What Is an Applet?

- A mini-application
- Distributed along with Web pages
- Run under a browser at the viewer's site
- Run under an applet viewer
- Is distributed in Bytecode form

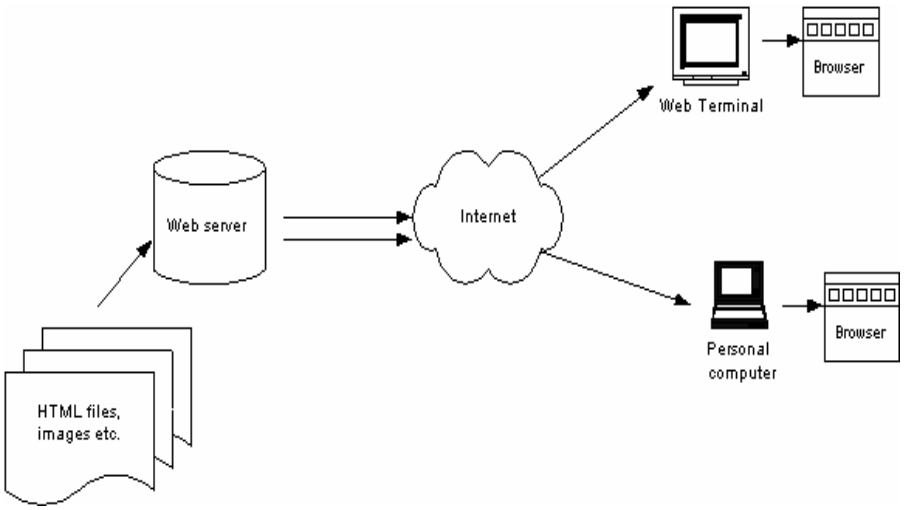
Applets Differ from Applications

- Applets don't have a `main` method
- Applets are invoked differently
- Applets are subject to more security constraints
- Applets are not in control of their own destiny

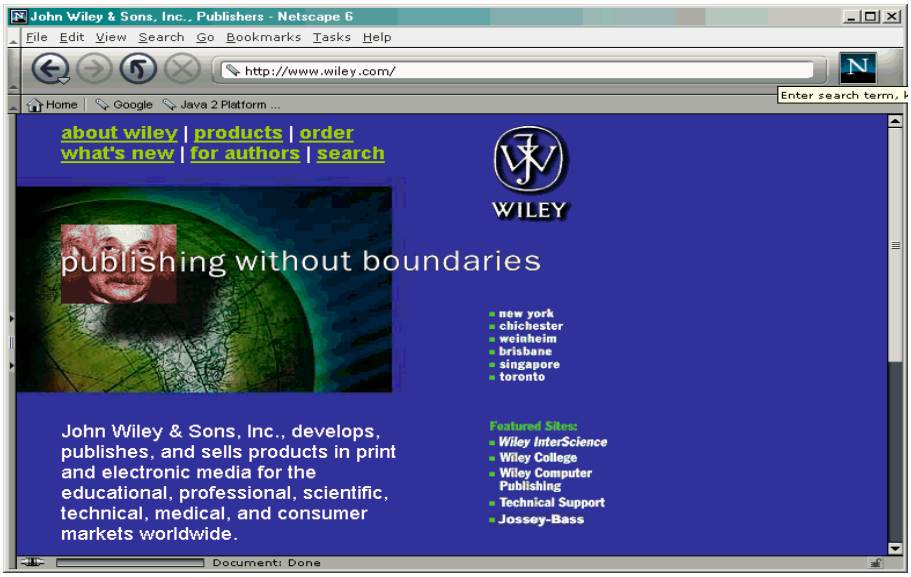
Applets Differ from Applications

- Applets do not have constructors
- Initializations are done in method `init`
- Applets are themselves containers in a window
- Applets must be explicitly derived from class `Applet` or `JApplet`

Web Browsers Accessing a Web Server



A Web Browser



Brief Introduction to HTML

- Text and tags:
  Java is an object-oriented
  programming language
- Browser renders the tags:
  Java is an *object-oriented* programming language
- Bulleted list (like this one) is defined by tags

  . . .
  . . .
  . . .

