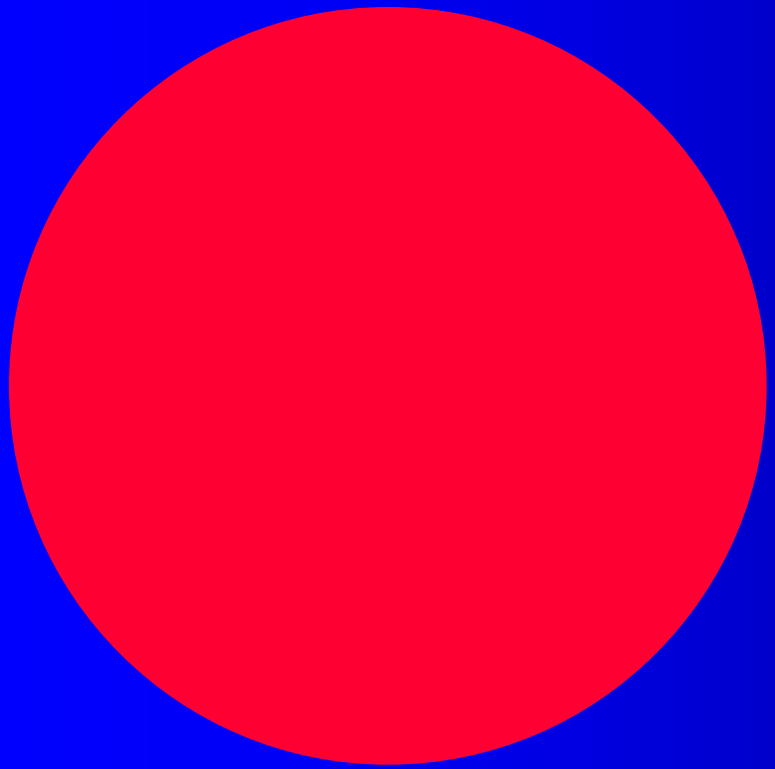
Lecture 33

- Covers
  - Applet examples
    - Calculating coordinates and sizes
    - Rectangles and polygons

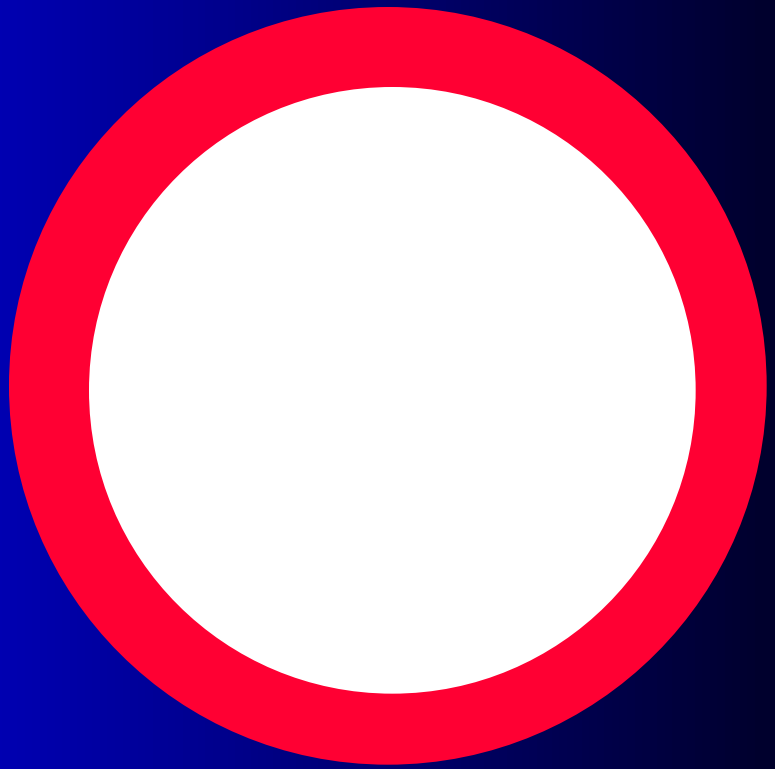
Example 1 - The Bull's Eye

- An applet to draw the bull's eye of *n* rings (for example *n* = 5)

