北京邮电大学计算机学院

软件工程专业实验报告

课程名称： Java 程 序 设 计

# 项目名称： 实验一：Java程序编制（基础练习）

项目完成人：

姓名： 禄禄鱼

学号：

指导教师： 崔毅东

日 期： 2023 年 10 月 25 日

（正文文字，请使用【正文缩进】样式）

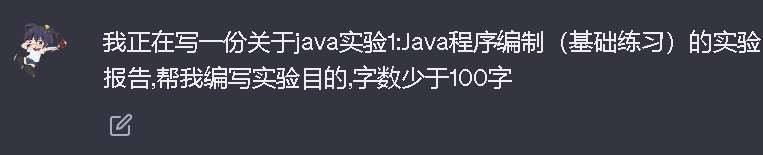
（标题编号请使用自动编号。在样式表中选中标题级别样式即可自动编号。）

（蓝色斜体字在提交之前应该删除。）

# 实验概述

## 实验目的

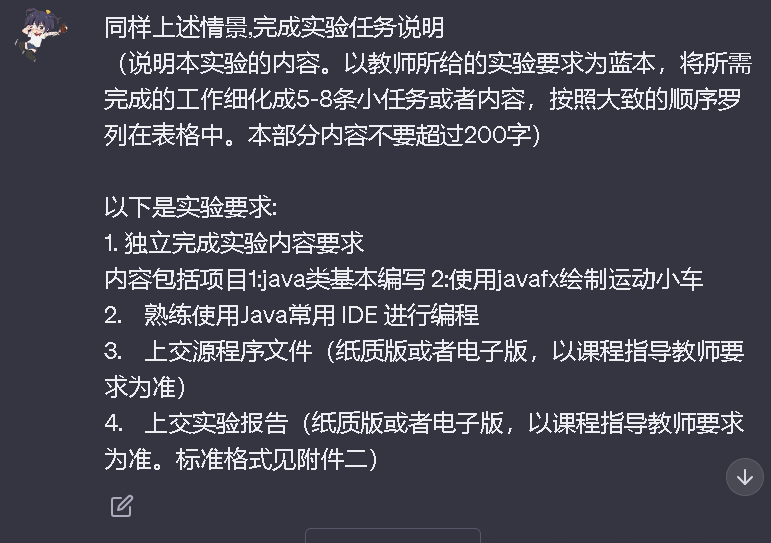
（说明通过本实验希望达到的目的。要根据你自己的理解撰写。例如，学习使用某种工具、学习某种方法、理解某种概念、锻炼某种实践能力等。一般不超过100个字）



实验目的：通过完成基础练习，旨在培养我编写Java程序的基本能力，理解Java语言的语法和逻辑结构，掌握变量、控制结构、函数等基本概念，以便能够编写简单的Java程序解决实际问题。

## 实验任务

（说明本实验的内容。以教师所给的实验要求为蓝本，将所需完成的工作细化成5-8条小任务或者内容，按照大致的顺序罗列在下方表格中之。注意，不能照抄教师所给实验要求。本部分内容不要超过200字）



|  |  |
| --- | --- |
|  | 任务说明 |
|  | **Java类基本编写,**编写Java类，包括类的定义、属性、方法。 |
|  | **使用JavaFX绘制运动小车**学习使用JavaFX库进行图形绘制。创建一个JavaFX应用程序，绘制运动小车的图形 |
|  | 熟悉IDE的基本功能，如创建项目、编辑代码、编译和运行程序。 |
|  | **源程序文件提交**准备项目1和项目2的源程序文件 |
|  | 撰写实验报告，按照指导教师提供的标准格式（附件二）进行文档撰写 |
|  | 在报告中总结实验成果、编程过程和遇到的问题 |
|  |  |