- Use <> for <> symbols

Images, Links and Applets

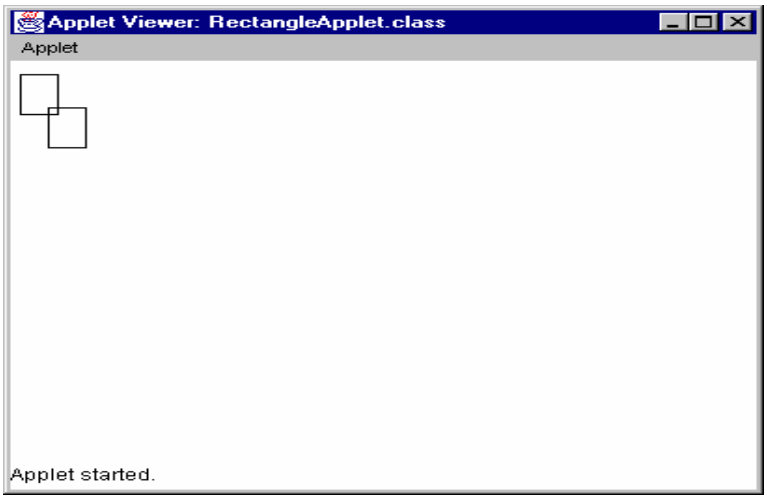
- Image tag has attributes: source, size, alternate text

- Link tag has attribute for link, body for linked text
  Java is an . . .
- Applets need class for applet code and size:
  <applet code="HamsterApplet.class" width="640" height="480">

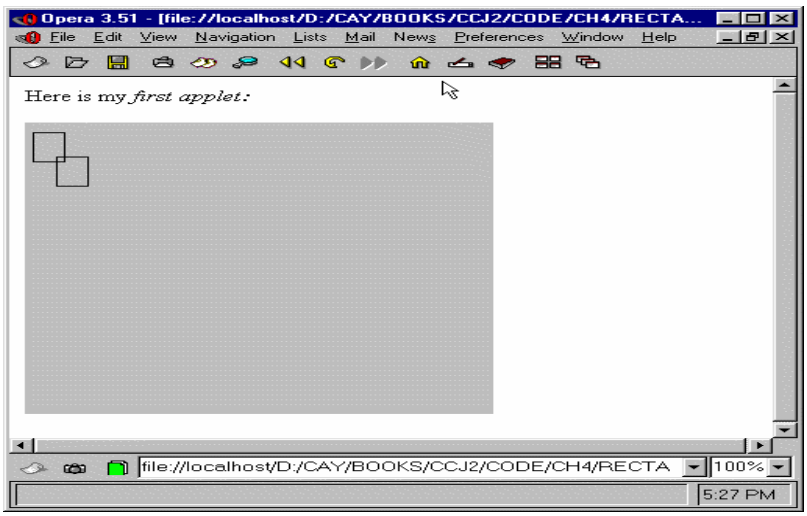
Viewing an Applet

- Make one or more Java source files to implement your applet
- One of the source files must define the applet class
- Compile the source files into class files
- Make an HTML file with the applet tag that references the applet class
- Run appletviewer myapplet.html
- Or load the HTML file into a Java 2 compliant browser

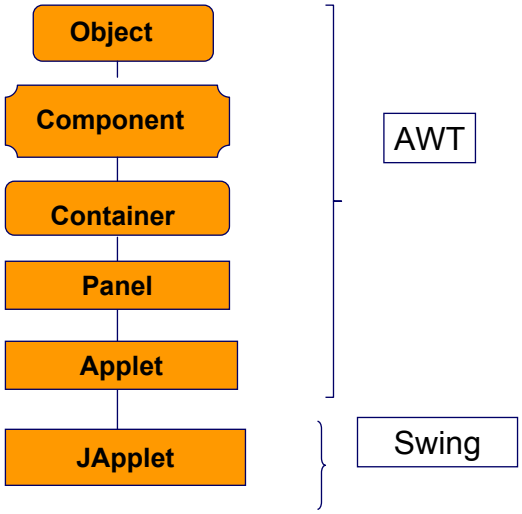
The RectangleApplet in the Applet Viewer



