|  |  |
| --- | --- |
| 成绩 |  |



**课程设计报告**



**题 目 Java图形处理设计**

**课 程 名 称 面向对象程序设计课程设计**

**院 部 名 称**  **软件工程学院**

**专 业** **软件工程**

**班 级 23软件工程3班**

**学 生 姓 名 周大钦**

**学 号 2312001064**

**课程设计地点 科技楼2-414**

**课程设计学时 20**

**指 导 教 师 龚如宾**

金陵科技学院教务处制

目录

[1绪论 2](#_Toc528268720)

[1.1系统设计目的 2](#_Toc528268721)

[2系统分析 3](#_Toc528268722)

[2.1 问题描述和求解 3](#_Toc528268723)

[2.2 问题求解流程 4](#_Toc528268724)

[2.3环境需求 4](#_Toc528268725)

[3系统设计 5](#_Toc528268727)

[3.1系统总体设计 5](#_Toc528268728)

[3.2概要设计 5](#_Toc528268729)

[3.3类的继承关系 5](#_Toc528268730)

[4系统实现 6](#_Toc528268731)

[4.1程序设计语言 6](#_Toc528268732)

[4.2编码 6](#_Toc528268733)

[4.3测试 15](#_Toc528268734)

[4.4实验结果 18](#_Toc528268735)

[5总结 19](#_Toc528268731)

[5.1基于OBE模式的学生自我评价与体会 20](#_Toc528268732)

## 一、课程设计目的

通过学习本课程设计的训练，综合应用面向对象程序设计方法的特性及Java语言的基本语法和类库，完成小型的控制台应用程序分析设计与实现，熟练使用Java程序开发平台Eclipse完成Java程序的编译、调试和执行。

## 二、课程设计任务

主要实现了一个图形处理和显示系统，包括图形的创建、计算面积和周长、按属性排序、文件输入输出以及图形的可视化。用户可以通过文本文件定义不同的几何图形，系统会解析这些数据并生成相应的图形对象，最终在图形界面中展示这些图形，并输出计算结果。这种设计有助于学习图形算法和面向对象编程的基本原则。