Example 1 - The Bull's Eye



draw filled outer circle

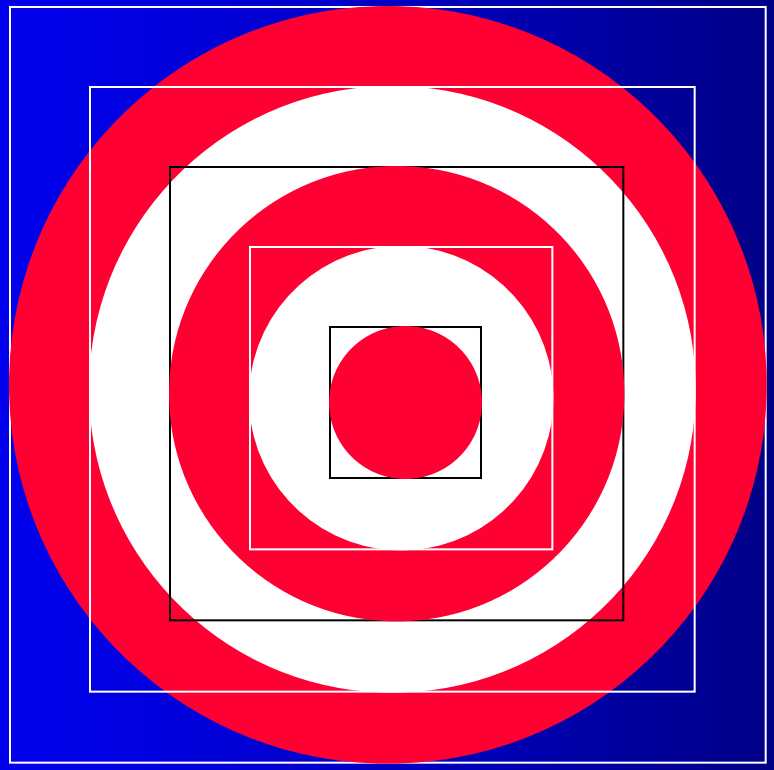


draw next layer

Example 1 - The Bull's Eye

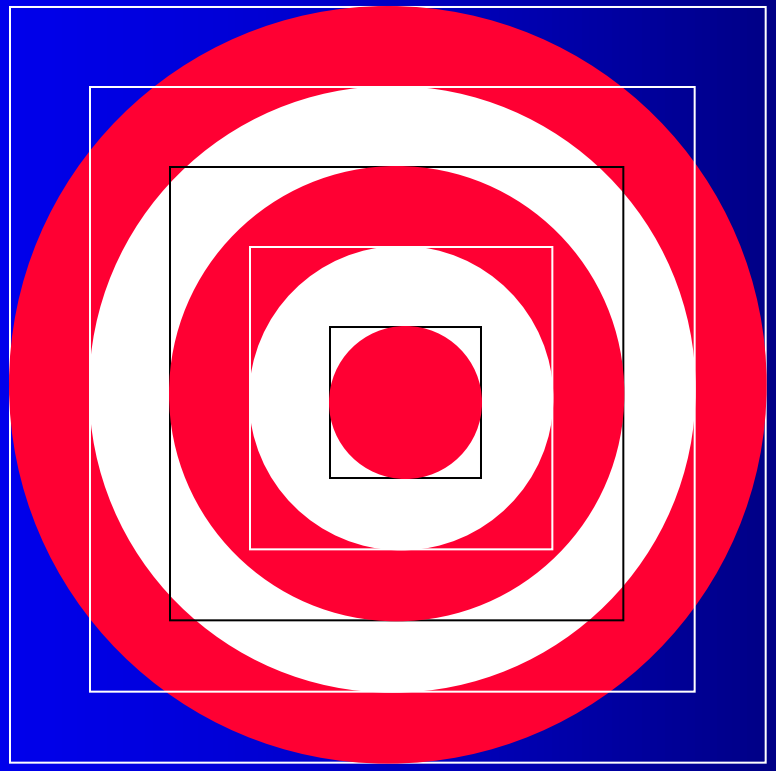
What are the specifications of each circle?

