Module V
Applets

Dr. Zahid Ansari

Applets

- A Java application is a stand-alone program with a `main` method
- An *applet* is a Java program that is intended to be transported over the web and executed using a web browser
- An applet doesn't have a `main` method
- Instead, there are several special methods that serve specific purposes
- appletviewer is a program that can run applets

The Applet class

- To create an applet, you must import the Applet class
  - This class is in the java.applet package
- The Applet class contains code that works with a browser to create a display window
- Applet contains several methods that give you control over the execution of your applet
- *Capitalization matters!*
  - applet and Applet are different names

Two Types of Applets

1. applets based on Applet class, these applets use Abstract Windows Toolkit (AWT) to provide the graphic user interface.
2. applets based on the Swing class JApplet, swing applets use Swing classes for GUI.

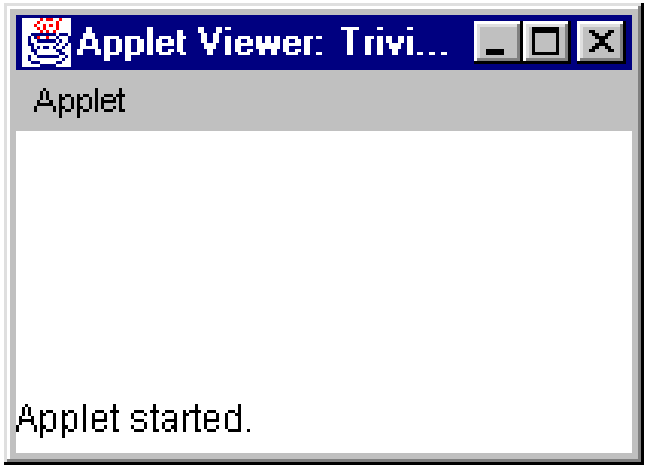
The simplest possible applet

TrivialApplet.java

```
import java.applet.Applet;
public class TrivialApplet extends Applet { }
```

TrivialApplet.html

```
<applet
    code="TrivialApplet”
    width=150 height=100>
</applet>
```



The simplest reasonable applet

```
import java.awt.*;
import java.applet.Applet;

public class HelloWorld extends Applet {
    public void paint( Graphics g ) {
        g.drawString( "Hello World!", 30, 30 );
    }
}
```

Applet methods

public void init ()

public void start ()

public void stop ()

public void destroy ()

public void paint (Graphics)

Also:

public void repaint()

public void update (Graphics)

public void showStatus(String)

public String getParameter(String)

Why an applet works

- You write an applet by *extending* the class Applet
- Applet defines methods init( ), start( ), stop( ), paint(Graphics), destroy( )
- These methods do nothing--they are stubs
- You make the applet do something by overriding these methods

public void init ( )

- This is the first method to execute
- It is an ideal place to initialize variables
- It is the best place to define the GUI Components (buttons, text fields, scrollbars, etc.), lay them out, and add listeners to them
- Almost every applet you ever write will have an `init( )` method

public void start ( )

- Not always needed
- Called after init( )
- Called each time the page is loaded and restarted
- Used mostly in conjunction with stop( )
- start() and stop( ) are used when the Applet is doing time-consuming calculations that you don't want to continue when the page is not in front

