehicle 类，增加了车辆类型、座位数等属性，并重写了展示车辆信息的方法。（实现多态）
- Trunk 类：卡车类，继承自 Vehicle 类，增加了车辆载重等属性，并重写了展示车辆信息的方法。（实现多态）

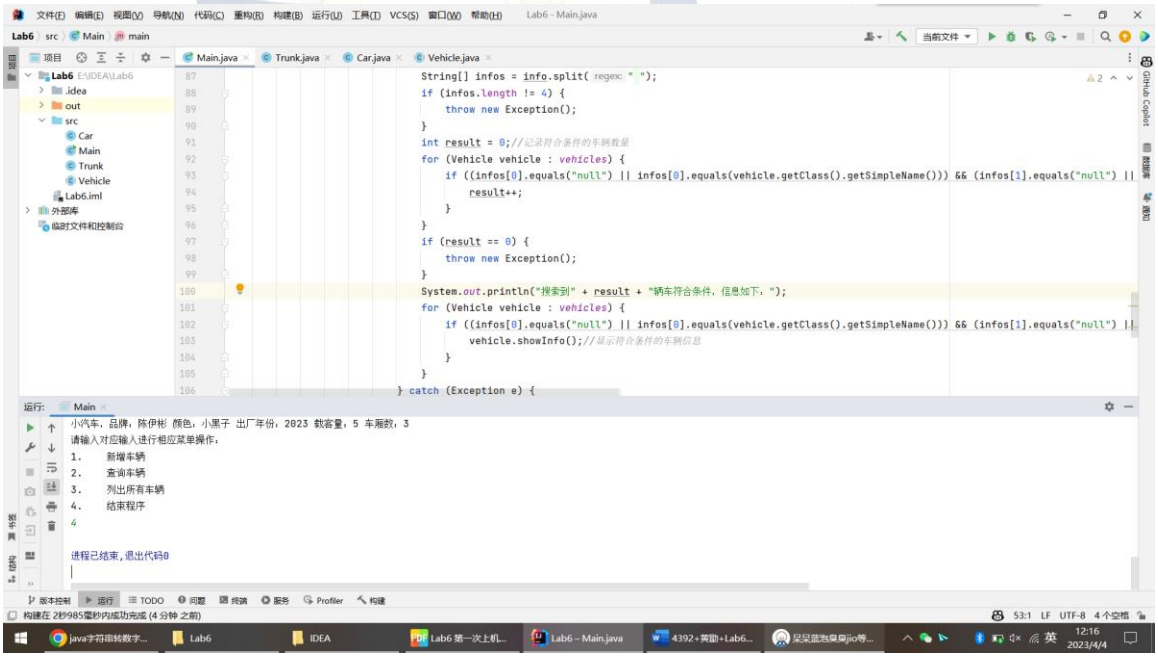
程序的主要逻辑如下：

1. 在 Main 类中定义了 Scanner 对象和 Vehicle 对象 ArrayList，用于获取用户输入和存储车辆信息。

2. 程序主菜单使用 do-while 循环实现，循环条件为 exit 变量，如果用户选择了“4. 结束程序”，则将 exit 设置为 true，程序结束。
3. 在每次循环中，使用 try-catch 语句捕获用户输入不合法的异常，并在 catch 块中输出提示信息。
4. 如果用户选择了“1. 添加车辆”，则进入添加车辆的菜单。使用 do-while 循环实现，循环条件为 true，用户输入 end 时结束循环。
5. 在添加车辆的菜单中，首先要求用户输入车辆信息，并使用 String 类的 split 方法将用户输入的字符串分割成字符串数组。
6. 根据数组长度判断用户输入的车辆类型是小汽车还是卡车，然后将输入的信息解析成相应的 Car 或 Trunk 对象，并将其添加到 vehicles 中。
7. 如果用户选择了“2. 查询车辆”，则进入查询车辆的菜单。同样使用 do-while 循环实现，循环条件为 true，用户输入 end 时结束循环。
8. 在查询车辆的菜单中，要求用户输入查询条件，并将条件解析成对应的字符串。然后遍历 vehicles，找出符合查询条件的车辆，并输出它们的信息。
9. 如果用户选择了“3. 列出所有车辆”，则输出 vehicles 中存储的所有车辆的信息。
10. 在每个菜单中，都有相应的异常处理和输入校验，以保证程序的稳定性和正确性。