## 实验环境

（说明本实验需要的环境）

Windows 10

JetBrain Intellij

### 硬件环境

（说明本实验所需的硬件环境，例如计算机型号、配置等）

Lenovo ThinkPad X13 Gen2

Intel 11代i5  
英特尔核显

### 网络环境

（说明本实验所需的网络环境，例如网络设备型号、配置等）

（如果实验中搭建了网络，或者网络配置是实验中的重要部分，请说明整个实验外部网络环境的状况。）

BUPT校园网

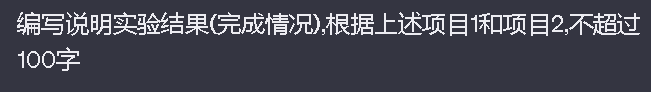
### 软件环境

（说明本实验所用的软件环境。例如操作系统类型、版本，所用的软件名称、版本等。此处无需说明软件如何配置。软件配置应作为实验方案的一部分放到第二章中介绍）

OS:Windows 10

IDE:JetBrain Intellij

## 实验结果



（简要说明实验完成情况，本部分 不超过100字）

实验结果说明：

项目1：成功编写了Java类，包括类定义、属性和方法，实现了基本的Java程序。

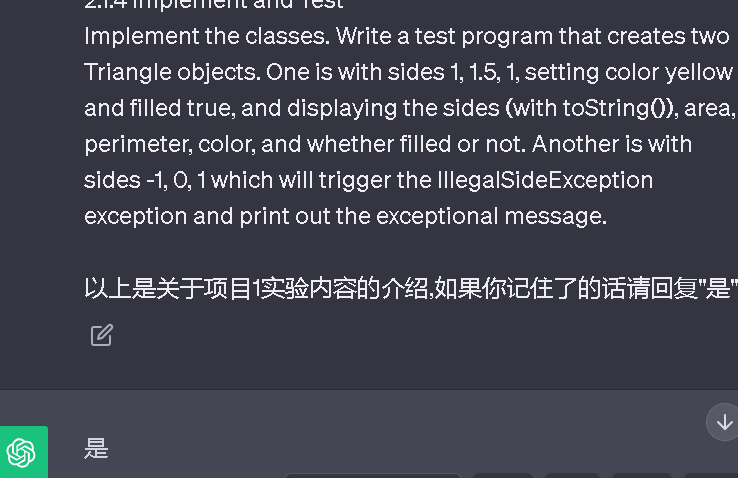
项目2：成功使用JavaFX库绘制了一个运动小车的图形，实现了JavaFX应用程序。

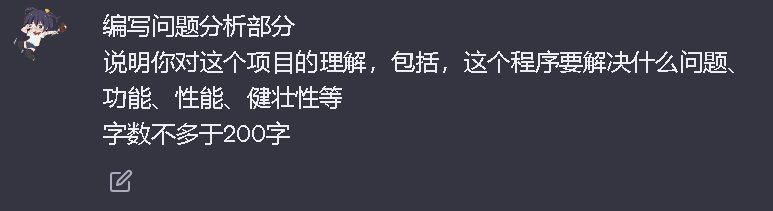
实验进展顺利，所有任务按要求完成。通过实验，我掌握了Java编程基础和图形绘制，提高了编程技能。源代码和实验报告已按规定提交。

# 项目1

（附上实验文档，如：问题分析、算法、UML类图、程序、仿真结果、运行结果、调试心得等，具体内容根据实验要求来定。源代码请附在这里。源代码排版请特别注意，用5 号字体，行间距为单倍行距。注意节省空间，不要浪费纸张。）

## 问题分析





（说明你对这个问题的理解，包括，这个程序要解决什么问题、功能、性能、健壮性等）

1. **问题描述**：
   * 首先，问题是要设计一个程序来处理三角形的相关信息，包括边长、面积、周长、颜色等。
   * 其次，要处理非法边长情况，当提供的边长小于等于0时，需要引发异常。
2. **功能**：
   * **Triangle**类提供了表示三角形的基本信息和计算功能。
   * **IllegalSideException**类用于处理非法边长情况。
3. **性能**：
   * 该程序的性能主要取决于三角形对象的创建和数据处理，这在通常情况下不会引起性能问题。
   * 异常处理可能会引入一些性能开销，但在这种情况下，性能不是主要关注点。
4. **健壮性**：
   * 为了保证程序的健壮性，需要处理非法边长情况，确保程序能够处理并报告这些情况，而不会崩溃。
   * 需要适当的异常处理来捕获非法输入，以确保程序可以继续运行并提供有关错误的信息。