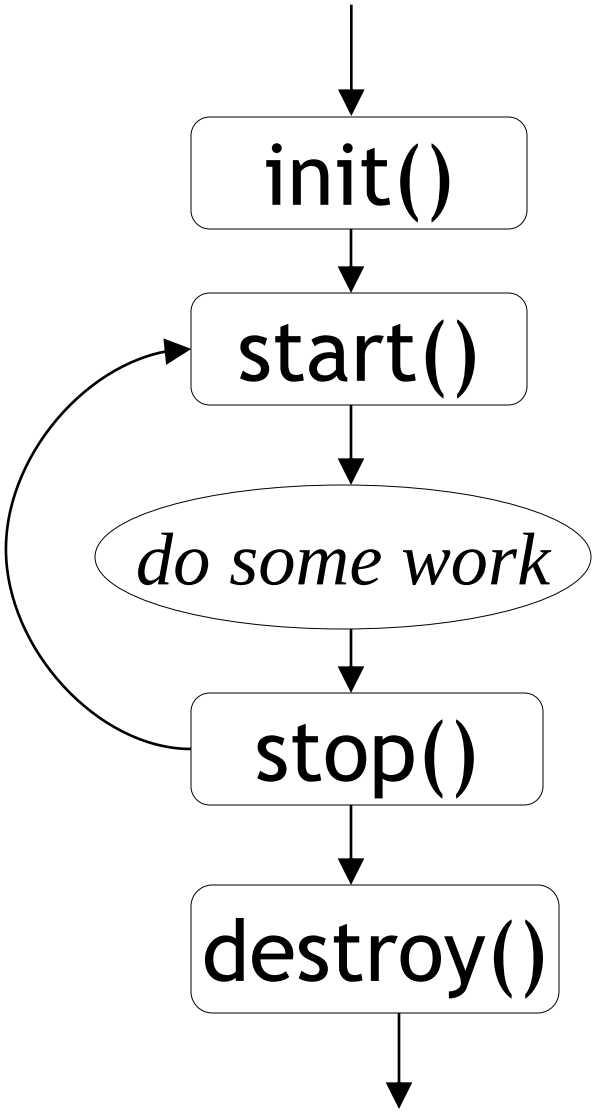
public void stop( )

- Not always needed
- Called when the browser leaves the page
- Called just before destroy( )
- Use stop( ) if the applet is doing heavy computation that you don't want to continue when the browser is on some other page
- Used mostly in conjunction with start()

public void destroy( )

- Seldom needed
- Called after stop( )
- Use to explicitly release system resources (like threads)
- System resources are usually released automatically

Applet Lifecycle
Methods are called in this order



- init and destroy are only called once each
- start and stop are called whenever the browser enters and leaves the page
- *do some work* is code called by your *listeners*
- paint is called when the applet needs to be repainted

public void paint(Graphics g)

- Needed if you do any drawing or painting other than just using standard GUI Components
- Any painting you want to do should be done here, or in a method you call from here
- *Never call* paint(*Graphics*), *c*all repaint()

repaint( )

- Call repaint( ) when you have changed something and want your changes to show up on the screen
- When you call repaint( ), Java schedules a call to update(Graphics g)

update( )

- When you call repaint( ), Java schedules a call to update(Graphics g)
- Here's what update does:

```
public void update(Graphics g) {
  // Fills applet with background color, then
    paint(g);
}
```

Sample Graphics methods

- A Graphics is something you can paint on

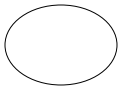
```
g.drawString("Hello", 20, 20);
```

Hello

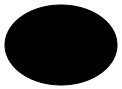
```
g.drawRect(x, y, width, height);
```

```
g.fillRect(x, y, width, height);
```

```
g.drawOval(x, y, width, height);
```



```
g.fillOval(x, y, width, height);
```



```
g.setColor(Color.red);
```

HTML

```
<html>
  <head>
    <title> Hi World Applet </title>
  </head>

  <body>
    <applet code="HiWorld.class”
      width=300 height=200>
        <param name="arraysize" value="10">
    </applet>
  </body>
</html>
```

Life Cycle of an Applet

- public void init(): This method is intended for whatever initialization is needed for an applet.
- public void start(): This method is automatically called after init method. It is also called whenever user returns to the page containing the applet after visiting other pages.
- public void paint(Graphics g): This method is called by the browser after init and start. Re-invoked whenever the browser redraws the screen. (Typically when part of the screen is obscured and then re-exposed). This method is where user level drawings are placed.