（三） 过程截图

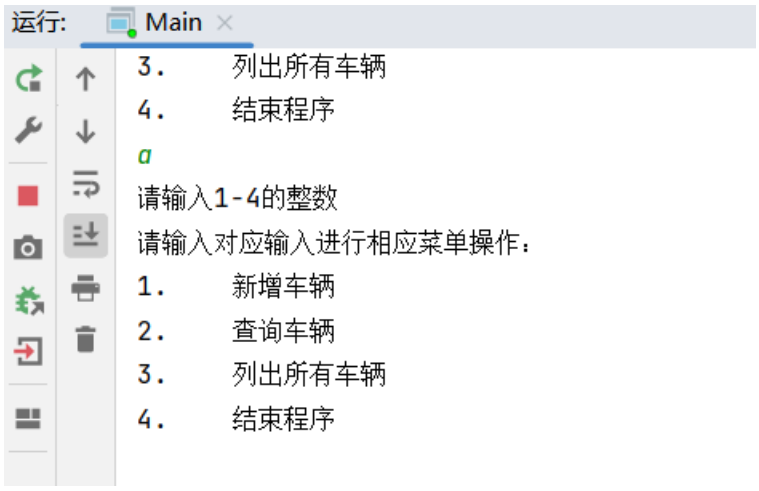
最终结果（全屏截图）

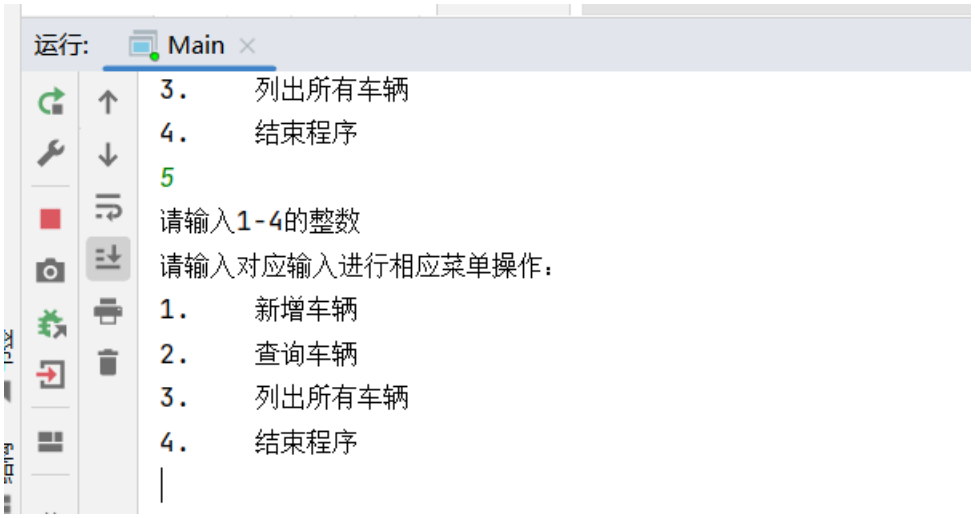


[1] 程序主菜单


```
9   public static void main(String[] args) {
10      boolean exit = false;
11      do {
12          try{
13              System.out.println("请输入对应输入进行相应菜单操作：");
14              System.out.println("1.    新增车辆");
15              System.out.println("2.    查询车辆");
16              System.out.println("3.    列出所有车辆");
17              System.out.println("4.    结束程序");
18              int choice = scanner.nextInt();
19              if(choice<1 || choice>4){
20                  throw new Exception();
21              }
```

[2] 主菜单错误处理





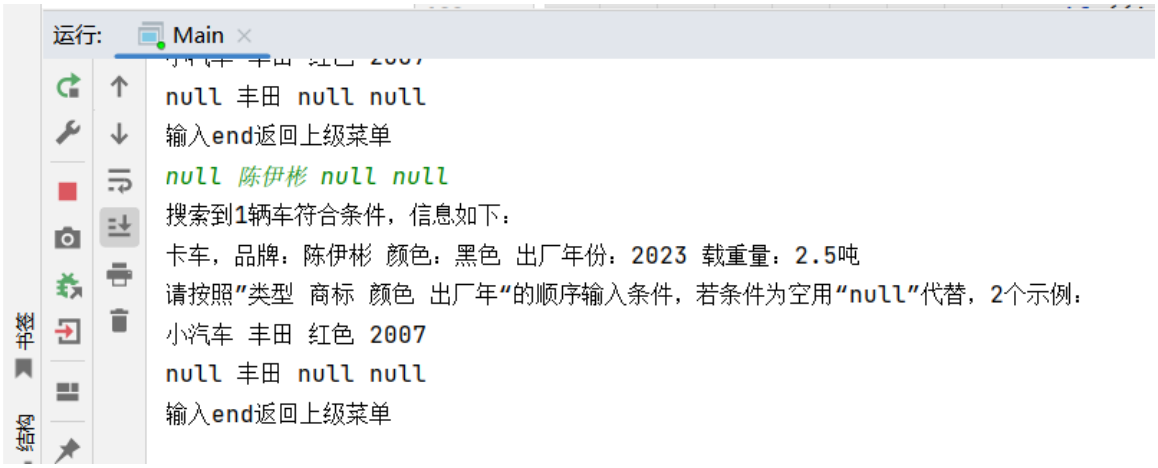

```
    }catch(Exception e){
        System.out.println("请输入1-4的整数");
        scanner.nextLine();
    }
}while(!exit);
```