## 三、系统分析

3.1技术原理

3.1用户需求

普通用户

功能描述：

能够通过文本文件输入不同类型的几何图形（如三角形、圆形、矩形等）。

能够查看每个图形的属性（面积、周长）。

能够按照面积和周长对图形进行排序，并查看排序结果。

能够在图形界面中查看绘制的图形。

开发者

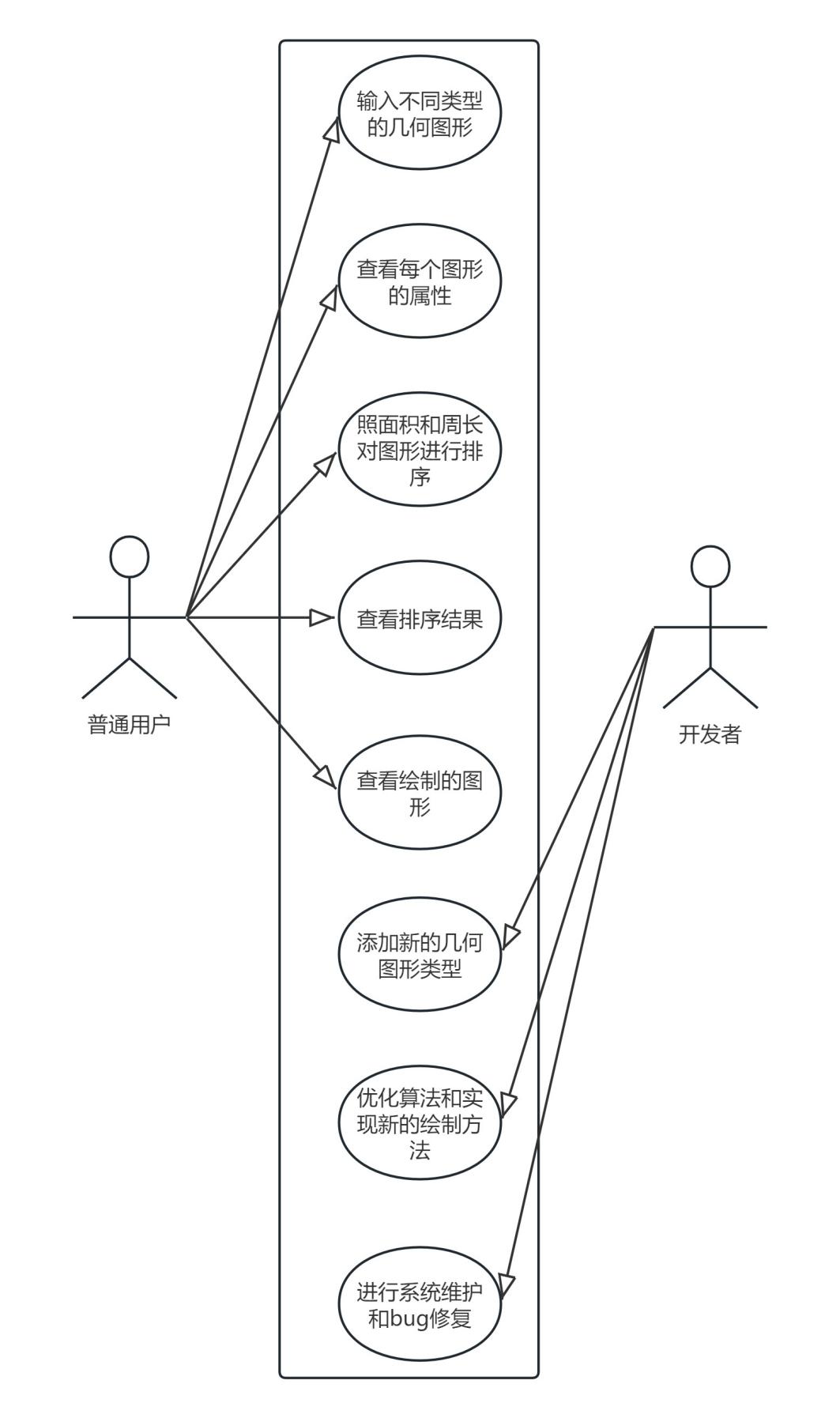
功能描述：

能够扩展系统，添加新的几何图形类型。

能够优化算法和实现新的绘制方法。

能够进行系统维护和bug修复。

3.2用例分析

3.2用例分析

3.3 用例功能描述

用例1：输入不同类型的几何图形

前置条件：用户已经准备好包含几何图形数据的文件。

基本流程：

用户通过界面选择或上传包含几何图形数据的文件。

系统读取文件内容并解析出几何图形的类型和参数。

系统根据解析的数据创建相应的几何图形对象。

后置条件：系统中成功添加了新的几何图形对象。

用例2：查看每个图形的属性

前置条件：系统中已经存在至少一个几何图形对象。

基本流程：

用户请求查看某个或所有几何图形的属性。

系统计算并显示每个几何图形的面积和周长。

后置条件：用户成功查看到几何图形的属性信息。

用例3：按面积和周长对图形进行排序

前置条件：系统中已经存在至少一个几何图形对象。

基本流程：

用户选择排序方式（按面积或周长）。

系统根据用户选择的排序方式对几何图形进行排序。

系统显示排序后的图形列表。

后置条件：用户成功看到按指定属性排序后的图形列表。

用例4：查看排序结果

前置条件：已经完成了图形的排序操作。

基本流程：