Life Cycle of an Applet

- **public void stop(): This method is automatically called whenever the user moves away from the page containing applets. This method can be used to stop an animation.**
- **public void destroy(): This method is only called when the browser shuts down normally.**

Useful Applet Methods

- **getCodeBase, getDocumentBase**
  - The URL of the:
  - Applet file – getCodeBase
  - HTML file – getDocumentBase
- **getParameter**
  - Retrieves the value from the associated HTML PARAM element
- **getSize**
  - Returns the dimensions (width, hight) of the applet
- **getGraphics**
  - Retrieves the current Graphics object for the applet
  - The Graphics object does not persist across paint invocations

Useful Applet Methods

- **showDocument (AppletContext Method)**
  - **getAppletContext().showDocument(…)**
  - **Asks the browser to retrieve and display a Web page**
- **showStatus**
  - **Displays a string in the status line at the bottom of the browser**
- **getCusrsor, setCusrosr**
  - **Define the Cursor for the mouse, for example CROSSHAIR_CURSOR, HAND_CURSOR, WAIT_CURSOR**

Useful Applet Methods

- **getAudioClip, play**
  - **Retrieves an audio file from a remote location and plays it.**
- **getBackground, setBackground**
  - **Gets/sets the background colour of the applet**
  - **SytemColor class provides access to desktop colors**
- **getForeground, setForeground**
  - **Gets/sets the foreground color of applet (default color of drawing operations)**

HTML tags for applets

```
<APPLET
  // the beginning of the HTML applet code
   CODE="demoxx.class"
   // the actual name of the applet (usually a 'class' file)
   CODEBASE="demos/"
   // the location of the applet (relative as here, or a full URL)
    NAME="SWE622"
    // the name of the instance of the applet on this page
    WIDTH="100"
    // the physical width of the applet on the page
    HEIGHT="50"
    // the physical height of the applet on the page
    ALIGN="Top"
    // align the applet within its page space (top, bottom, center)
```

```
// An Applet skeleton.
import java.awt.*;
import java.applet.*;
/*
<applet code="AppletSkel" width=300 height=100>
</applet>
*/
public class AppletSkel extends Applet {
    // Called first.
    public void init() { // initialization }

    /* Called second, after init.  Also called whenever  the applet is restarted. */
     public void start() {  // start or resume execution }

    // Called when the applet is stopped.
     public void stop() { // suspends execution }

    /* Called when applet is terminated.  This is the last  method executed. */
    public void destroy() { // perform shutdown activities }

    // Called when an applet's window must be restored.
    public void paint(Graphics g) { // redisplay contents of window }
}
```

```
// An applet that sets the foreground and background colors and outputs a string.
import java.awt.*;
import java.applet.*;
/*
<applet code="Sample" width=300 height=50>
</applet>
*/

public class Sample extends Applet{
    String msg;

    // set the foreground and background colors.
    public void init() {
                  setBackground(Color.cyan);
                  setForeground(Color.red);
                  msg = "Inside init( ) --";
    }

    // Initialize the string to be displayed.
    public void start() {
                  msg += " Inside start( ) --";
    }

    // Display msg in applet window.
    public void paint(Graphics g) {
                  msg += " Inside paint( ).";
                  g.drawString(msg, 10, 30);
    }
}
```

```
// Using the Status Window.
import java.awt.*;
import java.applet.*;
/*
<applet code="StatusWindow" width=300 height=50>
</applet>
*/

public class StatusWindow extends Applet{
    public void init() {
        setBackground(Color.cyan);
    }

    // Display msg in applet window.
    public void paint(Graphics g) {
        g.drawString("This is in the applet window.", 10, 20);
        showStatus("This is shown in the status window.");
    }
}
```

Use Parameters

```
// Use Parameters
import java.awt.*;
import java.applet.*;
/*
<applet code="ParamDemo" width=300
  height=80>
<param name=fontName value=Courier>
<param name=fontSize value=14>
<param name=leading value=2>
<param name=accountEnabled value=true>
</applet>
*/
```

```
public class ParamDemo extends Applet{
   String fontName;
   int fontSize;
   float leading;
   boolean active;

   // Initialize the string to be displayed.
   public void start() {
               String param;

               fontName = getParameter("fontName");
               if(fontName == null)
                       fontName = "Not Found";

               param = getParameter("fontSize");
               try {
                       if(param != null) // if not found
                               fontSize = Integer.parseInt(param);
                       else
                               fontSize = 0;
               } catch(NumberFormatException e) {
                       fontSize = -1;
               }
```

```
        param = getParameter("leading");

        try {
                if(param != null) // if not found
                        leading = Float.valueOf(param).floatValue();
                else
                        leading = 0;
        } catch(NumberFormatException e) {
                leading = -1;
        }

        param = getParameter("accountEnabled");
                if(param != null)
                        active = Boolean.valueOf(param).booleanValue();
    }

    // Display parameters.
    public void paint(Graphics g) {
        g.drawString("Font name: " + fontName, 0, 10);
        g.drawString("Font size: " + fontSize, 0, 26);
        g.drawString("Leading: " + leading, 0, 42);
        g.drawString("Account Active: " + active, 0, 58);
    }
}
```