总之，这个程序旨在解决三角形相关信息的计算和异常处理问题。它需要具备良好的健壮性，确保程序可以处理不合法的输入，并能够正确地计算和显示三角形的属性。

### 目标

（说明这个程序要解决什么问题，达到的目标）

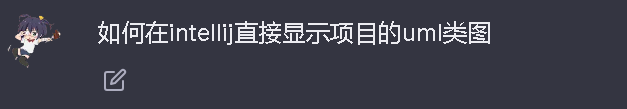


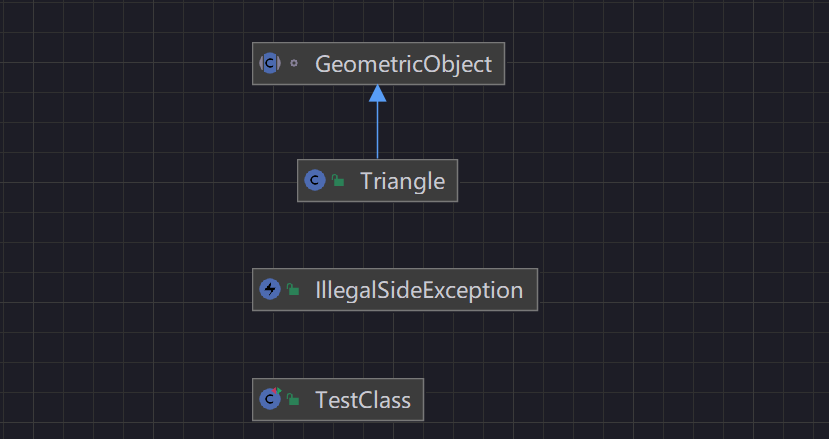
**目标**：

* 计算和展示给定三角形的属性，包括面积和周长。
* 提供可视化描述三角形的字符串表示，包括边长、颜色和填充状态。
* 当提供的三角形边长不合法时，引发异常并提供异常消息。
* 通过使用继承和异常处理，实现面向对象的程序设计

## 设计方案

### 类图

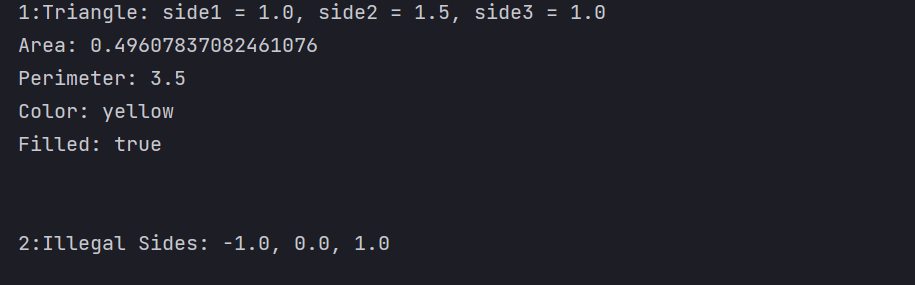




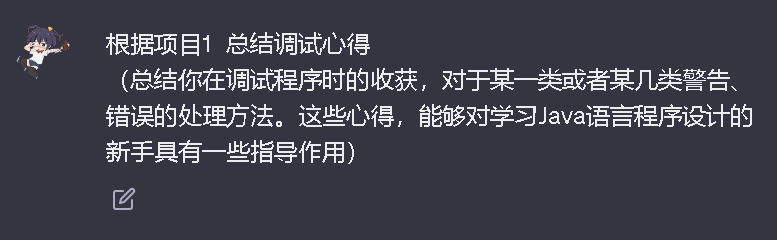
## 实验结果

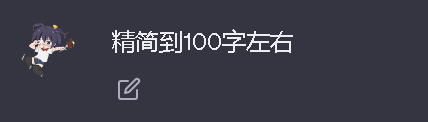
（说明实验完成情况，有需要文字回答的题目，以及实验结果截图，请写在这里）

（在此处，详细描述运行的结果。包括：不同输入对应的不同结果，异常情况的执行结果）



## 调试心得





（总结你在调试程序时的收获，对于某一类或者某几类警告、错误的处理方法。这些心得，能够对学习Java语言程序设计的新手具有一些指导作用）

调试心得：理解编译器警告，使用调试器，逐行检查代码，添加注释和文档，分解问题，编写测试用例，学习错误信息，查找资源，保持耐心，学习他人代码。调试是编程不可或缺的一部分，它提高技能。不断积累调试经验，变得更熟练和自信

## 源程序



package school.expr01**;**import java.util.ArrayList**;**import org.junit.Test**;**//定义GeometricObject类  
abstract class GeometricObject {  
  
 protected String color**;** protected boolean filled**;** protected ArrayList<Double> sides**;** public String getColor() {  
 return color**;** }  
  
 public boolean isFilled() {  
 return filled**;** }  
 // 无参构造方法，创建默认的GeometricObject对象  
 protected GeometricObject(String color**,** boolean filled) {  
 this.color = color**;** this.filled = filled**;** }  
 // 抽象方法，计算GeometricObject对象的面积  
 public abstract double getArea()**;** // 抽象方法，计算GeometricObject对象的周长  
 public abstract double getPerimeter()**;** @Override  
 public String toString() {  
 return "GeometricObject: sides = " + sides**;** }  
}  
  
public class Triangle extends GeometricObject {  
  
 private double side1**;** private double side2**;** private double side3**;** // 无参构造方法，创建默认的三角形  
 public Triangle() {  
 super("white"**,** false)**;** sides = new ArrayList<>()**;** side1 = 1.0**;** side2 = 1.0**;** side3 = 1.0**;** sides.add(side1)**;** sides.add(side2)**;** sides.add(side3)**;** }  
  