用户请求查看排序结果。

系统显示排序后的图形列表。

后置条件：用户成功看到排序后的图形列表。

用例5：查看绘制的图形

前置条件：系统中已经存在至少一个几何图形对象。

基本流程：

用户请求查看几何图形的绘制结果。

系统在图形界面中绘制出所有几何图形。

后置条件：用户成功看到绘制的几何图形。

用例6：添加新的几何图形类型

前置条件：开发者需要扩展系统功能。

基本流程：

开发者定义新的几何图形类并实现其属性和方法。

系统集成新图形类，使其可以被创建和使用。

后置条件：新几何图形类型成功添加到系统中。

用例7：优化算法和实现新的绘制方法

前置条件：开发者需要提高系统性能或增加新功能。

基本流程：

开发者分析现有算法并设计优化方案。

开发者实现新的绘制方法或优化现有算法。

系统更新以包含新的算法和绘制方法。

后置条件：系统性能得到提升，新绘制方法成功实现。

用例8：进行系统维护和bug修复

前置条件：系统存在已知的bug或需要维护。

基本流程：

开发者识别系统中的问题或bug。

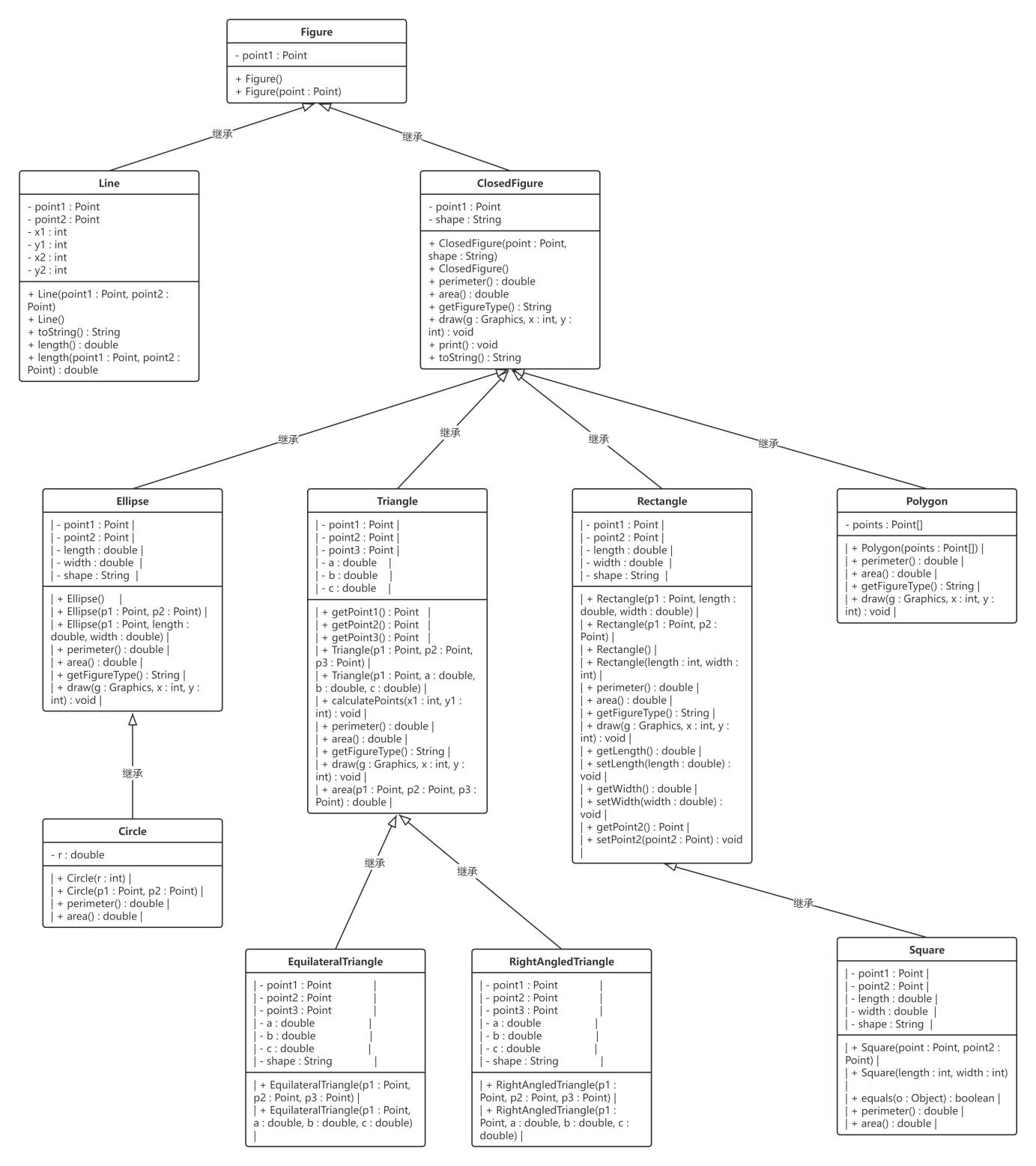
开发者进行代码修复或系统维护。

系统更新以解决这些问题。

后置条件：系统bug被修复，系统稳定性和性能得到提升。

## 四、系统设计

4.1 类设计



Figure（抽象类）

属性：Point point1

方法：

abstract double area()

abstract double perimeter()

abstract void draw(Graphics g, int x, int y)