The RectangleApplet in a Browser



Hierarchy of Component Classes



We Use Applets

- All browsers can run Applets
- Older browsers can only run Applets
- New browsers can run both Applets and JApplets
- Thus, we use Applets
- Window components are the same with the “J” removed

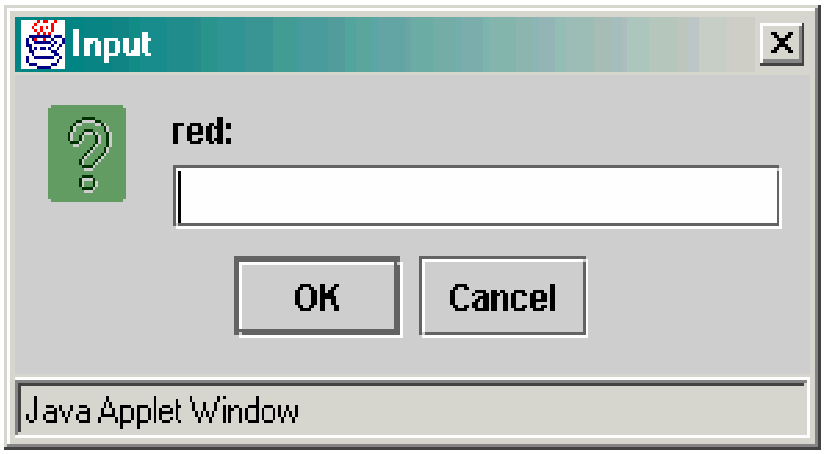
Applet Class Outline

```
import java.applet.Applet;  // Applet class
import java.awt.*;          // Layout manager
import java.awt.event.*;    // Event classes

public class MyApplet extends Applet implements ActionListener
{
    public void init()
    {

    }

    public void actionPerformed(ActionEvent   event)
                                                // Event handler method
  {

  }
}
```

Reading Text Input

- Call JOptionPane.showInputDialog in the constructor
- Dialog has warning label--security feature
- Set instance variables with the input results
- Read them in the paint method

Applet Dialog with Warning Label



Exemplo 1: Fatorial

```
// Applet Factorial computes the factorial of
// its input and stores it in a variable of
// type int, which is displayed on the
// screen.

import java.applet.Applet;   // Applet class
import java.awt.*;           // Layout manager
import java.awt.event.*;     // Event classes
public class FactInt extends Applet implements
  ActionListener {…}
```

Fatorial :`actionPerformed` Method

```
public void actionPerformed(ActionEvent
  event) // Event handler method
{
  int value;
  value =
    Integer.parseInt(inputField.getText());
  inputField.setText("");
  outLabel.setText(value + " factorial is ”
   + factorial(value));
}
```

method call

Fatorial: `factorial` Method

```
private int factorial(int n)
// Assumption:  n is not negative.
{
  if (n == 0)
    return 1;       // base case
  else              // general case
    return (n * factorial((n-1)));
}
```

Fatorial: Interface Gráfica

```
// Setting up a button, label, and input
// field
private static TextField inputField;
private static Label label;
private static Label outLabel;
private static Button button;
```

Note the lack of “J”s

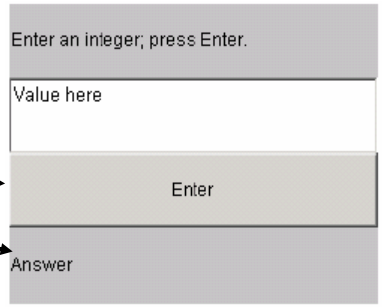
Fatorial: `init` Method

```
public void init()
{ // Instantiate components
  label = new Label("Enter an integer; ”
    + “press Enter.");
  outLabel = new Label("Answer");
  button = new Button("Enter");
  button.addActionListener(this);
  inputField = new
    TextField("Value here");
```

Note

Fatorial: `init` Method

```
// Add components
add(label);
add(inputField);
add(button);
add(outLabel);
// Specify a layout
// manager for the
// window object
setLayout(new GridLayout(4,1)); // 4 linhas
                                // 1 coluna
}
```



O método `add` is aplicado ao próprio objeto *Factorial* *(está implícito)*

Gerência do Applet

Carregando o Applet

- O applet *incializa*
- E começa a rodar.

