Asst. Prof. Dr. Sasiporn Usanavasin, Dr. Jessada Karnjana, Dr. Kasorn Galajit and Dr. Akkarawoot Takhom {sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.aj}@siit.tu.ac.th
School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology, Thammasat University.

DES103(Y23S2)-LAB01-REVIEW]

Graphics

Learning Objectives

1. Understand the functionality of the `paintComponent` method in Java's Component class.
2. Gain proficiency in creating graphics objects.
3. Learn to create basic drawings using a graphics object.

****Remark**** A ***pointer finger* (👉)** refers to an explanation between students and their TA.

7.1 `Graphics` Object

❖ The **Graphics class** is an abstract class that provides a device-independent graphics interface for displaying figures and images on the screen on different platforms.

❖ Whenever any component is displayed, a **Graphics object** is created for the component.

❖ We can then apply methods in the **Graphics class** to draw things on the label's graphics context.

7.2 The `paintComponent` Method

❖ The **paintComponent** method protected void paintComponent(Graphics g) is defined in the class JComponent.

❖ This method is invoked whenever the component is first displayed or redisplayed.

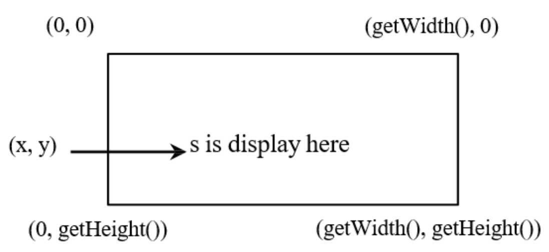
❖ The **Graphics object** g is automatically created by the JVM for every visible GUI component.

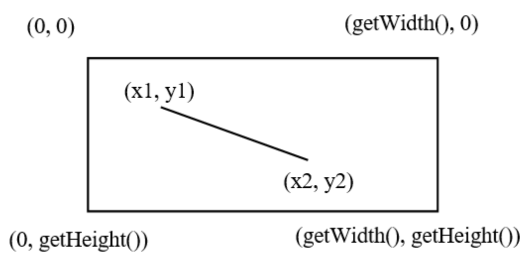
❖ The JVM obtains the Graphics object and passes it to invoke paintComponent automatically.

7.3 The `Graphics` class

❖ The `Graphics` class is in `java.awt.` It provides many useful methods for decorating the user interface.

❖ This section demonstrates examples for drawing graphics: `Drawline`, DrawOval, `DrawPolygon`.

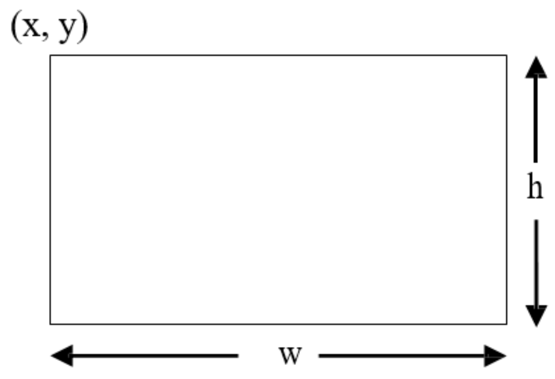




7.4. Drawing/Filling a Rectangle

7.4.1 `drawRect(int x, int y, int w, int h);`

This method is used to draw the outline of the specified rectangle. The left and right edges of the rectangle are at x and x + width. The top and bottom edges are at y and y + height. The rectangle is drawn using the graphics context's current color.

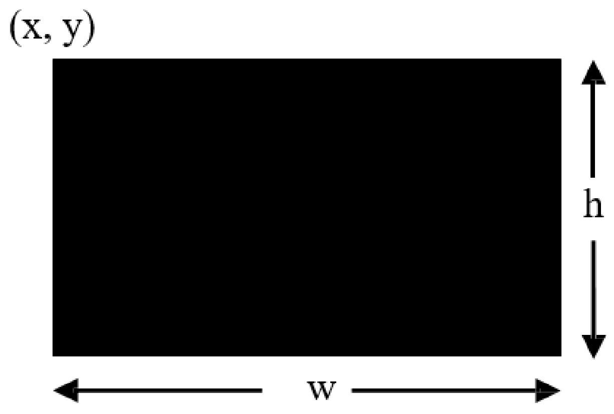


Parameters:

- x: the x-coordinate of the rectangle to be drawn.
- y: the y-coordinate of the rectangle to be drawn.
- w: the width of the rectangle to be drawn.
- h: the height of the rectangle to be drawn.