Example 1 - The Bull's Eye



Think about the bounding boxes

Example 1 - The Bull's Eye

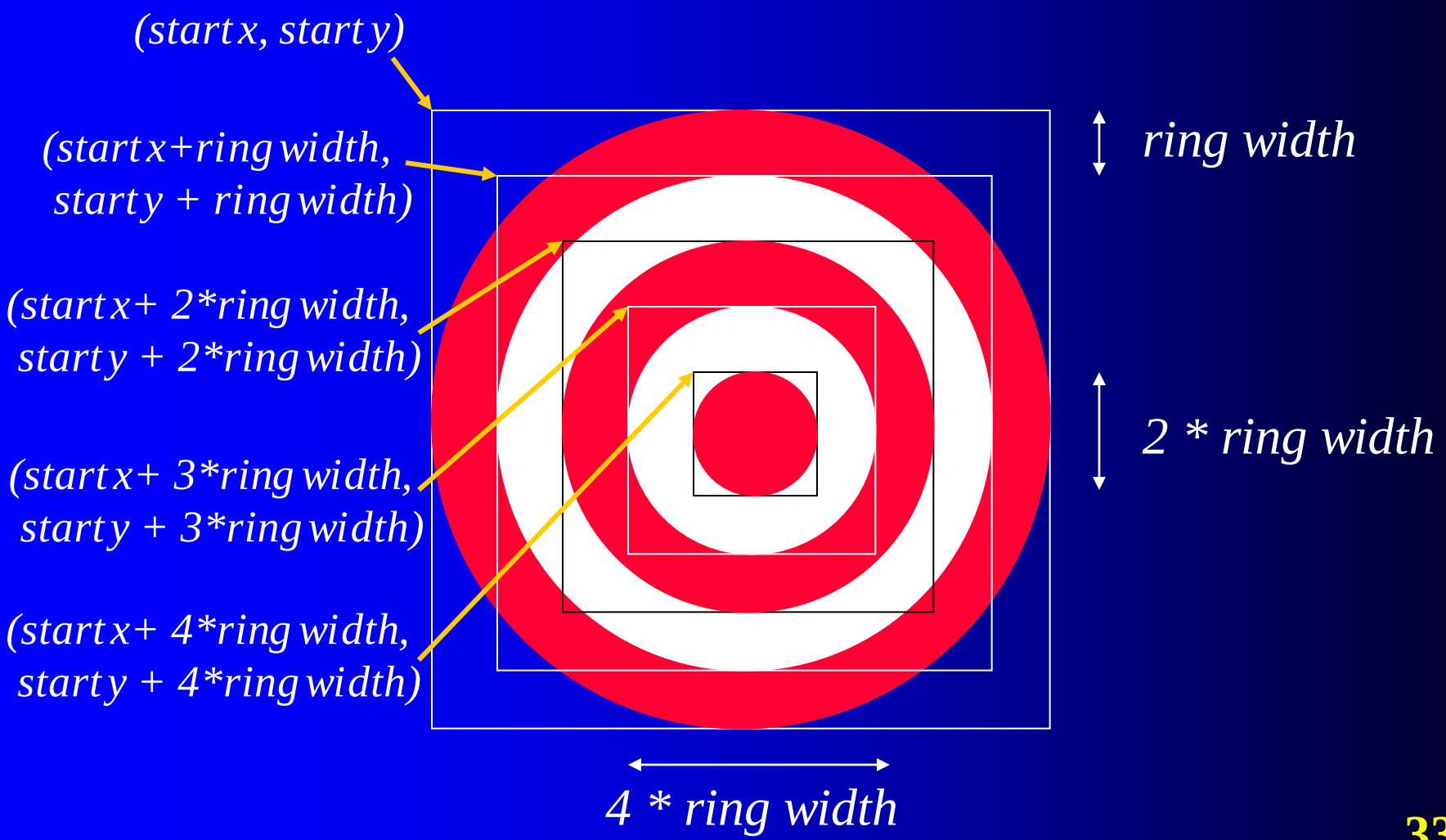


ring width

*2 * ring width*

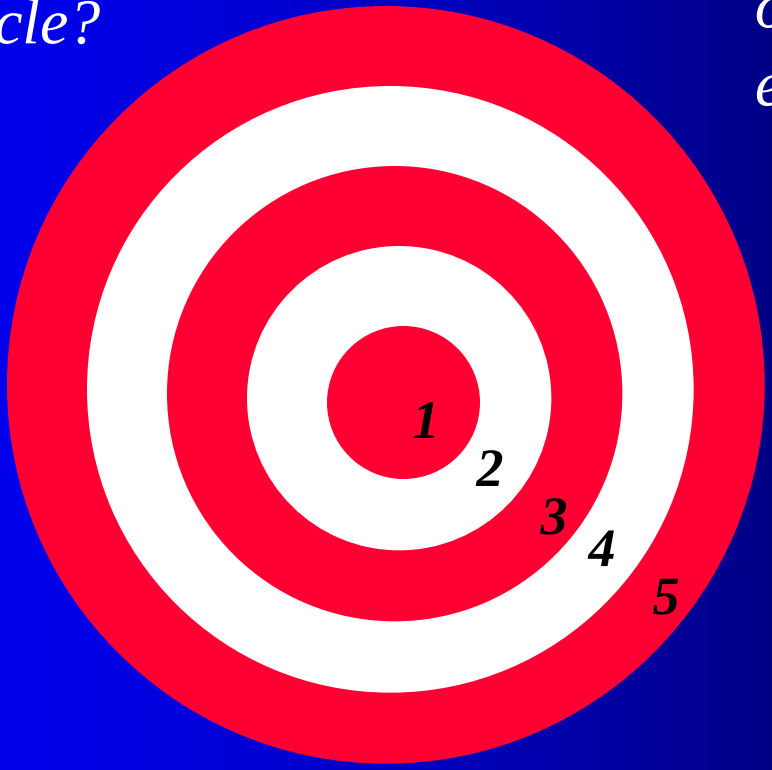
*4 * ring width*

Example 1 - The Bull's Eye



Example 1 - The Bull's Eye

How do we choose the colour of each circle?



Numbering from centre, odd numbers are red, even numbers are even

Example 1 - The Bull's Eye

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

public class Bullseye extends Applet
{
    public void paint(Graphics g)
    {
        setBackground(Color.orange);
        int n = 5;

        for (int i = n; i > 0; --i)
        {
            if (i%2 == 1)
                g.setColor(Color.red);
            else
                g.setColor(Color.white);