Usuário "sai" da página

- O applet pode parar

Usuário volta

- O applet pode voltar a rodar

Antes de fechar o browser

- O applet pode terminar suas atividades

```
public void init() {
   buffer = new StringBuffer();
   addItem("initializing... ");
}

public void start() {
   addItem("starting... ");
}

public void stop() {
   addItem("stopping... ");
}

public void destroy() {
   addItem("unloading...");
}
```

Exemplo 2: RectangleApplet.java

```
1  import java.applet.Applet;
2  import java.awt.Graphics;
3  import java.awt.Graphics2D;
4  import java.awt.Rectangle;
5
6  /**
7    An applet that draws two rectangles.
8  **/
9  public class RectangleApplet extends Applet
10 {
```

...Continue

Exemplo 2: RectangleApplet.java

```
11  public void paint(Graphics g)
12  {
13     // recover Graphics2D
14
15     Graphics2D g2 = (Graphics2D)g;
16
17     // construct a rectangle and draw it
18
19    Rectangle cerealBox = new Rectangle(5, 10, 20, 30);
20     g2.draw(cerealBox);
21
22  // move rectangle 15 units sideways and 25 units down
23
```

...Continue

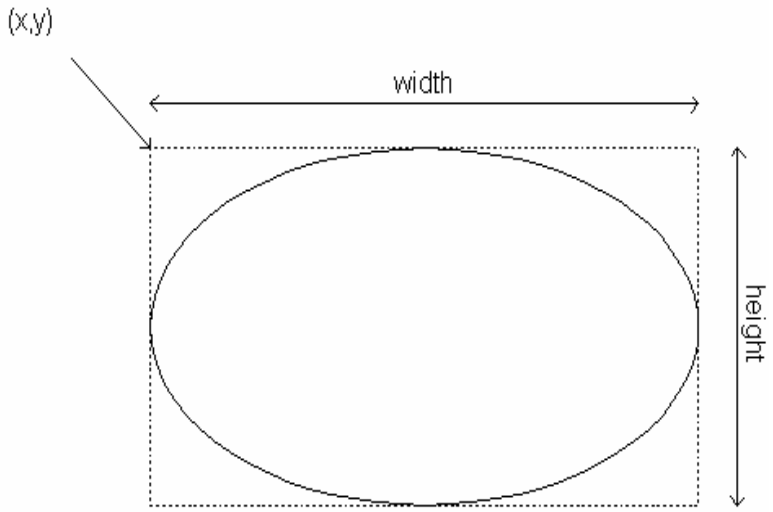
Exemplo 2: RectangleApplet.java

```
    cerealBox.translate(15, 25);

    // draw moved rectangle

    g2.draw(cerealBox);
  }
}
```

Graphical Shapes

- Shape classes Ellipse2D.Double, Line2D.Double, etc.
- We won't use the .Float classes
- These classes are inner classes--doesn't matter to us except for the import statement:
  import java.awt.geom.Ellipse2D; // no .Double
- Must construct and draw the shape
  Ellipse2D.Double easterEgg = new Ellipse2D.Double(5, 10, 15, 20);
  g2.draw(easterEgg)

Specifying an Ellipse



Lines and Points

- Line2D.Double segment = new Line2D.Double(x1, x2, y1, y2);
- More object-oriented to use Point2D.Double for the end points:
  Point2D.Double from = new Point2D.Double(x1, y1);
  Point2D.Double to = new Point2D.Double(x2, y2);
  Line2D.Double segment = new Line2D.Double(from, to);
- Draw thick lines:
  g2.setStroke(new BasicStroke(4.0F)); // 4 pixels

Colors

- Specify red, green, blue between 0.0F and 1.0F
  Color magenta = new Color(1.0F, 0.0F, 1.0F)