 // 构造方法，创建具有指定边长的三角形  
 public Triangle(double side1**,** double side2**,** double side3**,** String color**,** boolean filled) throws IllegalSideException {  
 super(color**,** filled)**;** if(side1 <= 0 || side2 <= 0 || side3 <= 0) {  
  
 throw new IllegalSideException(side1**,** side2**,** side3)**;** }  
 sides = new ArrayList<>()**;** this.side1 = side1**;** this.side2 = side2**;** this.side3 = side3**;** }  
  
 // 获取所有边长的getter方法  
 public ArrayList<Double> getSides() {  
 return sides**;** }  
  
 //获取第n条边的get方法,如果n不是1,2,3则返回-1  
 public double getSide(int n) {  
 if (n >= 1 && n <= 3) {  
 return sides.get(n - 1)**;** } else {  
 System.*out*.println("n不是1,2,3")**;** return -1**;** }  
 }  
 // 计算三角形的面积  
 @Override  
 public double getArea() {  
 double s = (side1 + side2 + side3) / 2**;** return Math.*sqrt*(s \* (s - side1) \* (s - side2) \* (s - side3))**;** }  
  
 // 计算三角形的周长  
 @Override  
 public double getPerimeter() {  
 return side1 + side2 + side3**;** }  
  
 // 返回三角形的字符串描述  
 @Override  
 public String toString() {  
 return "Triangle: side1 = " + side1 + ", side2 = " + side2 + ", side3 = " + side3**;** }  
  
}



package school.expr01**;**public class IllegalSideException extends Exception {  
 private double side1**;** private double side2**;** private double side3**;** // No-arg constructor with a default message  
 public IllegalSideException() {  
 super("Illegal Sides")**;** }  
  
 // Constructor with custom message  
 public IllegalSideException(double side1**,** double side2**,** double side3) {  
 super("Illegal Sides: " + side1 + ", " + side2 + ", " + side3)**;** this.side1 = side1**;** this.side2 = side2**;** this.side3 = side3**;** }  
  
 // Override getMessage() method to provide a customized message  
 @Override  
 public String getMessage() {  
 return super.getMessage()**;** }  
  
 public double getSide1() {  
 return side1**;** }  
  
 public double getSide2() {  
 return side2**;** }  
  
 public double getSide3() {  
 return side3**;** }  
}



package school.expr01**;**import org.junit.Test**;**public class TestClass  
{  
 @Test  
 public void test(){  
 //Implement the classes. Write a test program that creates two Triangle objects. One is with sides 1, 1.5, 1, setting color yellow and filled true, and displaying the sides (with toString()), area, perimeter, color, and whether filled or not. Another is with sides -1, 0, 1 which will trigger the IllegalSideException exception and print out the exceptional message.  
  
  
 try {  
 Triangle triangle1 = new Triangle(1**,** 1.5**,** 1**,** "yellow"**,** true)**;** System.*out*.println("1:"+ triangle1)**;** System.*out*.println("Area: " + triangle1.getArea())**;** System.*out*.println("Perimeter: " + triangle1.getPerimeter())**;** System.*out*.println("Color: " + triangle1.getColor())**;** System.*out*.println("Filled: " + triangle1.isFilled())**;** } catch (IllegalSideException e) {  
 System.*out*.println(e.getMessage())**;** }  
 System.*out*.println("\n")**;** try {  
 Triangle triangle2 = new Triangle(-1**,** 0**,** 1**,** "yellow"**,** true)**;** System.*out*.println(triangle2)**;** System.*out*.println("Area: " + triangle2.getArea())**;** System.*out*.println("Perimeter: " + triangle2.getPerimeter())**;** System.*out*.println("Color: " + triangle2.getColor())**;** System.*out*.println("Filled: " + triangle2.isFilled())**;** } catch (IllegalSideException e) {  
 System.*out*.println("2:"+e.getMessage())**;** }  
 }  
  