ClosedFigure（抽象类，继承自Figure）

属性：String shape

方法：

abstract String getFigureType()

Triangle（继承自ClosedFigure）

属性：Point point2, Point point3, double a, b, c

方法：

double perimeter()

double area()

void draw(Graphics g, int x, int y)

Rectangle（继承自ClosedFigure

属性：double length, width

方法：

double perimeter()

double area()

void draw(Graphics g, int x, int y)

Circle（继承自Ellipse）

属性：double r

方法：

double perimeter()

double area()

void draw(Graphics g, int x, int y)

DrawingPanel

属性：ArrayList<ClosedFigure> figures

方法：

void paintComponent(Graphics g)

PerimeterComparator / AreaComparator

方法：

int compare(ClosedFigure o1, ClosedFigure o2)

关系

ClosedFigure 继承自 Figure

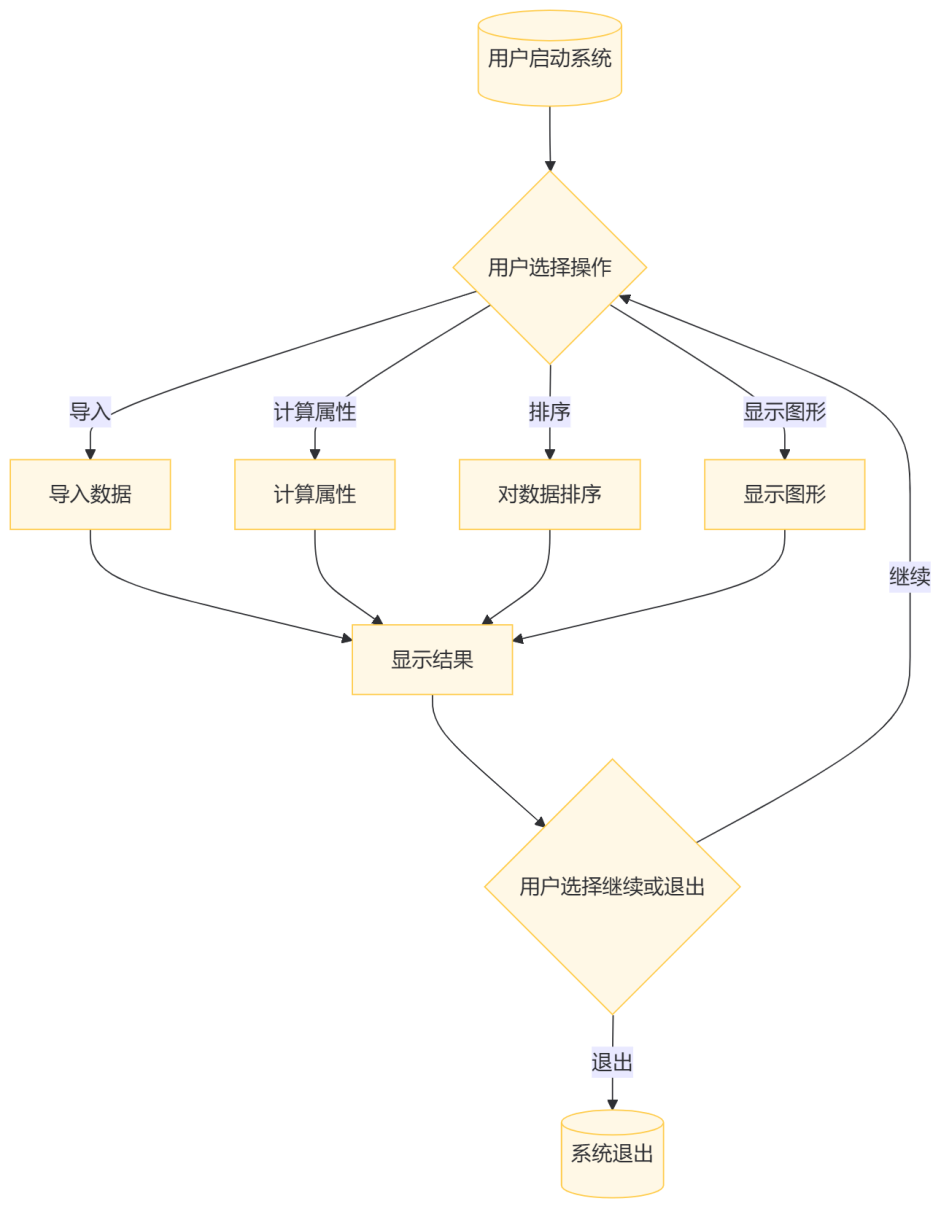
Triangle, Rectangle, Circle, Polygon, Ellipse 继承自 ClosedFigure