- Standard colors
  Color.black
  Color.yellow
  Color.pink
  . . .
- Set color in graphics context:
  g2.setColor(Color.pink);
- Then draw or fill shapes
  g2.fill(easterEgg);

Text and Fonts

- Specify text and base point:
  g2.drawString("Applet", 50, 100);
- Font object has
  - face name (Serif, SansSerif, Monospaced, ...)
  - style (Font.PLAIN, Font.BOLD, Font.ITALIC)
  - point size (12 point = normal size)

- g2.setFont(new Font("Serif", Font.BOLD, 36));

Common Fonts

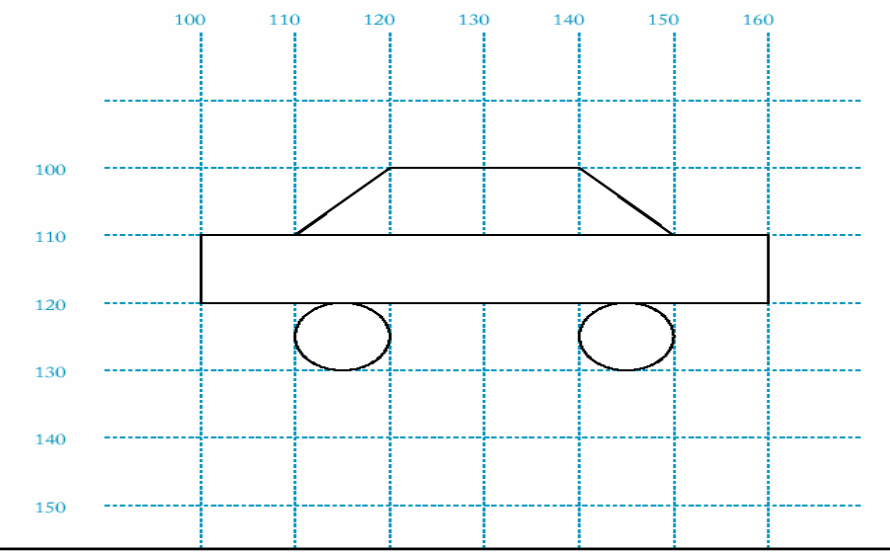
Helvetica

Times Roman

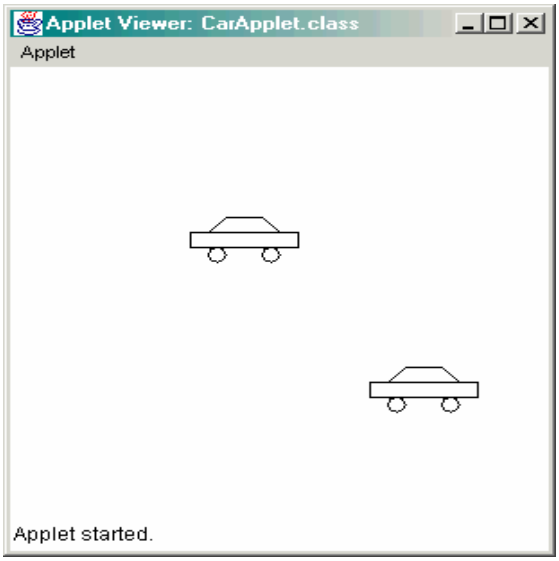
Courier

Basepoint and Baseline

Plan Complex Shapes with Graph Paper



The Car Drawer Applet



Exemplo 3: File CarApplet.java

```
import java.applet.Applet;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Rectangle;

/**
  An applet that draws two rectangles.
*/
public class RectangleApplet extends Applet
{
```

...Continue

Exemplo 3: File CarApplet.java

```
  public void paint(Graphics g)
  {
    // recover Graphics2D

    Graphics2D g2 = (Graphics2D)g;

    // construct a rectangle and draw it

    Rectangle cerealBox = new Rectangle(5, 10, 20, 30);
    g2.draw(cerealBox);