            g.fillOval(20+(n-i)*10, 20 + (n-i)*10, i*2*10, i*2*10);
        }
    }
}
```

10 = ring width

(20,20) = starting coordinates

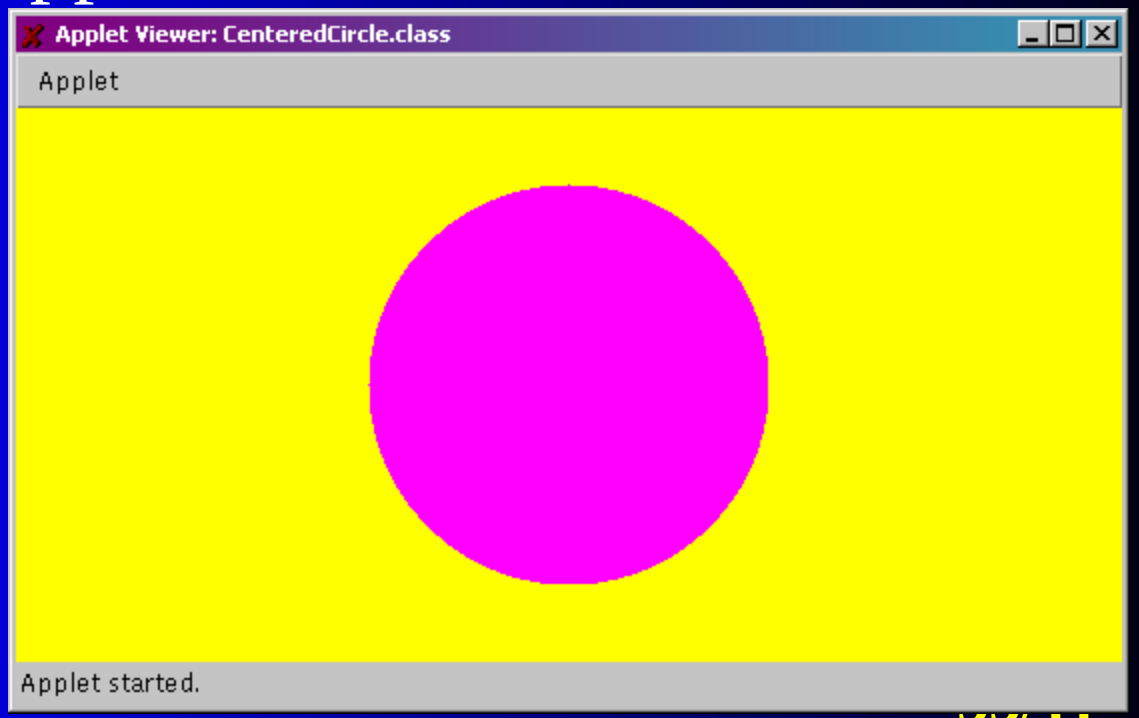
Example 1 - The Bull's Eye

```
public void paint(Graphics g)
{
    int startXcoord = 20;
    int startYcoord = 20;
    int ringWidth = 10;
    setBackground(Color.orange);
    int n = 5;

    for (int i = n; i > 0; --i)
    {
        if (i%2 == 1)
            g.setColor(Color.red);
        else
            g.setColor(Color.white);

        g.fillOval(startXcoord + (n-i)*ringWidth,