}

# 项目2

## 问题分析

### 目标

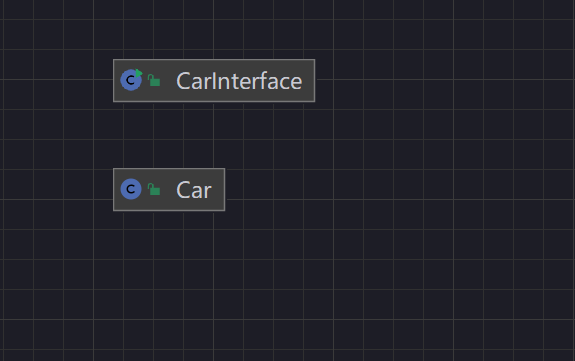
（说明这个程序要解决什么问题，达到的目标）

1. 设计用户界面，包括组件、容器和布局管理器。
2. 创建一个名为**Car**的类，用于表示赛车，包括数据字段、构造函数和方法。
3. 实现**Car**类，确保它能够模拟赛车的运动。
4. 实现主应用程序类，将用户界面和**Car**类结合起来，实现模拟赛车运动的功能。
5. 提交源代码和应用程序运行的截图，以展示实现的GUI应用程序。

## 设计方案

（在此处，用文字描述整体的设计，包括，分成多少个模块（Java package），多少个类。每个模块、类负责的大致功能； 此外，还需要包含一个程序整体的流程图，反映各个模块和类之间的关系；但是避免绘制过于详细的流程）