```

...Continue

Exemplo 3: File CarApplet.java

```
// move rectangle 15 units sideways and 25 units down

    cerealBox.translate(15, 25);

    // draw moved rectangle

    g2.draw(cerealBox);
  }
}
```

Exemplo3: File Car.java

```
import java.awt.Graphics2D;
import java.awt.geom.Ellipse2D;
import java.awt.geom.Line2D;
import java.awt.geom.Point2D;
import java.awt.geom.Rectangle2D;

/**
  A car shape that can be positioned anywhere on the screen.
*/
public class Car
{
```

…Continue

```
12  /**
13    Constructs a car with a given top left corner
14    @param x the x coordinate of the top left corner
15    @param y the y coordinate of the top left corner
16  */
17  public Car(double x, double y)
18  {
19    xLeft = x;
20    yTop = y;
21  }
22
```

...Continue

```
23  /**
24    Draws the car
25    @param g2 the graphics context
26  */
27  public void draw(Graphics2D g2)
28  {
29   Rectangle2D.Double body
30    = new Rectangle2D.Double(xLeft, yTop + 10, 60, 10);
31   Ellipse2D.Double frontTire
32    =  new  Ellipse2D.Double( xLeft + 10,yTop + 20, 10, 10);
33   Ellipse2D.Double rearTire
34    = new Ellipse2D.Double(xLeft + 40,  yTop + 20,  10,  10);
```

```
35
36      // the bottom of the front windshield
37      Point2D.Double r1
38        = new Point2D.Double(xLeft + 10, yTop + 10);
39      // the front of the roof
40      Point2D.Double r2
41        = new Point2D.Double(xLeft + 20, yTop);
42      // the rear of the roof
43      Point2D.Double r3
44        = new Point2D.Double(xLeft + 40, yTop);
45      // the bottom of the rear windshield
46      Point2D.Double r4
47        = new Point2D.Double(xLeft + 50, yTop + 10);
48
```

```
49      Line2D.Double frontWindshield
50        = new Line2D.Double(r1, r2);
51      Line2D.Double roofTop
52        = new Line2D.Double(r2, r3);
53      Line2D.Double rearWindshield
54        = new Line2D.Double(r3, r4);
55
56      g2.draw(body);
57      g2.draw(frontTire);
58      g2.draw(rearTire);
59      g2.draw(frontWindshield);
60      g2.draw(roofTop);
61      g2.draw(rearWindshield);
62    }
63
64    private double xLeft;
65    private double yTop;
66  }
```

File ColorApplet.java

```
1import java.applet.Applet;
2import java.awt.Color;
3import java.awt.Graphics;
4import java.awt.Graphics2D;
5import java.awt.Rectangle;
6import javax.swing.JOptionPane;
7
8/**
9  An applet that lets a user choose a color by speci
   fying
10  the fractions of red, green, and blue.
11*/
```

```
12public class ColorApplet extends Applet
13{
14  public ColorApplet()
15  {
16     String input;
17
18     // ask the user for red, green, blue values
19
20     input = JOptionPane.showInputDialog("red:");
21     float red = Float.parseFloat(input);
22
23     input = JOptionPane.showInputDialog("green:"
   );
24     float green = Float.parseFloat(input);
```

```
25
26      input = JOptionPane.showInputDialog("blue:");
27      float blue = Float.parseFloat(input);
28
29      fillColor = new Color(red, green, blue);
30   }
31
32   public void paint(Graphics g)
33   {
34      Graphics2D g2 = (Graphics2D)g;
35
36      // select color into graphics context
37
38      g2.setColor(fillColor);
```

```
39
40      // construct and fill a square whose center
   is
41      // the center of the window
42
43      Rectangle square = new Rectangle(
44         (getWidth() - SQUARE_LENGTH) / 2,
45         (getHeight() - SQUARE_LENGTH) / 2,
46         SQUARE_LENGTH,
47         SQUARE_LENGTH);
48
49      g2.fill(square);
50   }
51
```

```
52   private static final int SQUARE_LENGTH = 1
   00;
53
54   private Color fillColor;
55}
56
```