7.4.2 `fillRect(int x, int y, int w, int h);`

This method is used to fill the specified rectangle. The left and right edges of the rectangle are at x and x + width – 1. The top and bottom edges are at y and y + height – 1. The resulting rectangle covers an area width pixels wide by height pixels tall. The rectangle is filled using the graphics context's current color.



Parameters:

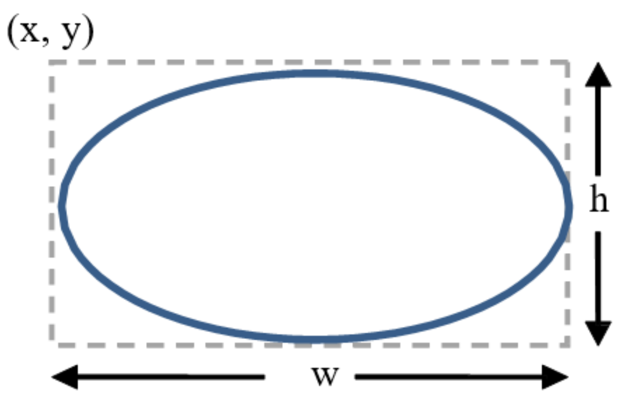
- x: the x-coordinate of the rectangle to be filled.
- y: the y-coordinate of the rectangle to be filled.
- w: the width of the rectangle to be filled.
- h: the height of the rectangle to be filled.

Asst. Prof. Dr. Sasiporn Usanavasin, Dr. Jessada Karnjana, Dr. Kasorn Galajit and Dr. Akkarawoot Takhom {sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.aj}@siit.tu.ac.th
School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology, Thammasat University.

7.5. Drawing/Filling an Oval

7.5.1 `drawOval(int x, int y, int w, int h);`

This method is used to draw the outline of an oval. The result is a circle or ellipse that fits within the rectangle specified by the x, y, width, and height arguments. The oval covers an area that is width + 1 pixels wide and height + 1 pixels tall.

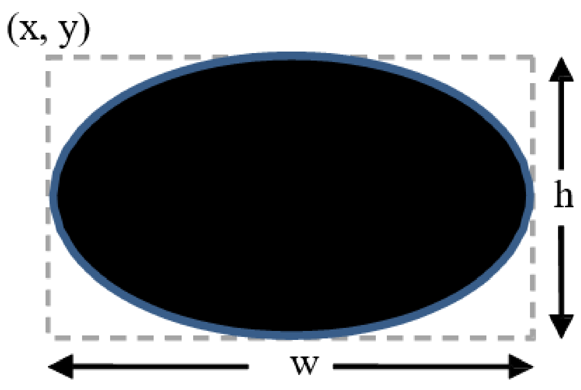


Parameters:

- x: the x-coordinate of the upper left corner of the oval to be drawn.
- y: the y-coordinate of the upper left corner of the oval to be drawn.
- w: the width of the oval to be drawn.
- h: the height of the oval to be drawn.

7.5.2 `fillOval(int x, int y, int w, int h);`

This method is used to fill an oval bounded by the specified rectangle with the current color.



Parameters:

- x: the x-coordinate of the upper left corner of the oval to be filled.
- y: the y-coordinate of the upper left corner of the oval to be filled.
- w: the width of the oval to be filled.
- h: the height of the oval to be filled.

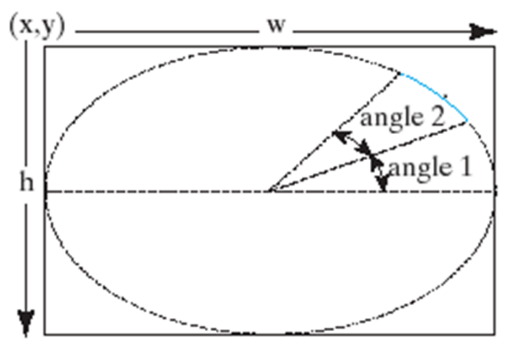
Asst. Prof. Dr. Sasiporn Usanavasin, Dr. Jessada Karnjana, Dr. Kasorn Galajit and Dr. Akkarawoot Takhom {sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.aj}@siit.tu.ac.th
School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology, Thammasat University.