DrawingPanel 依赖于 ClosedFigure

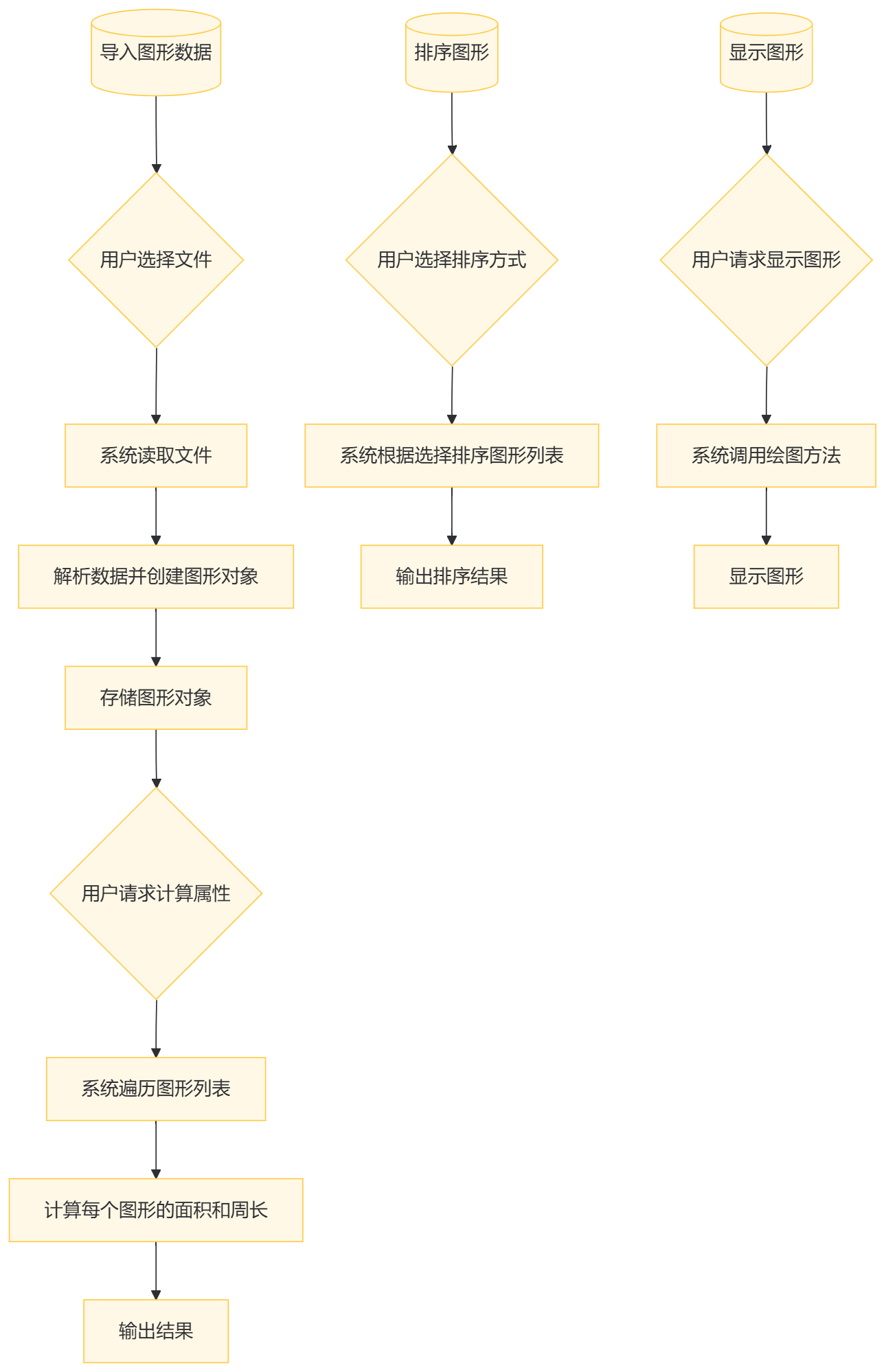
PerimeterComparator 和 AreaComparator 依赖于 ClosedFigure