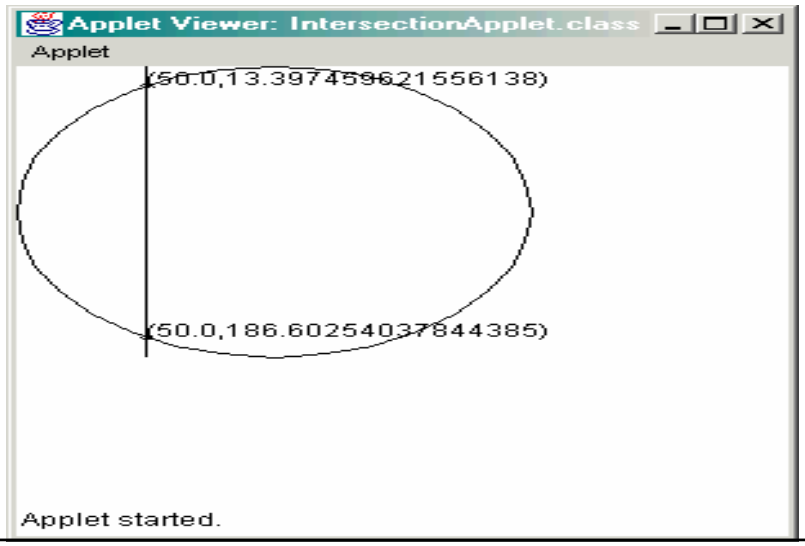
Comparing Visual and Numerical Information

- Compute intersection between circle and vertical line
- Circle has radius r = 100 and center (a, b) = (100, 100)

$$y = b \pm \sqrt{r^2 - (x - a)^2}$$

- Line has constant x value
- Intersection points are
  Plot circle, line, computed intersection points
- If the points are correct, then the visual and numerical results are the same

Intersection of a Line and a Circle



File IntersectionApplet.java

```
import java.applet.Applet;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.geom.Ellipse2D;
import java.awt.geom.Line2D;
import javax.swing.JOptionPane;

/**
  An applet that computes and draws the inter
section points
  of a circle and a line.
*/
```

```
12
   public class IntersectionApplet extends Applet
13 {
14   public IntersectionApplet()
15   {
16      String input
17         = JOptionPane.showInputDialog("x:");
18      x = Integer.parseInt(input);
19   }
20
21   public void paint(Graphics g)
22   {
23      Graphics2D g2 = (Graphics2D)g;
24
```

```
25      double r = 100; // the radius of the circle
26
27      // draw the circle
28
29      Ellipse2D.Double circle
30        = new Ellipse2D.Double(0, 0, 2 * RADIUS, 2 * RADIUS);
31      g2.draw(circle);
32
33      // draw the vertical line
34
35      Line2D.Double line
36        = new Line2D.Double(x, 0, x, 2 * RADIUS);
```

```
37      g2.draw(line);
38
39      // compute the intersection points
40
41      double a = RADIUS;
42      double b = RADIUS;
43
44      double root = Math.sqrt(RADIUS * RADIUS - (x - a) * (x - a));
45      double y1 = b + root;
46      double y2 = b - root;
47
48      // draw the intersection points
```

```
49
50        LabeledPoint p1 = new LabeledPoint(x, y1);
51        LabeledPoint p2 = new LabeledPoint(x, y2);
52
53        p1.draw(g2);
54        p2.draw(g2);
55    }
56
57    private static final double RADIUS = 100;
58    private double x;
59  }
```