Code and Document Bases

```
// Display code and document bases.
import java.awt.*;
import java.applet.*;
import java.net.*;
/*
<applet code="Bases" width=300 height=50> </applet>
*/

public class Bases extends Applet{
    // Display code and document bases.
   public void paint(Graphics g) {
                String msg;

                URL url = getCodeBase(); // get code base
                msg = "Code base: " + url.toString();
                g.drawString(msg, 10, 20);

                url = getDocumentBase(); // get document base
                msg = "Document base: " + url.toString();
                g.drawString(msg, 10, 40);
   }
}
```

showDocument

```
/* Using an applet context, getCodeBase(),
   and showDocument() to display an HTML file.
*/

import java.awt.*;
import java.applet.*;
import java.net.*;

public class ACDemo extends Applet{
   public void start() {
               AppletContext ac = getAppletContext();
               URL url = getCodeBase(); // get url of this applet

               try {
                       ac.showDocument(new URL(url+"Test.html"));
               } catch(MalformedURLException e) {
                       showStatus("URL not found");
               }
   }
}
```

Setting a color

- To use a color, we tell our Graphics g what color we want:

```
g.setColor(Color.RED);
```

- g will remember this color and use it for everything until we tell it some different color

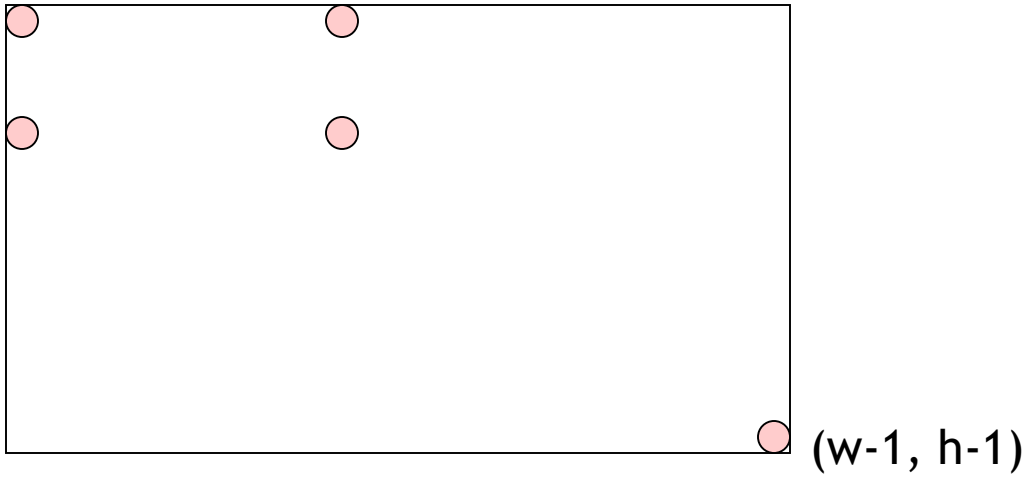
The paint method so far

```
public void paint(Graphics g) {
    g.setColor(Color.BLUE);
    ...draw a rectangle...
    g.setColor(Color.RED);
    ...draw another rectangle...
  }
}
```

Pixels

- A pixel is a picture (pix) element
  - one pixel is one dot on your screen
  - there are typically 72 to 90 pixels per inch
- java.awt measures everything in pixels

