

**实验报告**

|  |  |
| --- | --- |
| **课程名称**: | 面向对象高级编程 |
| **开课学期**: | 2023-2024学年 第1学期 |
| **专业**: | 软件工程 |
| **班级年级**: | 2022级1班 |
| **学生姓名**: | 吴孜远 |
| **学号:** | 222022321062009 |
| **实验教师:** | 王晓蒙 |

**计算机与信息科学学院 软件学院**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验项目名称 | | 实验4 抽象与接口 | | | |
| 实验时间 | | 2023.11.2 | Type | \*验证性 □设计性 □综合性 | |
| 问题描述：（1）设计一些支持`栅格化`的矢量形状对象，例如圆形、矩形、正方形、三角形、多边形等。（2）转换得到的栅格对象支持合并（merge）、相交（intersect）和裁剪（clip）等操作。（3）通过图形界面或控制台进行效果展示。   1. 程序运行截图   分别为圆形、矩形：    分别为merge、intersect、clip      2、基于UML绘制程序类图    3、矩形类的栅格化实现代码    4、增加几何形状类代码（如椭圆、三角形等）  **椭圆：**  public class Ellipse extends Shape{  public int centerX,centerY, semiMajorAxis, semiMinorAxis;    public Ellipse(int x, int y, int a, int b) {  centerX = x;  centerY = y;  semiMajorAxis = a;  semiMinorAxis = b;  }    @Override  public Raster raster() {  int minx = centerX - semiMajorAxis;  int miny = centerY - semiMinorAxis;  int maxx = centerX + semiMajorAxis;  int maxy = centerY + semiMinorAxis;  int rasterW = maxx - minx + 1;  int rasterH = maxy - miny + 1;  Raster r = new Raster(minx, miny, rasterW, rasterH);  for(int x = minx; x < minx + rasterW; x++) {  for(int y = miny; y < miny + rasterH; y++) {  if(isPointInsideEllipse(x, y)) {  r.setValue(x, y, 1);  }  }  }  return r;  }    //用椭圆的基本公式判断是否在椭圆内  public boolean isPointInsideEllipse(int x, int y) {  double value = Math.pow((double)(x - centerX) / (double) semiMajorAxis, 2) +  Math.pow((double)(y - centerY) / (double) semiMinorAxis, 2);  return value <= 1;  }  }    **三角形：**  public class Triangle extends Shape{  private int point1X, point1Y, point2X, point2Y, point3X, point3Y;    public Triangle(int p1X, int p1Y,  int p2X, int p2Y,  int p3X, int p3Y) {  setPoint1X(p1X);  setPoint1Y(p1Y);  setPoint2X(p2X);  setPoint2Y(p2Y);  setPoint3X(p3X);  setPoint3Y(p3Y);  }    @Override  public Raster raster() {  int minx = Math.min(Math.min(point1X, point2X), point3X);  int miny = Math.min(Math.min(point1Y, point2Y), point3Y);  int maxx = Math.max(Math.max(point1X, point2X), point3X);  int maxy = Math.max(Math.max(point1Y, point2Y), point3Y);  int rasterW = maxx - minx + 1; //+1是为了将两个界都算上  int rasterH = maxy - miny + 1;  //创建栅格  Raster r = new Raster(minx, miny, rasterW, rasterH);  for(int x = minx; x < minx + rasterW; x++) {  for(int y = miny; y < miny + rasterH; y++) {  if(isInsideTriangle(x, y)) {  r.setValue(x, y, 1);  }  }  }  return r;  }    //用计算当前位置在各个顶点权重的方法，来判断是否在三角形内  public boolean isInsideTriangle(int x, int y) {  double alpha = (double)((point2Y - point3Y) \* (x - point3X) +  (point3X - point2X) \* (y - point3Y)) /  (double)((point2Y - point3Y) \* (point1X - point3X) +  (point3X - point2X) \* (point1Y - point3Y));  double beta = (double)((point3Y - point1Y) \* (x - point3X) +  (point1X - point3X) \* (y - point3Y)) /  (double)((point2Y - point3Y) \* (point1X - point3X) +  (point3X - point2X) \* (point1Y - point3Y));  double gamma = 1 - alpha - beta;  // 判断点是否在三角形内  return alpha >= 0 && beta >= 0 && gamma >= 0 &&  alpha <= 1 && beta <= 1 && gamma <= 1;  }    public int getPoint1X() {  return point1X;  }  public void setPoint1X(int point1x) {  point1X = point1x;  }  public int getPoint1Y() {  return point1Y;  }  public void setPoint1Y(int point1y) {  point1Y = point1y;  }  public int getPoint2X() {  return point2X;  }  public void setPoint2X(int point2x) {  point2X = point2x;  }  public int getPoint2Y() {  return point2Y;  }  public void setPoint2Y(int point2y) {  point2Y = point2y;  }  public int getPoint3X() {  return point3X;  }  public void setPoint3X(int point3x) {  point3X = point3x;  }  public int getPoint3Y() {  return point3Y;  }  public void setPoint3Y(int point3y) {  point3Y = point3y;  }  }    5、栅格裁剪（clip）操作的实现代码  public Raster clip(Raster clipped) {  return project(clipped, new Raster[] {this, clipped}, (ins, x, y) ->{  return ins[0].getValue(x, y) == 1;  });  }    6、思考并阐述程序中接口的用法和作用  1. IRaster接口包含了一个抽象函数raster()，它的实现是根据具体情况决定的，不同的对象有不同的栅格化方式。我们可以对不同的对象进行栅格化，图形（shape）只是其中一类（我们还可以使图形以外的东西栅格化），所以我们先用抽象类shape来实例化IRaster接口，由于是抽象类，且shape又可以分为圆形矩形等，它们的raster函数都不同，因此不用在shape中定义具体实现，我们只需要在shape的子类中实现具体的raster就可以。  这样的设计体现出了多态性，不同的图形甚至其它类可以对raster实现不同的定义。IRaster 提供了一个抽象层次，这使得接口的使用者能够定义自己的具体实现，从而实现根据特定需要定制的栅格转换象层次。  2. 在Raster类中，有合并、交集、裁剪等操作，由于其执行过程有相似之处，所以我们用project函数来封装这部分操作，也就是进行判断并输出结果。在project中需要传入一个判断器，这个判断器predicate定义为**函数式接口**predicate，里面有一个抽象函数test用于判断，test在project中才会通过匿名函数lambda在传入的同时具体实现。  这样将共性操作抽象出来提高了代码的可读性，简化了代码。 | | | | | |
| 教师评阅 | 程序完整，执行正确（40%）： | | | |  |
| 设计逻辑合理，类图正确（40%）： | | | |  |
| 代码、文档格式规范（20%）： | | | |  |
| 实验成绩： | | | | |