### 类图



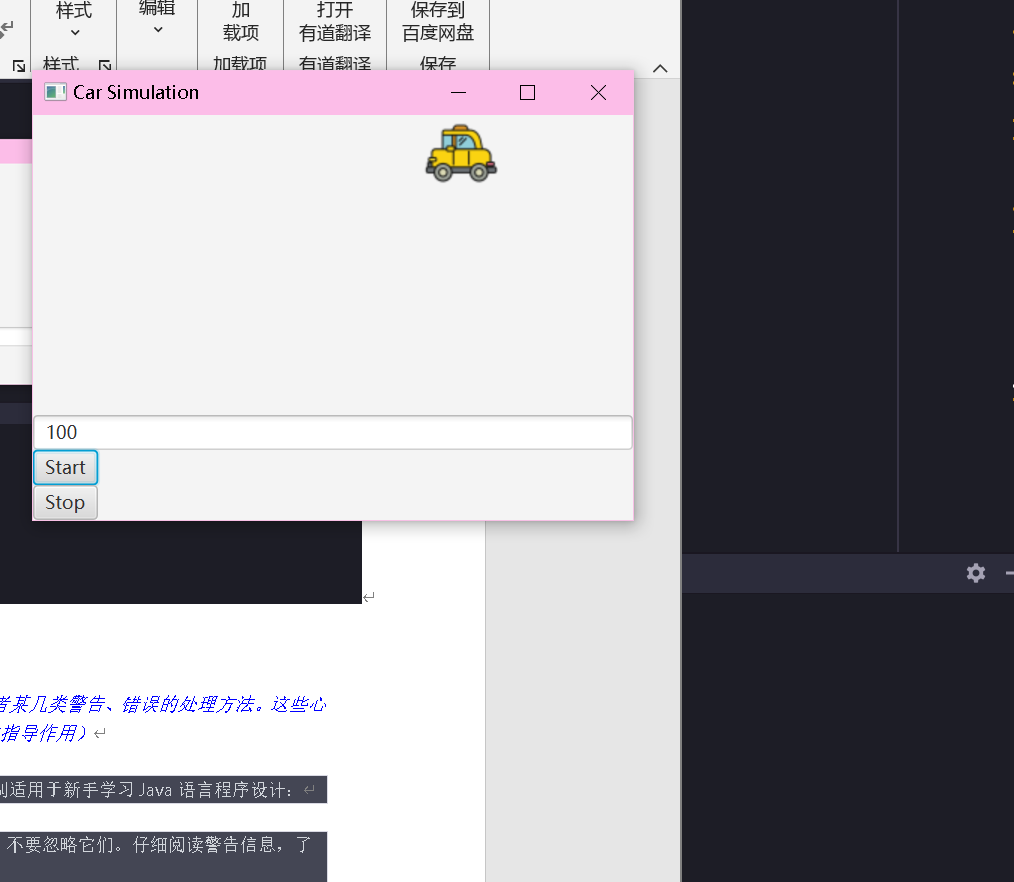
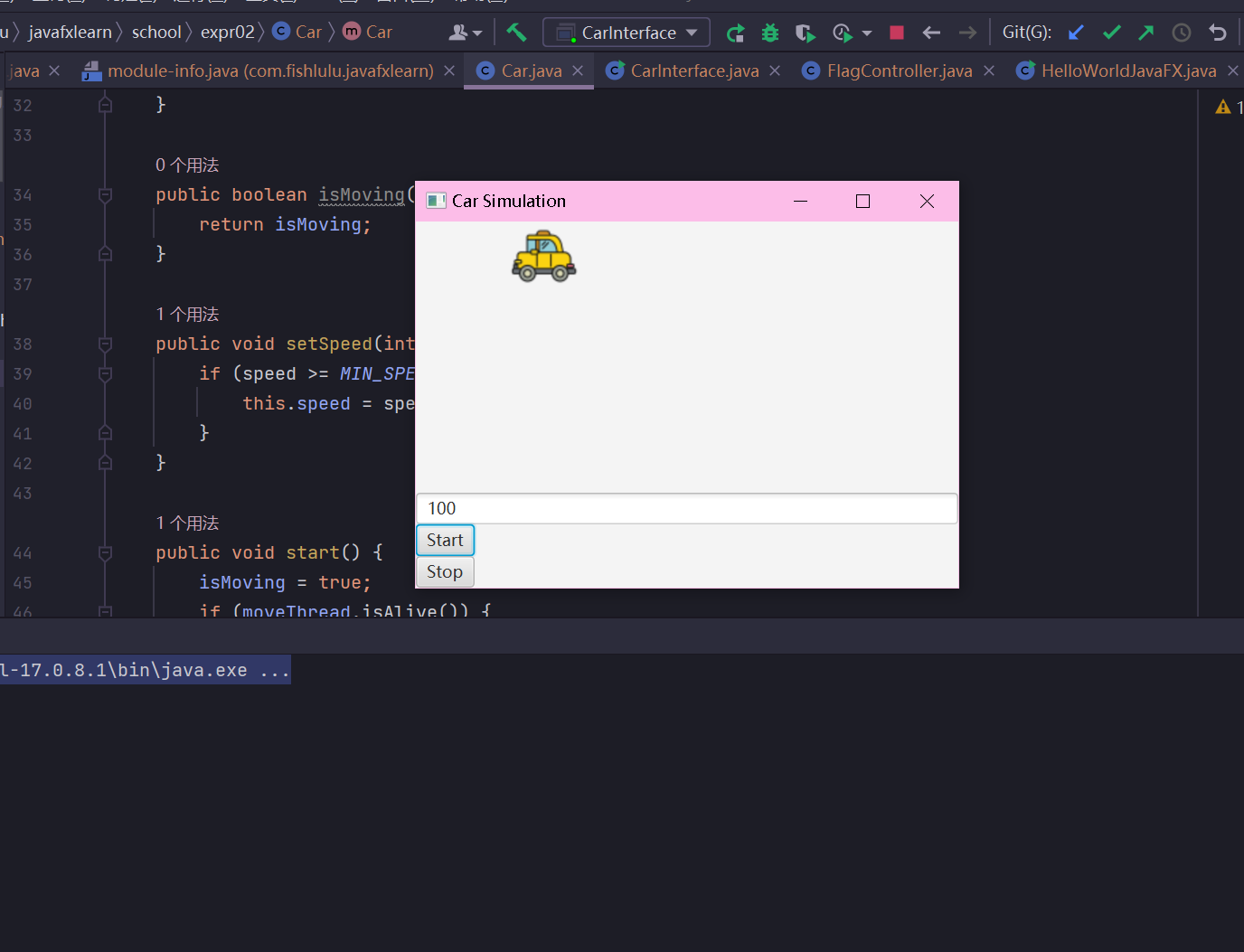
## 实验结果