7.6. Drawing/Filling an Arc

7.6.1 `drawArc(int x, int y, int w, int h, int angle1, int angle2);`

This method is used to draw the outline of a circular or elliptical arc covering the specified rectangle. The resulting arc begins at startAngle and extends for **arcAngle** degrees, using the current color. Angles are interpreted such that 0 degrees is at the 3 o'clock position. A positive value indicates a counter-clockwise rotation, while a negative value indicates a clockwise rotation. The center of the arc is the center of the rectangle whose origin is **(x, y)** and whose size is specified by the width and height arguments.

The resulting arc covers an area **width + 1 pixels** wide by **height + 1 pixels** tall. The angles are specified relative to the non-square extents of the bounding rectangle such that 45 degrees always falls on the line from the center of the ellipse to the upper right corner of the bounding rectangle. As a result, if the bounding rectangle is noticeably longer in one axis than the other, the angles to the start and end of the arc segment will be skewed farther along the longer axis of the bounds.

	**Parameters:** ● **x**: the x-coordinate of the upper-left corner of the arc to be drawn. ● **y**: the y-coordinate of the upper-left corner of the arc to be drawn. ● **w**: the width of the arc to be drawn. ● **h**: the height of the arc to be drawn. ● **angle1**: the beginning angle (startAngle). ● **angle2**: the angular extent of the arc, relative to the start angle (arcAngle).

7.6.2 `fillArc(int x, int y, int w, int h, int angle1, int angle2);`

This method is used to fill a circular or elliptical arc covering the specified rectangle. The resulting arc begins at startAngle and extends for arcAngle degrees. Angles are interpreted such that 0 degrees is at the 3 o'clock position. A positive value indicates a counter-clockwise rotation, while a negative value indicates a clockwise rotation. The center of the arc is the center of the rectangle whose origin is (x, y) and whose size is specified by the width and height arguments.

The resulting arc covers an area width + 1 pixels wide by height + 1 pixels tall. The angles are specified relative to the non-square extents of the bounding rectangle such that 45 degrees always falls on the line from the center of the ellipse to the upper right corner of the bounding rectangle. As a result, if the bounding rectangle is noticeably longer in one axis than the other, the angles to the start and end of the arc segment will be skewed farther along the longer axis of the bounds.

Parameters:

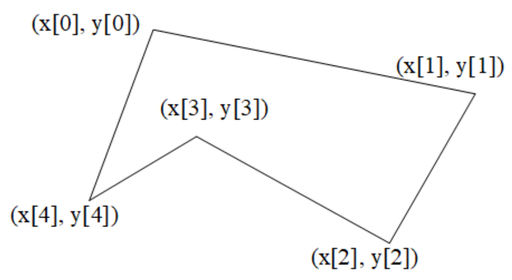
- x: the x-coordinate of the upper-left corner of the arc to be filled.
- y: the y-coordinate of the upper-left corner of the arc to be filled.
- w: the width of the arc to be filled.
- h: the height of the arc to be filled.
- angle1: the beginning angle (startAngle).
- angle2: the angular extent of the arc, relative to the start angle (arcAngle).

7.7. Drawing a Polygon and a Polyline

7.7.1 `drawPolygon(xPoints, yPoints, nPoints);`

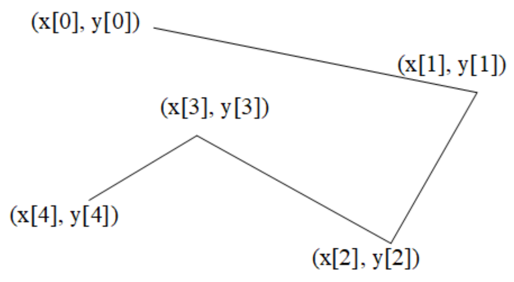
❖ This method is used to draw a closed polygon defined by arrays of x and y coordinates. Each pair of **(x, y)** coordinates defines a point.

❖ This method draws the polygon defined by **nPoints** line segments, where the first **nPoints** – 1 line segments are line segments from **(xPoints[i−1], yPoints[i−1]) to (xPoints[i], yPoints[i]), for 1 ≤ i ≤ nPoints**. The figure is automatically closed by drawing a line connecting the final point to the first point if those points are different.

 (x[0], y[0]) (x[1], y[1]) (x[3], y[3]) (x[4], y[4]) (x[2], y[2])	**Parameters:** ● **xPoints**: an array of x coordinates. ● **yPoints**: an array of y coordinates. ● **nPoints**: the total number of points.

7.7.2 `drawPolyline(xPoints, yPoints, nPoints);`

This method is used to draw a sequence of connected lines defined by arrays of x and y coordinates. Each pair of (x, y) coordinates defines a point. The figure is not closed if the first point differs from the last point.

