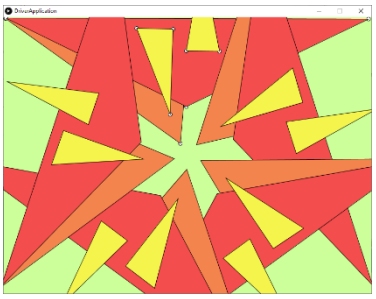
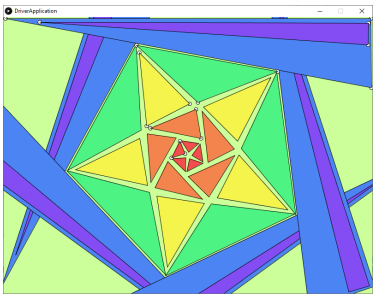
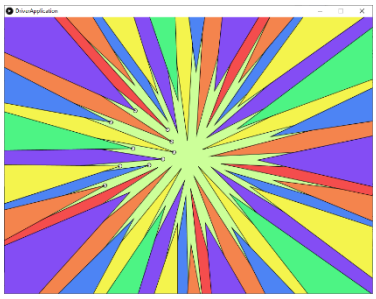
P03 Kaleidoscopic Pen

Programming II (CS300) Fall 2019	**Due: 9:59PM on September 25, 2019**
Department of Computer Sciences	Pair Programming Allowed Only
University of Wisconsin-Madison	When Registered through Quiz 03

Overview and Learning Objectives:

This assignment involves developing a graphical art tool to generate kaleidoscopic images. Here are some examples of images producible with this tool by the end of this assignment:

Through the process of completing this assignment, you will practice good object oriented design by organizing your code into instantiable classes with private fields. You will implement constructors for these classes, along with accessor and mutator methods. You will get more experience working with the processing library.

Grading Rubric:

5 points	**Pre-Assignment Quiz:** You should not have access to this write-up without first completing this pre-assignment quiz through Canvas. If you have not already, please complete this quiz as soon as you have access to this course on Canvas.
15 points	**Immediate Automated Tests:** Upon submission of your assignment to Gradescope, you will receive feedback from automated grading tests about whether specific parts of your submission conform to this write-up specification. If these tests detect problems in your code, they will attempt to give you some feedback about the kind of defect that they noticed. Note that passing all of these tests does NOT mean your program is correct. To become more confident in this, you must write and run additional tests of your own.
10 points	**Manual Grading Feedback:** After the deadline for an assignment has passed, the course staff will begin manually grading your submission. We will focus on looking at your algorithms, use of programming constructs, and the style and readability of your code. This grading usually takes about a week from the hard deadline, after which you will find feedback on Gradescope.
20 points	**Supplemental Automated Tests:** When your manual grading feedback appears on Gradescope, you will also see the feedback from these additional automated grading tests. These tests are similar to the Immediate Automated Tests, but check your code against different requirements within this specification.
50 points	**TOTAL**

Additional Assignment Requirements:

The ONLY static fields allowed in your code are the provided constants: POINT_DIAMETER and COLORS.

1. Project Files and Setup

Create a new Java Project in Eclipse (using Java 8) that is called: P03 Kaleidoscopic Pen. As with the previous assignment, you will need a jar file that contains the processing library. Download this core.jar file, copy it into your project folder, and then add it to your build path. For more detailed instructions about adding a jar file to your project, see the write-up for P02 Matching Game.

We have prepared a DriverApplication.java source file for you to download and copy into your project's src folder. You won't need to change anything in this file until the final step of this assignment. Before you can compile and run this program though, you will first need to add another new class called TrianglePen to your project. This TrianglePen class needs a constructor and instance method with the following signatures. Even with these empty definitions, you should now be able to compile and run your program: resulting in the appearance of blank window. Please seek help through piazza or from the course staff if you have any trouble getting this code to compile and run.

```
public TrianglePen(PApplet processing, boolean showPoints){}
public void update(int mouseX, int mouseY, boolean mousePressed, char keyPressed){}
```

2. Event Handling

Add the following private fields to your TrianglePen class, and initialize them to false and '\0' respectively. If you haven't already, also add processing and showPoints fields to store the arguments passed into the constructor.

```
private boolean mouseWasPressed; // mousePressed from previous update() call
private char keyWasPressed; // keyPressed from previous update() call
```

Then use the following code to define your update method:

```
// process mouse-based user input
if(mousePressed != mouseWasPressed) {
  if(mousePressed) handleMousePress(mouseX, mouseY);
  else handleMouseRelease(mouseX, mouseY);
}
if(mousePressed) handleMouseDrag(mouseX, mouseY);
mouseWasPressed = mousePressed;

// process keyboard-based user input
if(keyPressed != keyWasPressed) handleKeyPress(mouseX, mouseY, keyPressed);
keyWasPressed = keyPressed;

// draw everything in its current state
draw();
```

You will need to define five new private instance methods with void return-types, and signatures that allow for the arguments passed in the examples above: handleMousePress, handleMouseRelease, handleMouseDrag, handleKeyPress, and draw. For now, these methods can have empty definitions. Although, you are encouraged to experiment with temporary print statements to ensure that you understand when each of these methods is called by the provided implementation of update.