4.2 流程设计

1、系统流程设计



1. 程序流程设计



导入图形数据

用户通过图形用户界面（GUI）或命令行界面（CLI）启动系统。

系统提示用户选择一个包含图形数据的文件。

用户在文件系统中浏览并选择相应的文件。

系统读取文件

系统接收用户选择的文件路径。

系统打开并读取文件内容，准备进行解析。

解析数据并创建图形对象

系统根据文件格式解析文件内容。

系统根据解析出的数据创建图形对象，例如圆形、矩形、多边形等。

每个图形对象包含其特定的属性，如半径、边长、坐标等。

存储图形对象

解析并创建的图形对象被存储在系统内存中的数据结构里，如数组、列表或数据库。

这些对象随时准备被进一步处理或查询。

计算图形属性

用户请求系统计算图形的属性，如面积和周长。

系统遍历存储图形对象的数据结构。

对于每个图形对象，系统根据其属性计算面积和周长。

系统将计算结果存储或准备输出。

输出结果

系统将计算出的每个图形的面积和周长输出到GUI的表格、CLI的控制台或存储到文件中。

用户可以查看或进一步使用这些计算结果。

排序图形

用户请求对图形列表进行排序。

用户选择排序方式，如按面积、周长或创建时间等。

系统根据用户选择的排序方式对图形对象列表进行排序。

系统将排序后的结果输出到GUI、CLI或存储到文件中。

显示图形

用户请求系统显示图形。

系统调用内置或第三方绘图库的绘图方法。

系统在屏幕上绘制每个图形对象，根据需要可能还会包括图形的标签、尺寸等附加信息。

用户可以在GUI中交互式地查看图形，或在CLI中查看图形的文本表示。

## 五、系统实现

1. 类：ClosedFigure

实现能力描述： ClosedFigure 是所有闭合图形的抽象基类，定义了基本的图形属性和方法，如周长、面积和绘制图形的方法。

package project1;

import java.awt.\*;

public abstract class ClosedFigure extends Figure {

protected String shape;

protected ClosedFigure(Point point, String shape) {

super(point);

this.shape = shape;

}

protected ClosedFigure() {

this(new Point(), "未知");

}

public abstract double perimeter();

public abstract double area();

public abstract String getFigureType();

public abstract void draw(Graphics g, int x, int y);

public String toString() {

return this.shape + ", 周长: " + String.format("%1.2f, 面积: %1.2f", this.perimeter(), this.area());

}

}

2. 类：Triangle

实现能力描述： Triangle 类继承自 ClosedFigure，实现了三角形的周长和面积计算，以及绘制三角形的方法。

类实现代码：

package project1.ClosedFIgure;

import project1.ClosedFigure;

import project1.Line;

import java.awt.\*;

public class Triangle extends ClosedFigure {

private Point point2, point3;

protected double a, b, c;

public Triangle(Point p1, Point p2, Point p3) {

super(p1, "三角形");

this.point2 = p2;

this.point3 = p3;

this.a = Line.length(p1, p2);

this.b = Line.length(p2, p3);

this.c = Line.length(p3, p1);

}

@Override

public double perimeter() {

return a + b + c;

}

@Override

public double area() {

double s = (a + b + c) / 2;

return Math.sqrt(s \* (s - a) \* (s - b) \* (s - c));

}

@Override

public String getFigureType() {

return this.shape;

}

@Override

public void draw(Graphics g, int x, int y) {

g.drawLine(x, y, point2.x, point2.y);

g.drawLine(point2.x, point2.y, point3.x, point3.y);

g.drawLine(x, y, point3.x, point3.y);

g.drawString(this.shape, x, y - 20);

}

}

5.2程序运行主体

1. 系统总控程序（含菜单）

实现能力描述： 系统总控程序负责初始化程序、读取输入数据、处理各种图形的计算和排序，并输出结果到文件。此外，它还负责展示图形的图形界面。

package project1;

import java.util.ArrayList;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

ArrayList<ClosedFigure> figures = new ArrayList<>();