Java's coordinate system



- Java uses an (x, y) coordinate system
- (0, 0) is the top left corner
- (50, 0) is 50 pixels to the right of (0, 0)
- (0, 20) is 20 pixels down from (0, 0)
- (w - 1, h - 1) is just inside the bottom right corner, where w is the width of the window and h is its height

Drawing rectangles

- There are two ways to draw rectangles:
- g.drawRect(***left*** , ***top*** , ***width*** , ***height***);

- g.fillRect(***left*** , ***top*** , ***width*** , ***height***);

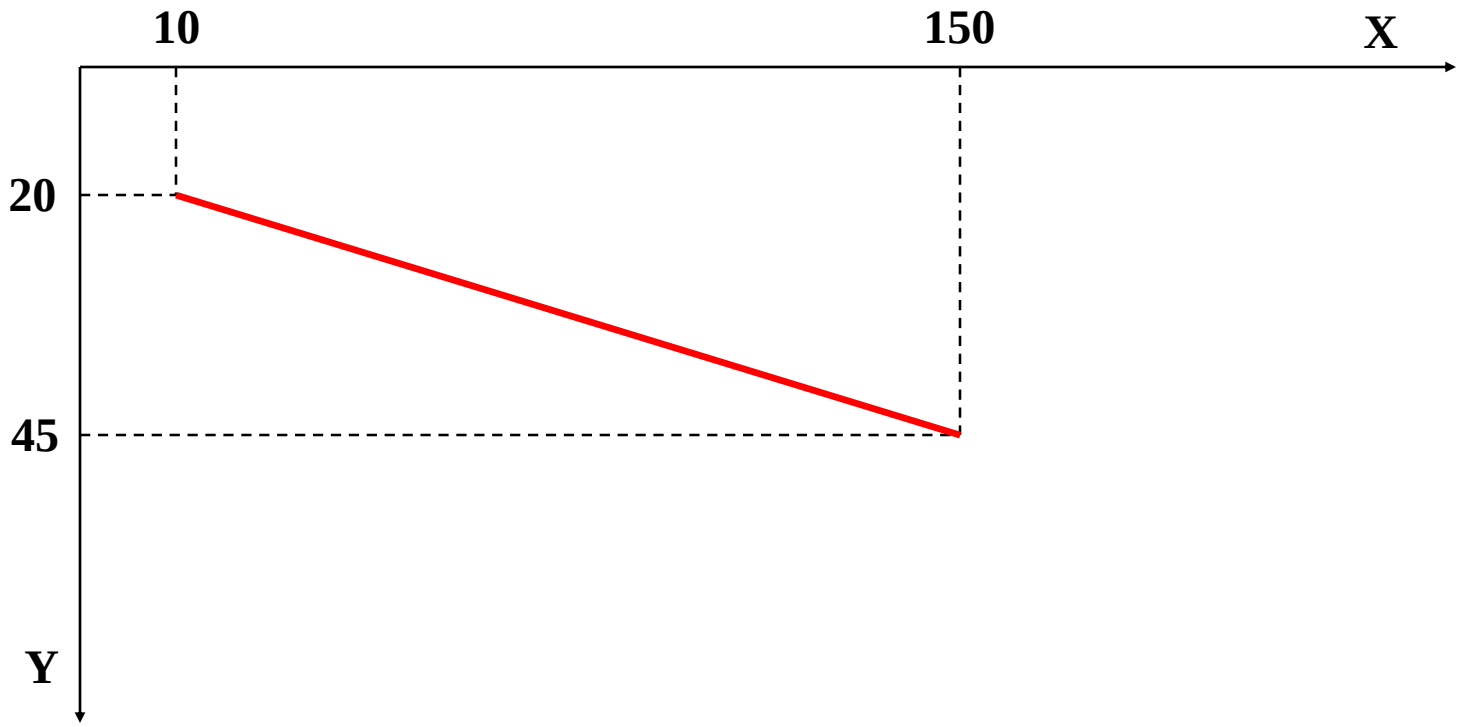
The complete applet

```
import java.applet.Applet;
import java.awt.*;

public class Drawing extends Applet {

    public void paint(Graphics g) {

        g.setColor(Color.BLUE);
        g.fillRect(20, 20, 50, 30);
        g.setColor(Color.RED);
        g.fillRect(50, 30, 50, 30);
    }
}
```