3. Drawing Points

Create a new class called Point in your project. The constructor for this class should take two integers: an x-position followed by a y-position and should store them in instance fields. You should add accessor methods for each of these fields named getX and getY. And you should add a mutator method called setPosition that takes the same input parameters as the constructor and updates these instance fields.

In order to see and manipulate Point objects on the screen, we will define a constant POINT_DIAMETER of value 8 within this class. You should make use of this constant while defining these last two instance methods within the point class. See the appendices for a listing of useful processing methods, and some tips on checking whether a specified point overlaps an arbitrary circle.

```
public void draw(PApplet drawTo) // draw a white circle at this point's position
public boolean isOver(int x, int y) // returns true when the position x,  y
// is within the circle drawn to visualize this point, otherwise returns false
```

Your TrianglePen class is responsible for creating and drawing instances of this Point class. Add a private ArrayList<Point> instance field to your TrianglePen class, and initialize it to be empty in the constructor. Then create and add a new point to this list from within the TrianglePen.handleMousePress() method's definition. In order to see these newly created points, you'll also need to update the TrianglePen.draw() method's definition to iterate through all of the points in your array list, and draw each of them. Make sure that you are able to create and see these points while running your program, before you move on to the next steps.

4. Drawing Triangles

Create a new class called Triangle in your project. The constructor of this class should take three Point references followed by an int color index, each of which should be stored in private instance fields. Add the following COLORS array to your Triangle class to work with colors. The color index used to instantiate new triangles refers to the color within this array that the triangle should be drawn with. Also create a mutator method named setColor() which takes an int color index as input, and updates the color of the triangle that it is called on.

```
private static final int[] COLORS = new int[] { // int packed w/8 bits of ARGB
//  WHITE, RED,     ORANGE,  YELLOW,  GREEN,     BLUE,      INDIGO,   VIOLET
    -1,    -766643, -752563, -723891, -11668348, -11696908, -8106508, -766476 };
```

As in the Point class, our Triangle class needs a draw() and isOver() method. The signatures of these methods should match exactly what we added to the Point class, but their implementations and behavior will clearly need to be different. See the appendices for helpful processing drawing methods, and for tips to determine whether a point lies within a triangle.

Next, we'll add the ability to create and see Triangles to our TrianglePen class. Add an ArrayList of Triangles to your pen class, similar to how this was done with our ArrayList of Points in the previous step. Modify the definition of TrianglePen.handleMousePress() so that when every third point is created, a triangle is also created with the previous three points as vertices. This means that the first triangle will be created when the 3rd point is created, and will use points 1-3 as vertices, the second triangle will be created when the 6th point is created, and will use points 4-6 as vertices, etc. Ensure that the number of triangles in your array list is never more than the number of points divided by three. If you haven't already, update the definition of TrianglePen.draw() so that it draws all the triangles in your

array list. You can decide whether you'd prefer to draw the points either before or after drawing the triangles, so that they end up either below or above them. For an extra challenge, you are also welcome (but not required: no extra points or credit) to interleave this drawing, so that the points of each triangle appear on top of their respective triangles but beneath other triangles that are drawn later.

The last requirement for this step is that your TrianglePen class should only draw and display points when the boolean showPoints argument passed into its constructor was true. It should still always display the triangles independent of showPoints.

5. Coloring Triangles and Dragging Points

Let's allow the user to change the color of a triangle by positioning their mouse over it and pressing a number key on their keyboard. We'll implement this in the TrianglePen.handleKeyPress() definition. Check whether the key pressed by the user is a digit key between 0 and 7 (inclusive), and also check whether the mouse is currently over any triangles. If so, set the color index for all such triangles to be: the character pressed by the user minus the character zero. Since all of the character codes for digits are consecutive and in numerical order, subtracting the zero character is a convenient way to convert a digit char into its corresponding int value. Make sure this works by running your program and changing the colors of a few triangles.

Next allow the user to change the positions of points that they have created by dragging on them. Add a private Point instance field to your TrianglePen class to keep track of which point the user is currently dragging. This field should be null when the user is not dragging a point. Within the method TrianglePen.handleMousePress(), check whether the mouse is being pressed over any points before creating any new points or triangles. If the mouse is over an already existing point, track that point as being dragged and skip the creating of any new points or triangles. When the mouse is pressed over multiple points, you can choose to drag any one of them. As long as a point is being dragged, that point's position should be updated to match the position of the mouse. This can be implemented within the TrianglePen.handleMouseDrag() method. And of course whenever the mouse is released (and the TrianglePen.handleMouseRelease() method is called), this will end the dragging of points. Ensure that you are able to change the positions of points before moving on to the last step of this assignment.