 (x[0], y[0]) (x[1], y[1]) (x[3], y[3]) (x[4], y[4]) (x[2], y[2])	**Parameters:** ● **xPoints**: an array of x coordinates. ● **yPoints**: an array of y coordinates. ● **nPoints**: the total number of points.

Noted: The **`Graphics` class** is the abstract base class for all graphics contexts that allow an application to draw onto components that are realized on various devices, as well as onto off-screen images.

Asst. Prof. Dr. Sasiporn Usanavasin, Dr. Jessada Karnjana, Dr. Kasorn Galajit and Dr. Akkarawoot Takhom {sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.aj}@siit.tu.ac.th
School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology, Thammasat University.

7.8 Summary of `java.awt.Graphics`

A **`Graphics` object** encapsulates state information needed for the basic rendering operations that Java supports. This method summary is shown as follows:

`java.awt.Graphics`	**Description**
`+setColor(color:Color): void`	Sets a new **color** for subsequent drawings.
`+setFont(font:Font): void`	Sets a new **font** for subsequent drawings.
`+drawString(s:String, x:int, y:int): void`	Draws a **string** starting at point (x, y).
`+drawLine(x1:int, y1:int, x2:int, y2:int): void`	Draws a **line** from (x1, y1) to (x2, y2).
`+drawRect(x:int, y:int, w:int, h:int): void`	Draws a **rectangle** with specified upper-left corner point at (x, y) and width w and height h.
`+fillRect(x: int, y: int, w: int, h: int): void`	Draws a filled **rectangle** with specified upper-left corner point at (x, y) and width w and height h.
`+drawRoundRect(x:int, y:int, w:int, h:int, aw:int, ah:int): void`	Draws a **round-cornered rectangle** with specified arc width aw and arc height ah.
`+fillRoundRect(x:int, y:int, w:int, h:int, aw:int, ah:int): void`	Draws a filled round-cornered rectangle with specified arc width aw and arc height ah.
`+draw3DRect(x:int, y:int, w:int, h:int, raised:boolean): void`	Draws a 3-D rectangle raised above the surface or sunk into the surface.
`+fill3DRect(x:int, y:int, w:int, h:int, raised:boolean): void`	Draws a filled 3-D rectangle raised above the surface or sunk into the surface.
`+drawOval(x:int, y:int, w:int, h:int): void`	Draws an oval bounded by the rectangle specified by the parameters x, y, w, and h.
`+fillOval(x:int, y:int, w:int, h:int): void`	Draws a filled oval bounded by the rectangle specified by the parameters x, y, w, and h.
`+drawArc(x:int, y:int, w:int, h:int, startAngle:int, arcAngle:int): void`	Draws an arc conceived as part of an oval bounded by the rectangle specified by the parameters x, y, w, and h.
`+fillArc(x:int, y:int, w:int, h:int, startAngle:int, arcAngle:int): void`	Draws a filled arc conceived as part of an oval bounded by the rectangle specified by the parameters x, y, w, and h.
`+drawPolygon(xPoints:int[], yPoints:int[], nPoints:int): void`	Draws a closed polygon defined by arrays of x and y coordinates. Each pair of (x[i], y[i]) coordinates is a point.
`+fillPolygon(xPoints:int[], yPoints:int[], nPoints:int): void`	Draws a filled polygon defined by arrays of x and y coordinates. Each pair of (x[i], y[i]) coordinates is a point.

Asst. Prof. Dr. Sasiporn Usanavasin, Dr. Jessada Karnjana, Dr. Kasorn Galajit and Dr. Akkarawoot Takhom {sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.aj}@siit.tu.ac.th
School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology, Thammasat University.

`+drawPolygon(g:Polygon): void`	Draws a closed polygon defined by a **Polygon** object.
`+fillPolygon(g:Polygon): void`	Draws a filled polygon defined by a **Polygon** object.
`+drawPolyline(xPoints:int[], yPoints:int[], nPoints:int): void`	Draws a **polyline** defined by arrays of x and y coordinates. Each pair of (x[i], y[i]) coordinates is a point.

Reference : https://docs.oracle.com/javase/7/docs/api/java/awt/Graphics.html

[DES103(Y23S2)-LAB07-EXERCISES]

Students MUST adhere to the lab instructions and regulations provided below.
Please consult your TA to review your completed exercises and submit them on Google Classroom.