[3] 新增车辆

Main.java Trunk.java Car.java Vehicle.java

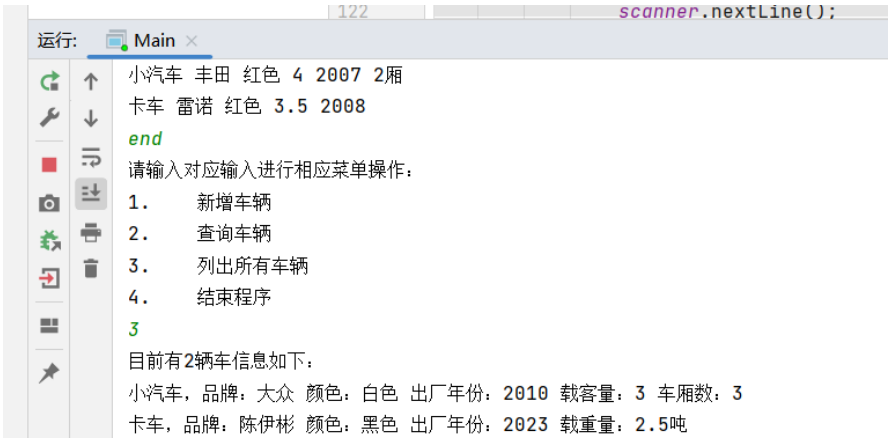
```
                switch (choice) {
                    case 1 -> {
                        do {
                            try {
                                System.out.println("请输入车辆信息，示例：");
                                System.out.println("小汽车 丰田 红色 4 2007 2厢");
                                System.out.println("卡车 雷诺 红色 3.5 2008");
                                String info = scanner.nextLine();
                                if (info.equals("")) {
                                    info = scanner.nextLine();
                                }
                                String[] infos = info.split( regex: " ");
                                //若用户输入end则结束新增，重新显示主菜单
                                if (infos[0].equals("end")) {
                                    break;
                                }
                                //检查用户输入的车辆信息是否符合要求
                                if (infos.length == 6) {
                                    if (!infos[0].equals("小汽车")) {
                                        System.out.println("第一个参数必须为小汽车或者卡车");
                                        throw new Exception();
                                    }
                                    if (!infos[5].equals("2厢") && !infos[5].equals("3厢")) {
                                        throw new Exception();
                                    }
                                    vehicles.add(car);
                                } else if (infos.length == 5) {
                                    if (!infos[0].equals("卡车")) {
                                        System.out.println("第一个参数必须为小汽车或者卡车");
                                        throw new Exception();
                                    }
                                    Trunk trunk = new Trunk(infos[1], infos[2], Integer.parseInt(infos[4]), Double.parseDouble(infos[3]));
                                    vehicles.add(trunk);
                                } else {
                                    throw new Exception();
                                }
                                System.out.println("创建成功");
                                vehicles.get(vehicles.size() - 1).showInfo();//显示创建的车辆信息
                                System.out.println();
                            } catch (Exception e) {
                                System.out.println("创建不成功");
                            }
                        } while (true);
```