（说明实验完成情况，有需要文字回答的题目，以及实验结果截图，请写在这里）

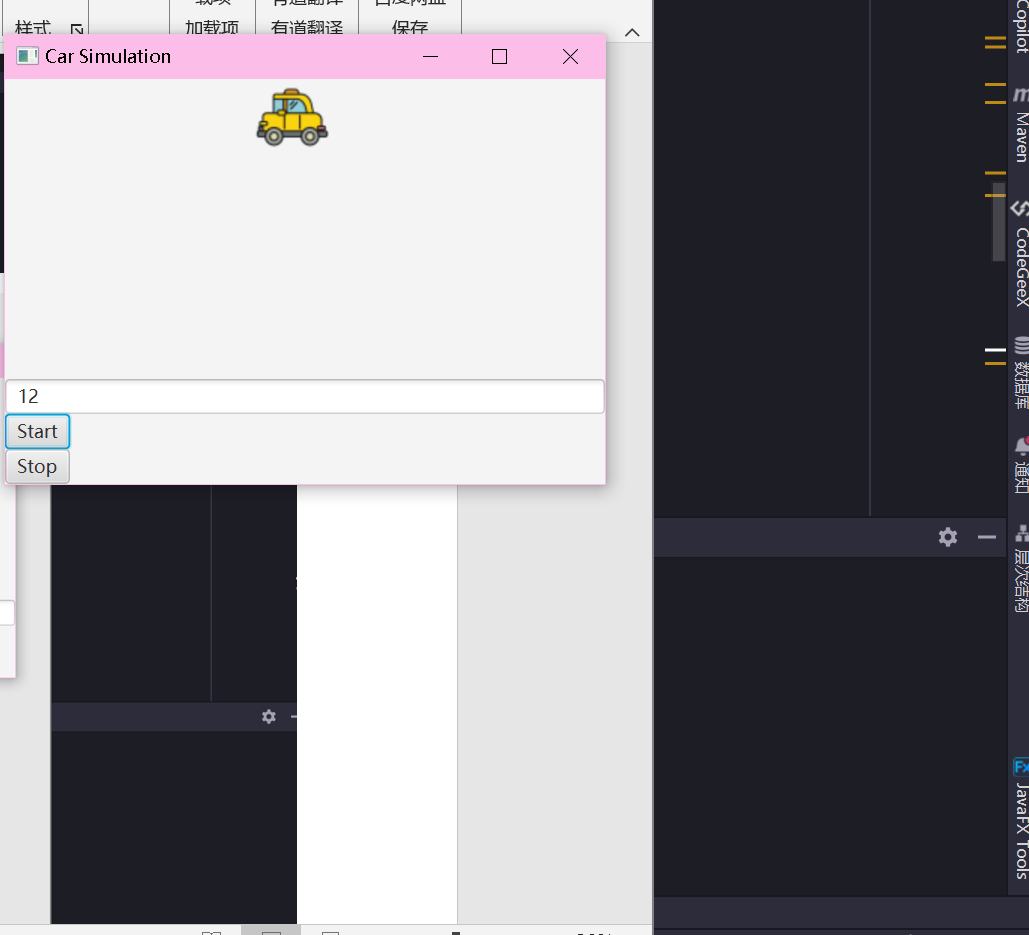
（在此处，详细描述运行的结果。包括：不同输入对应的不同结果，异常情况的执行结果）

Start 小车开始运动,stop小车停止运动,输入框可以输入小车运动的速度,如果输入非法数字或者不在1-100的speed数值,小车将默认以1的速度前进

当小车运动到外面的时候会自动返回到起点



慢速移动



## 调试心得

（总结你在调试程序时的收获，对于某一类或者某几类警告、错误的处理方法。这些心得，能够对学习Java语言程序设计的新手具有一些指导作用）

在进行项目2的调试过程中，我积累了一些经验，特别适用于新手学习Java语言程序设计：

1. **理解编译器警告**：编译器警告是重要的线索，不要忽略它们。仔细阅读警告信息，了解它们的含义，然后采取适当措施来解决。
2. **模块化代码**：将代码模块化，即将功能拆分为小模块。这样，当出现问题时，可以更容易定位和解决。
3. **单元测试**：编写并运行小规模的单元测试。通过逐一测试代码的不同部分，可以发现问题并排除错误。
4. **调试工具**：充分利用集成开发环境（IDE）的调试工具。学会使用断点、单步执行和变量监视，这些工具可以帮助你更好地理解代码的执行过程。
5. **打印调试信息**：在代码中添加调试语句，输出变量值和中间结果。这对于理解程序的执行流程和检查变量状态非常有帮助。
6. **理解异常**：学会理解异常的类型和含义。异常消息通常提供了出错的线索，所以要仔细查看异常信息。
7. **查找资源**：在遇到问题时，不要犹豫去查找资源，包括官方文档、编程论坛和在线教程。很多问题都有解决方案。
8. **持之以恒**：调试可能是耗时的，但要保持耐心。每个错误和问题都是学习的机会，不要害怕失败。