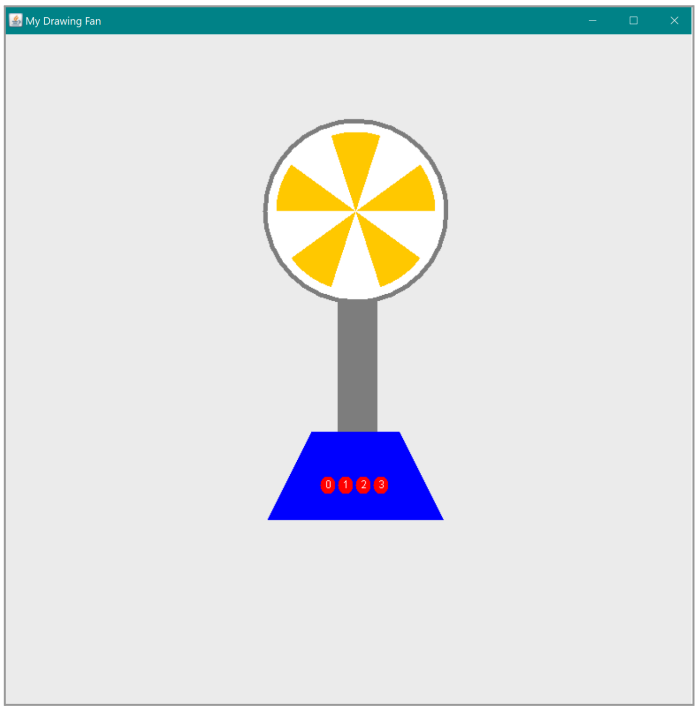
Be noticed that for all lab exercises, you need to define your *Java* project as the following name format:

<StudentID>_<Lab number>_<Exercise name>

If your student's ID is 6722300208, the name format of your java project should be:

Project1 6422300208_LAB07_ElectricFan

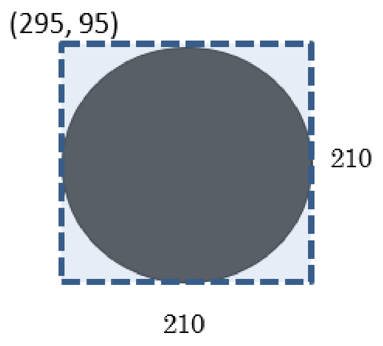
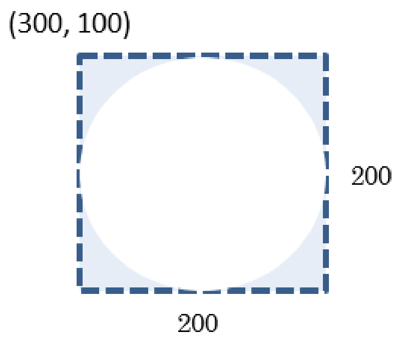
For LAB07's exercises, students are going to draw an electric fan using `paintComponent` and methods of a graphics object learned in class. The final output should look like this.



Asst. Prof. Dr. Sasiporn Usanavasin, Dr. Jessada Karnjana, Dr. Kasorn Galajit and Dr. Akkarawoot Takhom {sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.aj}@siit.tu.ac.th
School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology, Thammasat University.

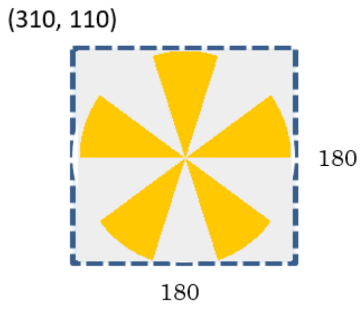
Exercise 1: (2 points)

- **Java Project**: <Student_ID>_ LAB07_ElectricFan
- **Objective**: To learn how `paintComponent` method in Component work, graphics object is created, and drawing from a `Graphics` object
- **Instruction:**
  - Create a java project, and name **"<Student_ID>_ LAB07_ElectricFan"** and write code in the following tasks.
  - Create a class namely ElectricFan which is a subclass of JPanel and override the public void paintComponent(Graphics g) method from its superclass. To draw the frame of the fan, follow the following instructions.

1. Fill an Oval with the gray color using the given dimension and location.	
2. Fill another Oval with the white color using the guided dimension and location.	
3. Download and `ElectricFanTesting.java` from Google Classroom, and run **`ElectricFanTesting`** class to see the fan's frame.	