                   startYcoord + (n-i)*ringWidth,
                   i*2*ringWidth, i*2*ringWidth);
    }
}
```

Replace literals with variables

Example 2 – Centred circle

- Write an applet to display a circle that is always in the middle of the applet's screen even when the applet is resized



Getting the size of an applet

- To do this, we need to be able to obtain the size of the applet at runtime
- The width and height of an applet can be obtained by

  getHeight( )

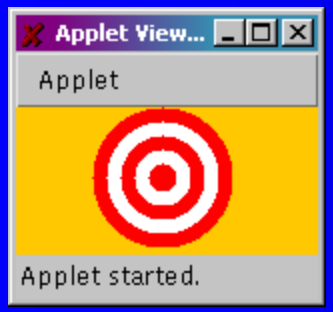
  getWidth( )

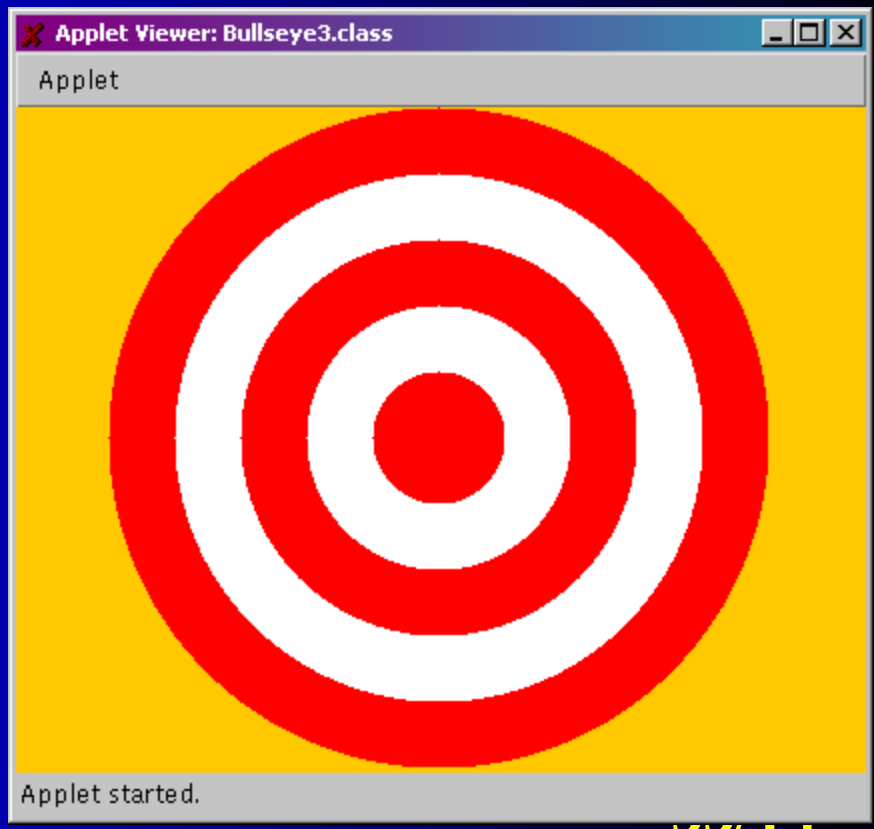
Example 2 – Centred circle