6. The Kaleidoscopic Pen

Notice that we have used instantiable classes to organize the methods and data of elements that we obviously wanted multiple instances of: Points and Triangles. However, we only have a single Pen object. This may have made you wonder whether it would be better to organize this code as a collection of static fields and methods, rather than as an instantiable class. If we had chosen to use static methods and fields, this last step would have been much more difficult… In this step we will be create a new class that manages several TrianglePens that each create and edit their own lists of points and triangles.

Create one last class called KaleidoscopePen in your project, with the following constructor and instance method signatures:

```
public KaleidoscopePen(PApplet drawTo, int numberOfTrianglePens)
public void update(int mouseX, int mouseY, boolean mousePressed, char keyPressed)
```

This class should have a private TrianglePen array instance field, which must be initialized to hold as many new pen objects as is specified by the constructor's numberOfTrianglePens parameter. The

update method for this class, should then iterate through this array of pens and call each those pens update methods with the same arguments. Test this out by updating your DriverApplication to make use of a KaleidoscopePen instead of a TrianglePen. Construct this KaleidoscopePen to contains numberOfTrianglePens = 5. You should NOT see anything different when running this code, since all five pens are drawing the same triangles in the same positions.

To change this, you can temporarily modify KaleidoscopePen.update() to send different mouseX coordinates to each of the five pen's update methods. Instead of sending mouseX, try sending (mouseX+i*10) where i corresponds to the index of the pen that is being updated, and is therefore different for each pen. With all of these points, it can become confusing to know which can be edited. For this reason, change the construction of these five TrianglePen objects so that only first is created with a true showPoints argument, and other pass false. This should help you see only those points that can be edited with your mouse.

Instead of adjusting each pen by moving mouseX by (i*10) to the right, we would like the mouse position passed into each pen to be rotated around the center of the screen. The amount of rotation for each pen should be (i * 2π / numberOfTrianglePens) where i is the index of the pen being updated. The appendix below includes a helper method that you may use to compute these rotated positions.

7. Assignment Submission

Congratulations on completing this assignment! The only four files that you should submit for this assignment are: Point.java, Triangle.java, TrianglePen.java, and KaleidoscopePen.java. Your score for this assignment is based on your submission that is marked "active" and is made prior to the hard deadline.

Appendix I: Helpful Instance Methods from the Processing Library and Java API

processing.core.PApplet methods:

void fill(int color) // pass -1, to set fill color to white

void circle(int x, int y, int diameter)

java.util.ArrayList<E> methods:

boolean add(E element)

E get(int index)

void triangle(int x1, int y1, int x2, int y2, int x3, int y3) int size()

Appendix II: Implementation Tips

a. *Tips for checking whether a Point is within a Circle*

A point is within a circle, when the distance between that point and the center of the circle is smaller than the circle's radius (radius = diameter / 2). Calculate the distance between these two points as:

$$distance = \sqrt{(point1x - point2x)^2 + (point1y - point2y)^2}$$

b. *Tips for checking whether a Point is within a Triangle*

Since the arithmetic for checking whether a point is within a triangle is a bit more complex, we are providing you with the following code to perform the computation. When using this helper method: (px,py) is the point that you are testing, and the positions of the triangles that you are testing that point against are (t1x,t1y), (t2x,t2y), and (t3x,t3y).

```
private static boolean isPointInTriangle(int px, int py,
    int t1x, int t1y, int t2x, int t2y, int t3x, int t3y) {
  px -= t1x; // don't worry about this arithmetic
  py -= t1y;
  t2x -= t1x;
  t2y -= t1y;
  t3x -= t1x;
  t3y -= t1y;
  double dotp2 = px*t2x+py*t2y;
  double dotp3 = px*t3x+py*t3y;
  double dot22 = t2x*t2x+t2y*t2y;
  double dot23 = t2x*t3x+t2y*t3y;
  double dot33 = t3x*t3x+t3y*t3y;
  double invDen = 1 / (dot33 * dot22 - dot23 * dot23);
  double a = (dot22 * dotp3 - dot23 * dotp2) * invDen;
  double b = (dot33 * dotp2 - dot23 * dotp3) * invDen;
  return (a >= 0) && (b >= 0) && (a + b < 1);
}
```

c. *Tips for rotating a point around the center of the screen*

```
/**
 * Rotates a position around the center of an 800x600 screen by the specified
 * amount, and then returns an array containing the resulting position.
 * @param x position of the point to be rotated (0 to 800 pixels)
 * @param y position of the point to be rotated (0 to 600 pixels)
 * @param angle amount of rotation to apply (angle in radians units: 0 to 2*PI)
 * @return the rotated position array: x @ index 0, y @ index 1
 */
private static int[] rotate(int x, int y, double angle) {
  x -= 400; // translate center of screen to origin (0,0)
  y -= 300;
  int[] rotatedPosition = new int[] { // rotate around origin
      (int)(x * Math.cos(angle) - y * Math.sin(angle)),
      (int)(x * Math.sin(angle) + y * Math.cos(angle)) };
  rotatedPosition[0] += 400; // return to center of screen
  rotatedPosition[1] += 300;
  return rotatedPosition;
}
```