Exercise 2: (2 points)

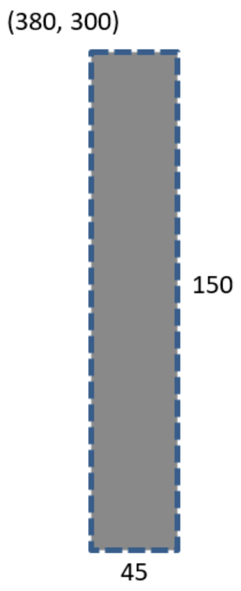
- **Java Project**: <Student_ID>_ LAB07_ElectricFan
- **Objective**: To learn how `paintComponent` method in Component work, graphics object is created, and drawing from a `Graphics` object
- **Instruction:** Continue in the method `paintComponent` of the `ElectricFan`.

1. Draw 5 orange blades of the fan (each blade's arc is 36 degree) in the guided location and dimension.	
2. Run **ElectricFanTesting** class to see the fan's frame with blades.	

Asst. Prof. Dr. Sasiporn Usanavasin, Dr. Jessada Karnjana, Dr. Kasorn Galajit and Dr. Akkarawoot Takhom {sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.aj}@siit.tu.ac.th
School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology, Thammasat University.

Exercise 3: (2 points)

- ❖ **Java Project**: <Student_ID>_ LAB07_ElectricFan
- ❖ **Objective**: To learn how `paintComponent` method in Component work, graphics object is created, and drawing from a `Graphics` object
- ❖ **Instruction:** Continue in the method `paintComponent` of the `ElectricFan`.

1. Fill the gray neck of the fan using a rectangle at the guided location and a specified dimension.	
2. Run **`ElectricFanTesting`** class to see the fan's frame with blades and the neck.	

Asst. Prof. Dr. Sasiporn Usanavasin, Dr. Jessada Karnjana, Dr. Kasorn Galajit and Dr. Akkarawoot Takhom {sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.aj}@siit.tu.ac.th
School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology, Thammasat University.

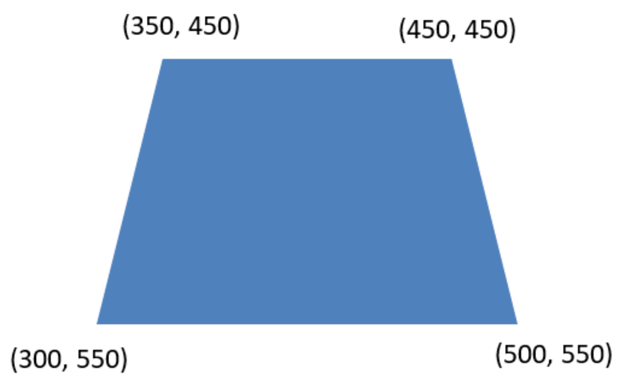
Exercise 4: (2 points)

Java Project: <Student_ID>_ LAB07_ElectricFan

Objective: To learn how `paintComponent` method in Component work, graphics object is created, and drawing from a `Graphics` object

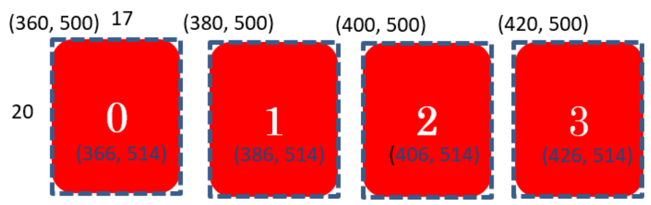
Instruction: Continue in the method paintComponent of the `ElectricFan`.

❖

1. Fill the blue trapezoidal base of the fan using the guided coordinates.	
2. Run `ElectricFanTesting` class to see the fan's frame with the blades, neck and base.	

Exercise 5: (2 points)

- ❖ **Java Project**: <Student_ID>_ LAB07_ElectricFan
- ❖ **Objective**: To learn how `paintComponent` method in Component work, graphics object is created, and drawing from a `Graphics` object
- ❖ **Instruction:** Continue in the method `paintComponent` of the `ElectricFan`.
- ❖

1. Create **4 red round speed buttons** using the guided location and a specified dimension. Remark, use `fillRoundRect` with arc's width and arc's height = 20.	 (360, 500) 17 (380, 500) (400, 500) (420, 500) 20 0 (366, 514) 1 (386, 514) 2 (406, 514) 3 (426, 514)
2. Add the white speed text 0, 1, 2, and 3 on each button in the specified locations.	
3. Run `ElectricFanTesting.java` to see the final fan output.	