```
import java.applet.*;
import java.awt.*;
public class CentredCircle extends Applet
{
    public void paint(Graphics g)
    {
        setBackground(Color.yellow);
        g.setColor(Color.magenta);
        int xc = getWidth( );
        int yc = getHeight( );
        int radius = 100;
        g.fillOval( xc/2 - radius, yc/2 - radius, 2 * radius, 2 * radius );
    }
}
```

Example 1 - revisited

- Change the Bull's Eye applet so that the Bull's Eye fills the applet screen (as much as possible)





Example 1 - revisited

```
public void paint(Graphics g)
{
    int n = 5;
    int width = getWidth( );
    int height = getHeight( );
    int minDimension = width < height? width: height;
    int startXcoord = (width - minDimension) / 2;
    int startYcoord = (height - minDimension) / 2;
    int ringWidth = minDimension / (n * 2);
    setBackground(Color.orange);
    for (int i = n; i > 0; --i)
    {
        if (i%2 == 1)
            g.setColor(Color.red);
        else
            g.setColor(Color.white);

        g.fillOval(startXcoord + (n-i)*ringWidth, startYcoord + (n-i)*ringWidth,
                                        i*2*ringWidth, i*2*ringWidth);
    }
}
```

Constructing polygons

- A polygon can be constructed in 2 ways
- First method
  - Define 2 arrays representing the x- and y-coordinates of its vertices
  - Call the following constructor to create the polygon

