Assignment 2

1.

Get the Point2D package that contains all the essential tools for geometry

Define a class named SegmentIntersection.

Declare eight variables with a decimal type; they are x1, x2, x3, x4y1, y1, y1, y4;

Define a method Point2D and name the variable point1. The reference variable is of the type Point2D. The horizontal and vertical coordinate of this point1 is (x1, y1). In the same way, define variable point2, point3, point4.

Define a decimal variable kA, which refers to the slope of the segment A. Assign an equation for kA. Same way with kB. Define variables bA and bB.

Assume there is an intersection (pointX, pointY) of segment A and B. Their coordinate can be calculated with kA kB bA bB. Define a method Point2D named intersectionPoint. Assign the new value (pointX, pointY) to the new reference variable.

Define and public a function that returns no value, named calculateLength.

Compare and calculate the largest and the smallest horizontal coordinate and vertical coordinate of the two segments, naming them yA\_MaxX yA\_MinX, yA\_MaxY, yA\_MinY. These values of segment 2 can be done in the same manner.

Return the largest and the smallest value of two segments.

If any maximum value of segment 1 is smaller than that of segment 2, or, if any minim value of segment 1 is larger than that of segment 2, it means the Straddle Test (跨立实验) fails. There is no need to continue to run the rest of code. Function ends and out prints “There is no intersection between these two segments.”

If the biggest and smallest value meet the demand of Straddle test. Execute next function: calculate the coordinates of the intersection of these two segments. Because if there is an intersection point, the point will locate on both of segments, which means the coordinate of this point will work for both of the equations of two segments.

Call solve function pass parameter segment 1, segment 2.

pointX = (bA - bB)/(kA - kB);

pointY = (bA \* kB - kA \* bB)/(kB - kA);

Declare a main method

In this method, assign values to x1, x2, x3, x4, y1, y2, y3, y4.

Define a method called result, under the type of segmentIntersection.

Execute the calculation process and out print the results.

2. Draw a circle, with (x, y, radius, length) = (10, 10, 5, 0.05)

//java.lang.Object

javafx.scene.Node

javafx.scene.shape.Shape

javafx.scene.shape.Circle // some package suggestions I found online.

java.awt.Graphics;

java.awt.Graphics2D;

public class Circle //define a class named Circle//

Circle circle = new Circle (); //declare a new function under the type of Circle and name it circle//

circle.setCenterX (10); //declare a method named setCenterX//

circle.setCenterY (10); //declare a method named setCenterY//

circle.setRadius (5); //declare a method named setRadius//

circle.setLength (0.05); //declare a method named setLength//

declare an array named Curve to represent all the points in the circle. There will be n values in this array. n = 2πr/0.05. I use the perimeter to divide the length of each small segments. The result n represents “how many segments I’m going to draw to make it a circle”. So n is also the number of values in the array.

declare a function to rank the values in this Curve array.

//I felt something is missing here, but I cannot figure out what it is//

Public static void main(String [] args){ //Public a main function to draw all the variabes//

PS: I read about several ready-to-use packages for geometry in java. I think I could use the methods that already defined in that package to draw the circle. The code was really short and simple.

For example, use Graphics2D.draw(Shape) with an instance of java.awt.geom.Ellipse2D. They declare a method:

Shape theCircle = new Ellipse2D.Double(centerX – radius, center – radius, 2\*radius, 2\*radius);

G2d.draw(the circle);