[4] 查询车辆



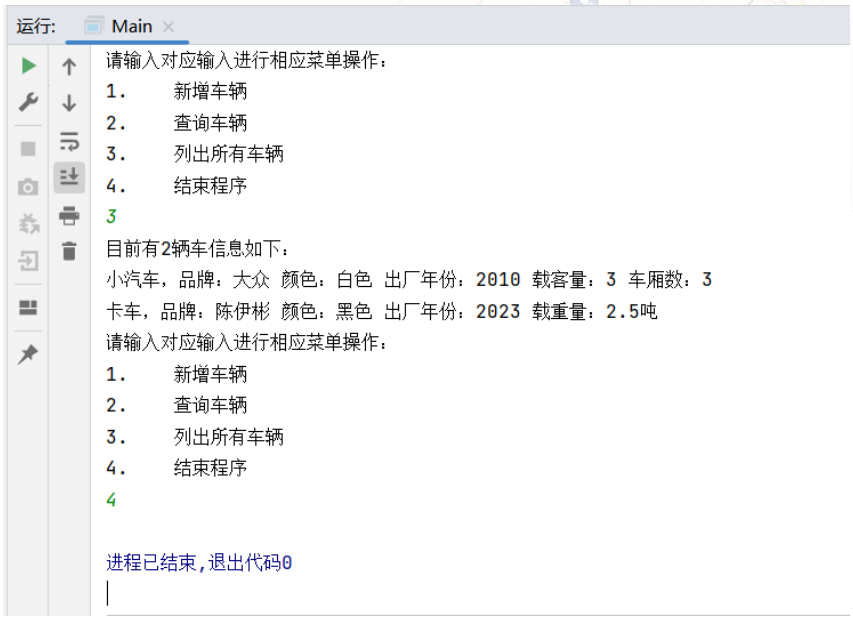

```
case 2 -> {
    do {
        try {
            System.out.println("请按照"类型 商标 颜色 出厂年"的顺序输入条件，若条件为空用"null"代替，2个示例：");
            System.out.println("小汽车 丰田 红色 2007");
            System.out.println("null 丰田 null null");
            System.out.println("输入end返回上级菜单");
            String info = scanner.nextLine();
            if (info.equals("")) {
                info = scanner.nextLine();
            }
            if (info.equals("end")) {
                break;
            }
            String[] infos = info.split( regex: " ");
            if (infos.length != 4) {
                throw new Exception();
            }
            int result = 0;//记录符合条件的车辆数量
            for (Vehicle vehicle : vehicles) {
                if ((infos[0].equals("null") || infos[0].equals(vehicle.getClass().getSimpleName())) && (infos[1].equals("null") ||
                    result++;
                }
            }
            if (result == 0) {
                throw new Exception();
            }
            System.out.println("搜索到" + result + "辆车符合条件，信息如下：");
            for (Vehicle vehicle : vehicles) {
                if ((infos[0].equals("null") || infos[0].equals(vehicle.getClass().getSimpleName())) && (infos[1].equals("null") ||
```

[5] 列出所有车辆




```
case 3 -> {
    System.out.println("目前有" + vehicles.size() + "辆车信息如下：");
    for (Vehicle vehicle : vehicles) {
        vehicle.showInfo();//利用多态，显示所有车辆信息
    }
}
```

[6] 结束程序




```
118                 case 4 -> exit = true;
```

三、实验总结与心得记录

Java 中的继承和多态是面向对象编程的重要概念，对于 Java 程序员来说是必须掌握的基础知识。继承是一种创建新类的方式，让已有类的属性和方法在新的类中得以重复使用。通过继承，可以实现代码的复用性和可维护性。Java 中的继承是单继承的，即每个类只能直接继承自一个父类。

在继承的基础上，Java 还提供了多态机制。多态是指同一种行为（方法）在不同的对象上有不同的表现形式。在 Java 中，多态可以通过方法重载和方法重写来实现。方法重载是指在同一个类中定义多个方法，方法名相同但参数不同，编译器会根据不同的参数类型和个数选择合适的方法。方法重写是指子类重写父类中的方法，使得子类可以使用自己的实现方式。

继承和多态是 Java 面向对象编程的内核概念，理解和掌握这些概念对于 Java 程序员来说非常重要。在实际编程中，应该尽可能地使用继承和多态，以提高代码的复用性和可维护性。总之，这个实验对我的 Java 编程技能和面向对象编程能力的提升非常有帮助。通过这个实验，我深入了解了 Java 的更多知识。这些技能和知识将在我的未来的 Java 开发中起到重要的作用。

Java 中的异常处理机制可以帮助我们有效地诊断和调试程序，以及提高程序的健壮性。在 Java 中，异常处理机制主要由以下几个方面组成：

1. 异常类：Java 中提供了多个异常类，包括 RuntimeException、IOException、SQLException 等。程序员也可以自己定义异常类。

2. 异常处理语句：Java 中提供了 try-catch 和 try-catch-finally 语句用于捕获和处理异常。try 块中的代码可能会抛出异常，catch 块中的代码用于处理异常。

3. 抛出异常：在 Java 中，程序员可以使用 throw 关键字抛出异常。如果一个方法可能抛出异常，就需要在方法签名中声明 throws 子句。

4. 异常链：Java 中的异常可以形成链式结构。一个异常对象可能包含另一个异常对象，这样就可以更好地追踪异常的来源和原因。

在实际编程中，我们应该注意以下几点：

1. 对于可预见的异常情况，我们应该尽可能地通过 try-catch 语句来处理异常，以保证程序的正常运行。

2. 在处理异常时，应该注意将异常信息打印出来，以便诊断问题。

3. 对于不可预见的异常情况，应该在程序中使用 assert 语句或编写单元测试，以保证程序的健壮性。

4. 在自定义异常类时，应该遵循 Java 的命名规范，并且要确保异常类具有清晰的语义。

综上所述，Java 中的异常处理机制是 Java 程序设计中非常重要的一部分。我们应该熟练掌握 Java 中的异常类和异常处理语句，并且在实际编程中注重异常处理，以保证程序的正常运行和健壮性。