```
Polygon(int[ ] xCoords, int[ ] yCoords,
        int nrPoints)
```

Constructing polygons

- Second method
  - Call the following constructor to create a new (empty) polygon

    Polygon( )
  - Call the following method to add the points to the polygon

    addPoint(int x, int y)

Displaying polygons

- To display a solid polygon, use

  fillPolygon(Polygon p)

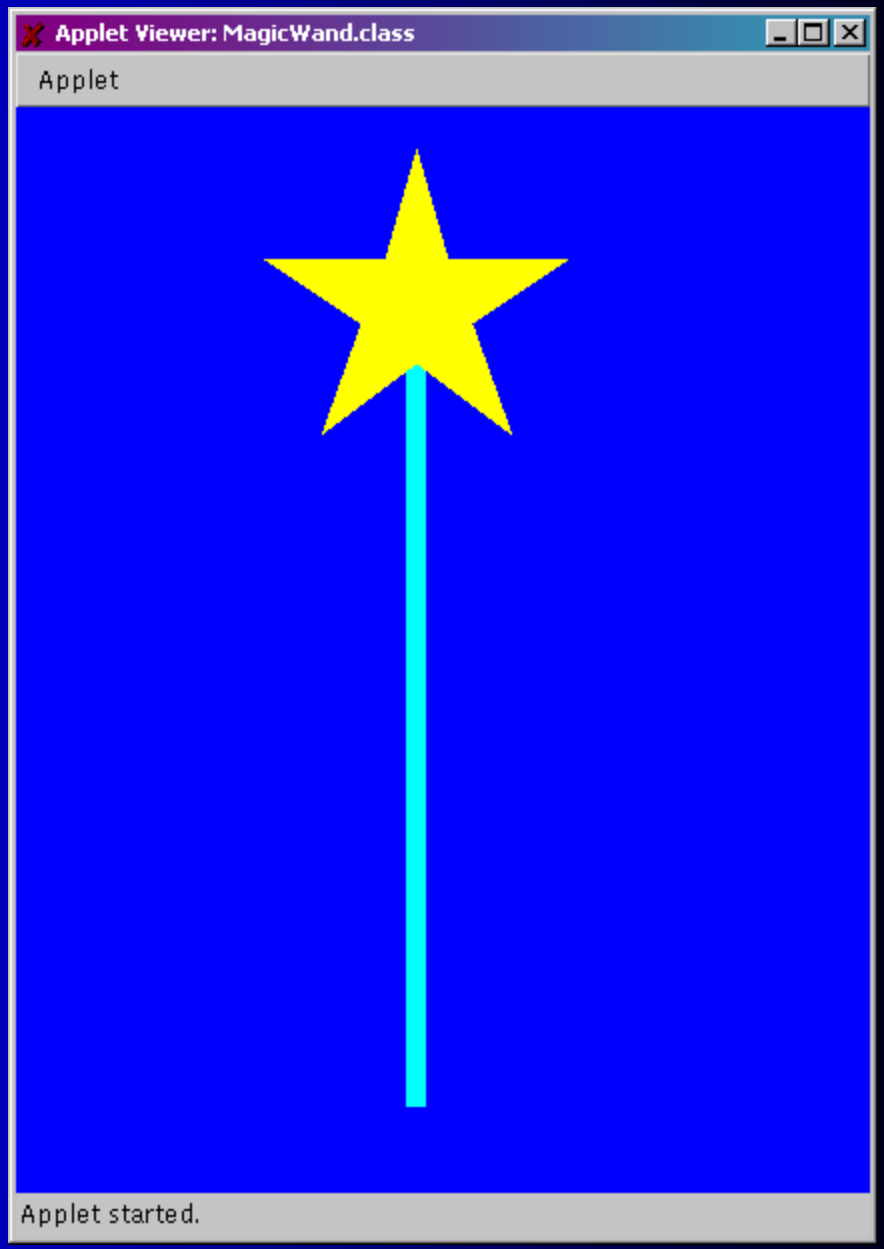
  of class Graphics
- To draw the edges of a polygon, use

  drawPolygon(Polygon p)
- To draw a polyline, except the last one, use

  drawPolyline(int[ ] xcoords, int[ ] ycoords,
  int numberPoints)

Example 3 – Magic wand

- Draw a magic wand (a star on a stick)



Example 3 – Magic wand

- Given that the following are x- and y-coordinates of a star with centre at (0,0) and radius 100

```
int[ ] xs = {   0,  20,  95, 35, 60,   0, -60, -35, -95, -20};
int[ ] ys = {-100, -30, -30, 11, 80, 35,  80,  11, -30, -30};
```

Example 3 – Magic wand

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

public class MagicWand extends Applet
{
    private Polygon star;
    private Polygon stick;

    public void init( )
    {
        int xc = 200;          // xc, yc define the centre of the star
        int yc = 100;          // and the top of the stick
        int radius = 80;       // radius of the star

        int width = 10;        // width of the stick
        int length = 400;      // length of the stick

        //  xs and ys below represent the coordinates of the 10 vertices
        //  of the stars whose centre is at (0, 0) and of radius 100
        int[ ] xs = {      0,   20,  95, 35, 60,  0, -60, -35, -95, -20 };
        int[ ] ys = { -100, -30, -30, 11, 80, 35,  80,  11, -30, -30 };
```