Scanner scanner = new Scanner(System.in);

while (true) {

System.out.println("请选择操作:");

System.out.println("1. 读取图形数据");

System.out.println("2. 计算并输出图形信息");

System.out.println("3. 显示图形");

System.out.println("4. 退出");

int choice = scanner.nextInt();

switch (choice) {

case 1:

readFigureData(figures);

break;

case 2:

calculateAndOutputFigures(figures);

break;

case 3:

showFigures(figures);

break;

case 4:

System.out.println("退出程序");

return;

default:

System.out.println("无效选择，请重新输入。");

}

}

}

private static void readFigureData(ArrayList<ClosedFigure> figures) {

// 读取图形数据的实现

}

private static void calculateAndOutputFigures(ArrayList<ClosedFigure> figures) {

// 计算和输出图形信息的实现

}

private static void showFigures(ArrayList<ClosedFigure> figures) {

// 显示图形的实现

}

}

2. 系统用户交互界面的设计

交互流程分析： 用户通过命令行界面选择不同的操作，可以读取图形数据、计算并输出图形信息或显示图形。每个操作对应不同的功能模块，系统会根据用户的选择执行相应的操作。

交互界面实现代码： 上面的代码中，用户通过菜单选项与程序进行交互，程序会根据用户的输入执行不同的功能。以下是交互流程的进一步实现：

private static void readFigureData(ArrayList<ClosedFigure> figures) {

// 示例：从文件中读取图形数据并添加到 figures 中

// 使用 BufferedReader 读取文件内容

}

private static void calculateAndOutputFigures(ArrayList<ClosedFigure> figures) {

// 计算图形的面积和周长，并按要求输出到文件

}

private static void showFigures(ArrayList<ClosedFigure> figures) {

JFrame frame = new JFrame("图形显示");

frame.setSize(800, 600);

frame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

DrawingPanel drawingPanel = new DrawingPanel(figures);

frame.add(drawingPanel);

frame.setVisible(true);

}

## 六、总结

1. 设计开发过程中存在的各种问题和难点

在开发过程中，主要遇到的问题包括类之间的继承关系设计、图形的绘制逻辑以及数据输入输出的处理。特别是在实现不同图形的周长和面积计算时，确保每个图形类的方法实现一致性和正确性是一个挑战。此外，如何有效管理和显示多种图形对象的状态和数据也增加了开发的复杂性。

2. 考虑的不同方案的得失

在设计图形类时，曾考虑过使用接口而非抽象类来定义图形的基本属性和方法。这种方案的优点是更加灵活，可以支持多种不同实现。但缺点是需要在每个实现类中重复定义一些基本方法，增加了代码冗余。最终选择使用抽象类来减少重复代码，提高了可维护性。

3. 调试过程中发现的问题

调试过程中发现一些逻辑错误，特别是在计算图形面积和周长的公式实现上。此外，图形在界面上的绘制位置和大小调整也需要反复测试，以确保能够正确显示。数据输入时的格式不一致也导致了程序崩溃，需要加强对用户输入的验证和处理。

4. 现有开发设计的系统的优势和局限性

该系统的优势在于设计结构清晰，功能模块化，使得扩展新图形类型相对简单。同时，界面友好，用户可以方便地进行各种操作。然而，局限性在于目前仅支持基本的图形类型，缺乏对更复杂图形（如不规则多边形）的处理。此外，命令行界面的交互体验相对较差，未来可以考虑采用更现代的图形用户界面。

5. 对课程设计的总结和展望

通过本次课程设计，我深刻体会到了面向对象设计的重要性及其在实际开发中的应用。同时，对图形处理的基本概念有了更深入的理解。未来可以考虑将该系统扩展为更完整的图形处理软件，增加更多图形类型和计算功能，提高用户体验。同时，学习并应用更先进的图形库或框架，提升系统的交互性和可用性，将是我在后续学习中的目标。

6.体会和思考

遇到的难点：主要集中在数据解析和图形绘制的准确性上。

改进的地方：可以增强用户界面交互性，考虑使用更现代的图形用户界面库，如JavaFX或Swing。

展望：希望能进一步完善图形处理功能，支持更多类型的几何图形，同时提升系统的性能和用户体验。