Some more java.awt methods

- g.drawLine(***x1*** , ***y1*** , ***x2*** , ***y2***);
- g.drawOval(***left*** , ***top*** , ***width*** , ***height***);
- g.fillOval(***left*** , ***top*** , ***width*** , ***height***);
- g.drawRoundRect(***left*** , ***top*** , ***width*** , ***height***);
- g.fillRoundRect(***left*** , ***top*** , ***width*** , ***height***);
- g.drawArc(***left*** , ***top*** , ***width*** , ***height*** , ***startAngle*** , ***arcAngle***);
- g.drawString(***string*** , ***x*** , ***y***);

Drawing a Line

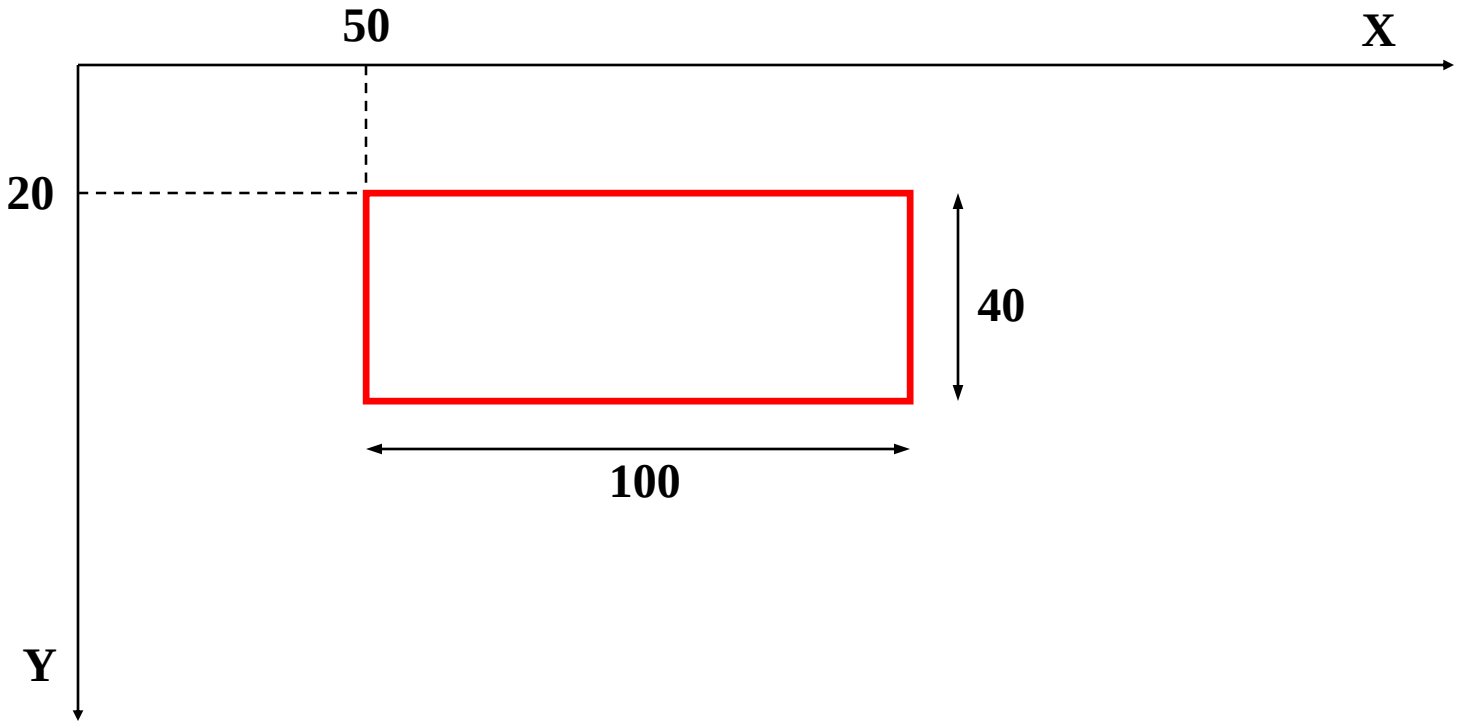



```
page.drawLine (10, 20, 150, 45);
```