Example 3 – Magic wand

```
//  create the star of radius radius with centre at xc, yc
for( int i = 0; i <= 9; i++ )
{
    xs[ i ] = xc + (int) (xs[ i ] * radius / 100.0);
    ys[ i ] = yc + (int) (ys[ i ] * radius / 100.0);
}
star = new Polygon(xs, ys, 10);

// create the stick (use the second method)
stick = new Polygon( );

stick.addPoint(xc - width / 2, yc);
stick.addPoint(xc + width / 2, yc);
stick.addPoint(xc + width / 2, yc + length);
stick.addPoint(xc - width / 2, yc + length);

setBackground(Color.blue);

} // init
```

Example 3 – Magic wand

```
    public void paint(Graphics g)
    {
        // display the stick
        g.setColor(Color.cyan);
        g.fillPolygon(stick);

        // display the star
        g.setColor(Color.yellow);
        g.fillPolygon(star);
    }
}
```

The Rectangle class

- We can create and draw Rectangle objects
- With an existing Rectangle object, we can draw the object, move it, or change its size and draw it again

```
Rectangle box = new Rectangle(50, 50, 80, 80);
```

The Rectangle class

- Create Rectangle objects

  Rectangle(int x, int y, int width, int height)

- Move a rectangle to a new position

  translate(int dx, int dy)

  where *dx* and d*y* are the x- and y- displacements

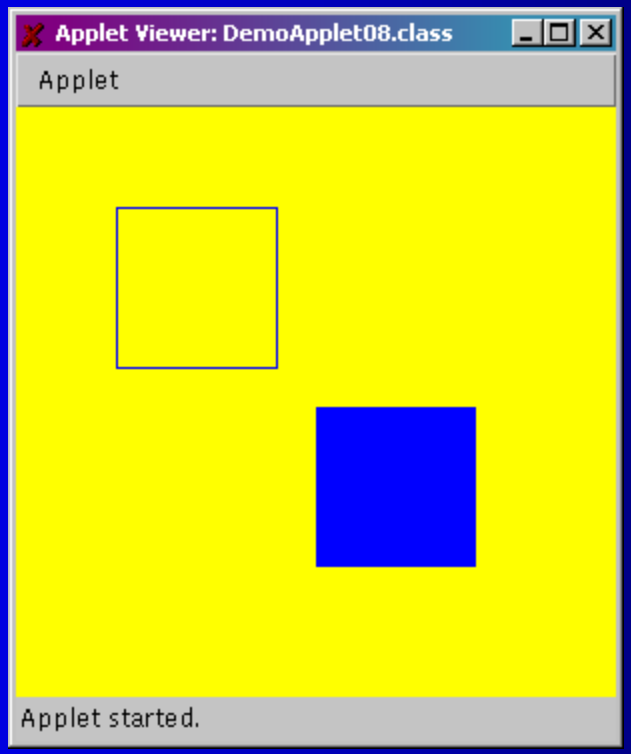
- Reset position and size

  setLocation(int x, int y)

  setSize(int width, int height)

Example 4 – Rectangle objects

- An applet to create a Rectangle object, display it, then move it and display it again



Example 4 – Rectangle objects

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

public class DemoApplet extends Applet
{
    public void paint(Graphics g)
    {
        setBackground(Color.yellow);
        g.setColor(Color.blue);

        Rectangle box = new Rectangle(50, 50, 80, 80);

        Graphics2D g2D = (Graphics2D) g;
        g2D.draw(box);

        // move the object 100 pixels to the right and 100 pixels down
        box.translate(100, 100);

        g2D.fill(box);
    }
}
```

Graphics and Graphics2D objects

- Consider the statement

  ```
  Graphics2D g2D = (Graphics2D) g;
  ```

- g is a Graphics object - essentially it represents the applet screen together with information for graphics rendering (e.g. color, font)
- The above statement takes g and casts it into a Graphics2D object, and refers to it as g2D
- g2D has all the capabilities of g, plus some additional ones defined in the Graphics2D class
- For example, g2D (but *not* g) can respond to a message such as

  ```
  draw(box);
  ```

Next lecture

- Life cycle of an applet
- Fonts and font metrics