Coordinate Transformations

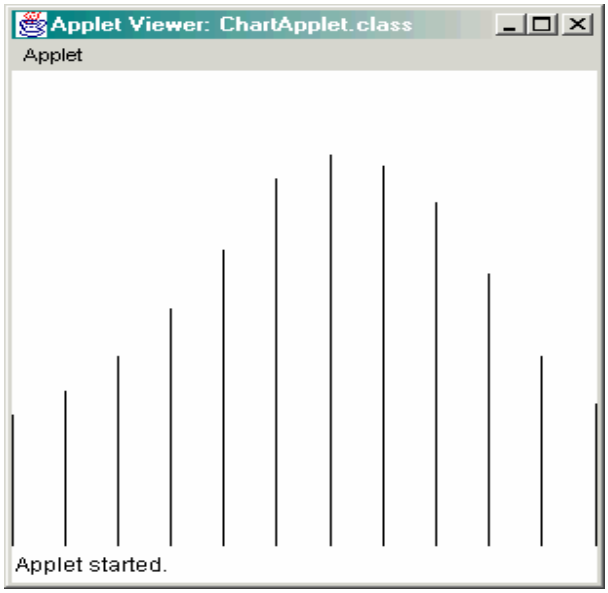
- Plot temperature data in Phoenix
- *x* ranges from 1 (January) to 12 (December)
- *y* ranges from 11 degrees (Celsius) to 33 degrees
- Transform user coordinates to pixel coordinates

$$x_{\text{pixel}} = (x_{\text{user}} - x_{\text{min}}) \cdot (\text{width} - 1) / (x_{\text{max}} - x_{\text{min}})$$

$$y_{\text{pixel}} = (y_{\text{user}} - y_{\text{min}}) \cdot (\text{height} - 1) / (y_{\text{max}} - y_{\text{min}})$$

- Encapsulate computation in convenience methods xpixel, ypixel (see code)
- Even better, use graphics context transforms (advanced topic)

Temperature Chart



File ChartApplet.java

```
import java.applet.Applet;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.geom.Line2D;

/**
This applet draws a chart of the average monthly
  temperatures in Phoenix, AZ.
*/
public class ChartApplet extends Applet
{
```

```
  public void paint(Graphics g)
  {
    Graphics2D g2 = (Graphics2D)g;

    month = 1;

    drawBar(g2, JAN_TEMP);
    drawBar(g2, FEB_TEMP);
    drawBar(g2, MAR_TEMP);
    drawBar(g2, APR_TEMP);
    drawBar(g2, MAY_TEMP);
    drawBar(g2, JUN_TEMP);
    drawBar(g2, JUL_TEMP);
    drawBar(g2, AUG_TEMP);
```

```
26      drawBar(g2, SEP_TEMP);
27      drawBar(g2, OCT_TEMP);
28      drawBar(g2, NOV_TEMP);
29      drawBar(g2, DEC_TEMP);
30   }
31
32   /**
33 Draws a bar for the current month and increments
34      the month.
35      @param g2 the graphics context
36 @param temperature the temperature for the
     month
37   */
```

```
38   public void drawBar(Graphics2D g2, int temperature)
39   {
40      Line2D.Double bar
41         = new Line2D.Double(xpixel(month), ypixel(0),
42            xpixel(month), ypixel(temperature));
43
44      g2.draw(bar);
45
46      month++;
47   }
48
```

```
/**
Converts from user coordinates to pixel coordinates
@param xuser an x-value in user coordinates
@return the corresponding value in pixel coordinates
*/
public double xpixel(double xuser)
{
   return (xuser - XMIN) * (getWidth() - 1) / (XMAX - XMIN);
}

```

```
/**
   Converts from user coordinates to pixel coordinates
   @param yuser a y-value in user coordinates
   @return the corresponding value in pixel coordinates
*/
public double ypixel(double yuser)
{
   return (yuser - YMAX) * (getHeight() - 1) / (YMIN - YMAX);
}

```

```
69  private static final int JAN_TEMP = 11;
70  private static final int FEB_TEMP = 13;
71  private static final int MAR_TEMP = 16;
72  private static final int APR_TEMP = 20;
73  private static final int MAY_TEMP = 25;
74  private static final int JUN_TEMP = 31;
75  private static final int JUL_TEMP = 33;
76  private static final int AUG_TEMP = 32;
77  private static final int SEP_TEMP = 29;
78  private static final int OCT_TEMP = 23;
79  private static final int NOV_TEMP = 16;
80  private static final int DEC_TEMP = 12;
```

```
81
82   private static final double XMIN = 1;
83   private static final double XMAX = 12;
84   private static final double YMIN = 0;
85   private static final double YMAX = 40;
86
87   private int month;
88 }
89
90
```