综合而言，调试是编程中不可或缺的一部分，它可以提高你的问题解决能力。不断积累调试经验，你将更加熟练和自信

## 源程序

界面调用端

package com.fishlulu.javafxlearn.school.expr02**;**import javafx.application.Application**;**import javafx.scene.Scene**;**import javafx.scene.canvas.Canvas**;**import javafx.scene.canvas.GraphicsContext**;**import javafx.scene.control.Button**;**import javafx.scene.control.TextField**;**import javafx.scene.image.Image**;**import javafx.scene.layout.VBox**;**import javafx.stage.Stage**;**public class CarInterface extends Application {  
  
 public static void main(String[] args) {  
 *launch*(args)**;** }  
  
 @Override  
 public void start(Stage primaryStage) {  
 final int MIN\_SPEED = 1**;** final int MAX\_SPEED = 100**;** Canvas canvas = new Canvas(400**,** 200)**;** GraphicsContext gc = canvas.getGraphicsContext2D()**;** //传入图片,并缩小到25%大小  
 Image carImage = new Image("file:D:\\Pictures\\图标\\car.png"**,** canvas.getWidth() / 4**,** canvas.getHeight() / 4**,** true**,** true)**;** Car car = new Car(carImage**,** gc**,** canvas.getWidth() - carImage.getWidth()**,** canvas.getHeight() - carImage.getHeight())**;** TextField speedInput = new TextField()**;** Button startButton = new Button("Start")**;** Button stopButton = new Button("Stop")**;** startButton.setOnAction(e -> {  
 int speed**;** try {  
 speed = Integer.*parseInt*(speedInput.getText())**;** if (speed < MIN\_SPEED || speed > MAX\_SPEED) {  
 speed = MIN\_SPEED**;** speedInput.setText(String.*valueOf*(speed))**;** }  
 } catch (NumberFormatException ex) {  
 speed = MIN\_SPEED**;** speedInput.setText(String.*valueOf*(speed))**;** }  
 car.setSpeed(speed)**;** car.start()**;** })**;** stopButton.setOnAction(e -> car.stop())**;** VBox layout = new VBox()**;** layout.getChildren().addAll(canvas**,** speedInput**,** startButton**,** stopButton)**;** Scene scene = new Scene(layout)**;** primaryStage.setScene(scene)**;** car.draw()**;** primaryStage.setTitle("Car Simulation")**;** primaryStage.show()**;** }  
}

Car类

package com.fishlulu.javafxlearn.school.expr02**;**import javafx.scene.canvas.GraphicsContext**;**import javafx.scene.image.Image**;**public class Car {  
 private static final int *MIN\_SPEED* = 1**;** private static final int *MAX\_SPEED* = 100**;** private static final int *SPEED\_STEP* = 1**;** private GraphicsContext gc**;** private Image carImage**;** private int speed = *MIN\_SPEED***;** private boolean isMoving = false**;** private double posX = 0**;** private double posY = 0**;** private double maxX = 0**;** private double maxY = 0**;** private Thread moveThread**;** // 线程对象  
  
 public Car(Image carImage**,** GraphicsContext gc**,** double maxX**,** double maxY) {  
 this.gc = gc**;** this.carImage = carImage**;** this.maxX = maxX**;** this.maxY = maxY**;** // 初始化线程  
 moveThread = new Thread(this::move)**;** }  
  
 public int getSpeed() {  
 return speed**;** }  
  
 public boolean isMoving() {  
 return isMoving**;** }  
  
 public void setSpeed(int speed) {  
 if (speed >= *MIN\_SPEED* && speed <= *MAX\_SPEED*) {  
 this.speed = speed**;** }  
 }  
  
 public void start() {  
 isMoving = true**;** if (moveThread.isAlive()) {  
 // 如果线程仍在运行，则不重复启动  
 return**;** }  
 // 启动线程  
 moveThread = new Thread(this::move)**;** moveThread.start()**;** }  
  
 public void stop() {  
 isMoving = false**;** // 中断线程  
 if (moveThread.isAlive()) {  
 moveThread.interrupt()**;** }  
 }  
  
 private void move() {  
 while (isMoving) {  
 try {  
 Thread.*sleep*(1000 / speed)**;** // 擦除旧位置  
 clearOldPosition()**;** // 更新小车位置  
 posX += *SPEED\_STEP***;** if (posX >= maxX) {  
 posX = posX - maxX**;** }  
 // 绘制新位置  
 draw()**;** } catch (InterruptedException e) {  
 // 检测线程中断，以便停止移动  
 return**;** }  
 }  
 }  
  
 public void draw() {  
 gc.drawImage(carImage**,** posX**,** posY)**;** }  
  
 private void clearOldPosition() {  
 gc.clearRect(posX**,** posY**,** carImage.getWidth()**,** carImage.getHeight())**;** }  
}