or

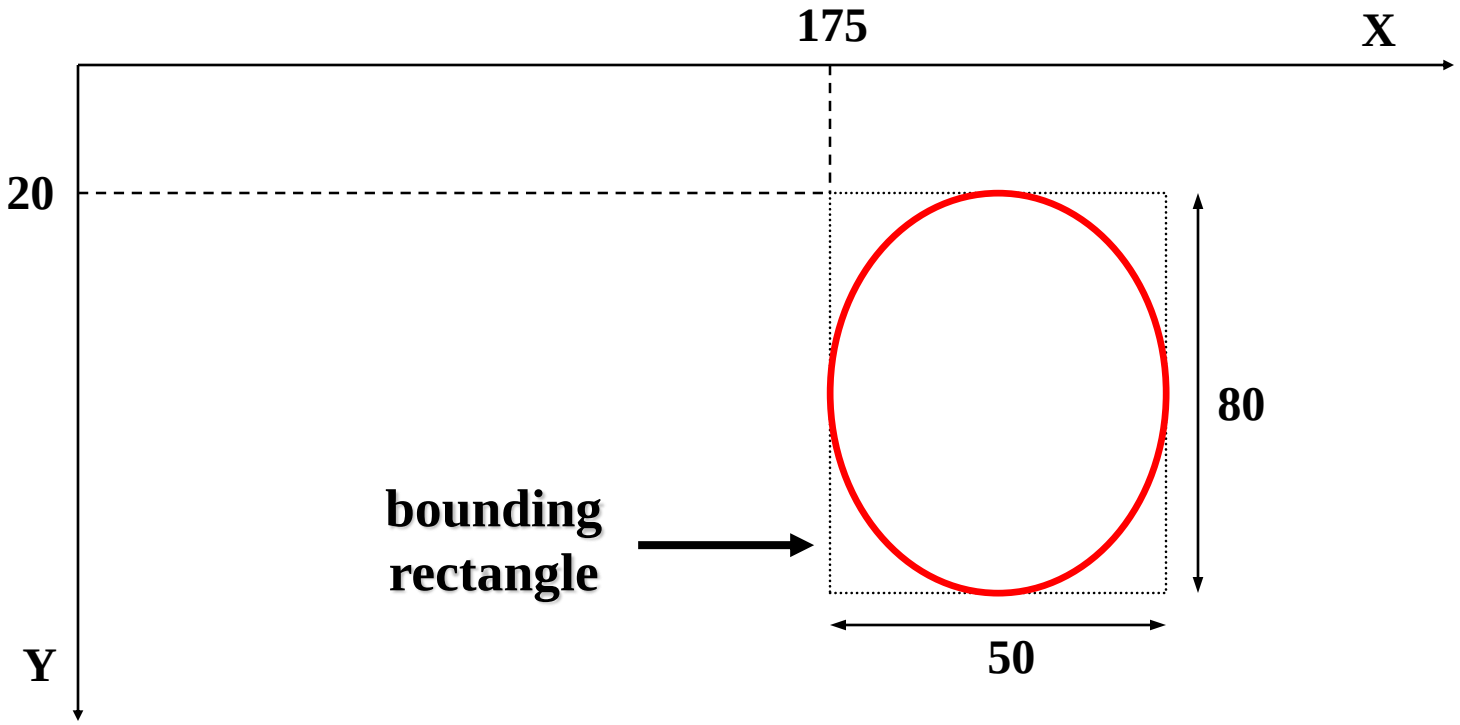
```
page.drawLine (150, 45, 10, 20);
```

Drawing a Rectangle




```
page.drawRect (50, 20, 100, 40);
```

Drawing an Oval




```
page.drawOval (175, 20, 50, 80);
```