Lecture 34

- Covers
  - Life cycle of an applet
  - Fonts and font metrics

- Life cycle of an applet

An applet that misbehaves

- The applet on the next slide, which redefines method init( ), displays the message “Hello World!”
- But the message appears only briefly

An applet that misbehaves

```
import java.applet.*;
public class HelloApplet extends Applet
{
    public void init( )
    {
        setVisible(true);      // display applet on screen
        setSize(300, 300);
        Graphics g = getGraphics( );

        g.drawString("Hello World!", 100, 150);
     }
}
```

Applet's life cycle

- Why do applets behave the way they do?
- What happens behind the scenes?
- The key to these questions is to know what methods an applet has and when a particular method is called

Applet's life cycle

- Due to inheritance, an applet has, among others, the following methods

  init( )

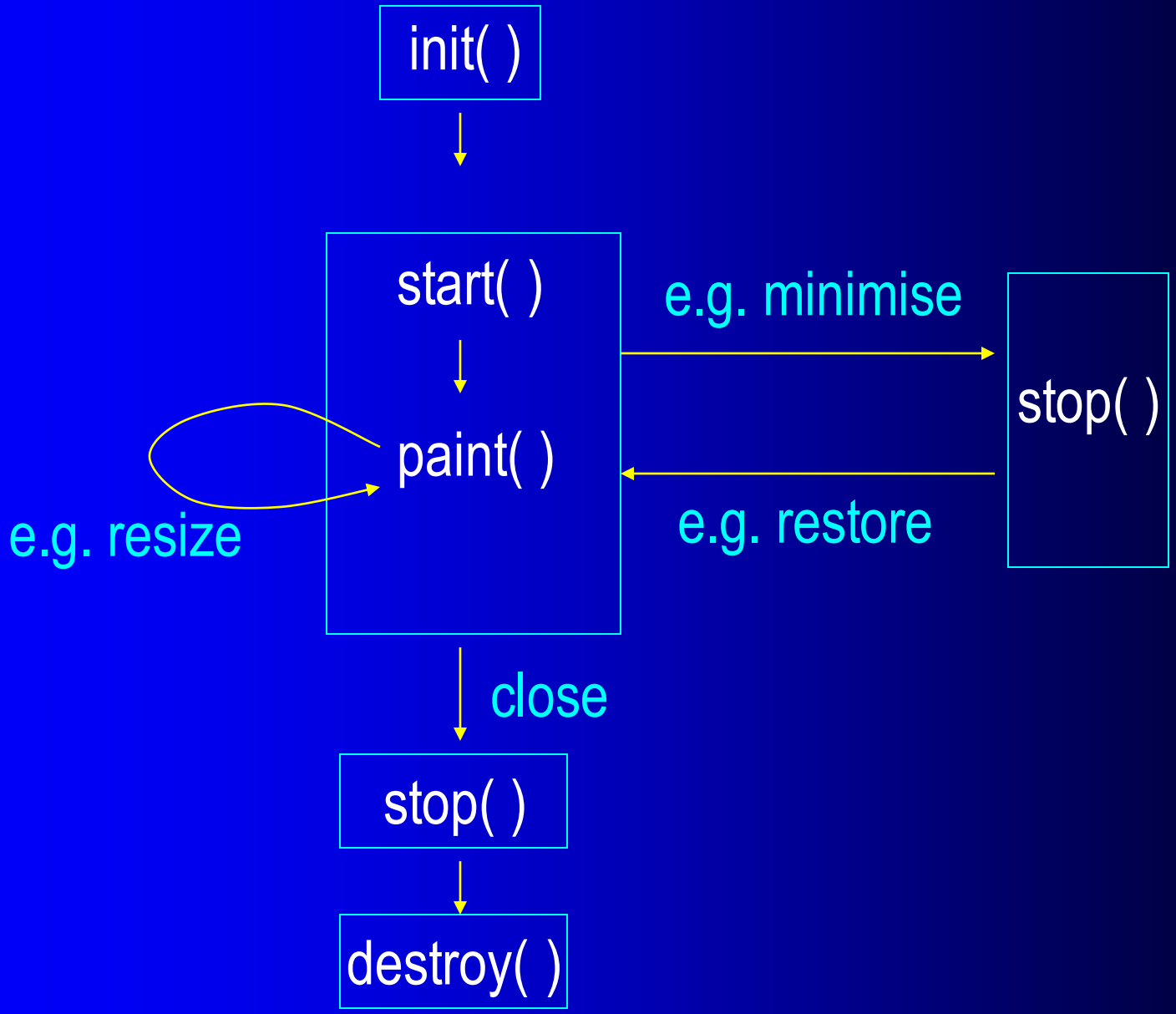
  start( )

  paint(Graphics)

  stop( )

  destroy( )

- The way these methods react to various common events is shown in the next diagram


init( )
start( )
paint( )
e.g. resize
e.g. minimise
stop( )
e.g. restore
close
stop( )
destroy( )

Compare two “Hello” applets

- Explain the behaviour of
  - The Hello applet that behaves properly (displays text in method paint)
  - The Hello applet that misbehaves (displays text in method init)

The init method

- Use the init method to set
  - The initial applet's size
  - The background color
  - The foreground color
- Note that
  - The background color should be set here, not in the paint method which may cause flickering
  - By default, the Graphics object takes the foreground color to be its drawing color

The init method

```
import java.applet.*;
import java.awt.*;
public class SampleApplet extends Applet
{
    public void init( )
    {
        setSize(300, 300);     // make your choices
        setBackground(Color.yellow);
        setForeground(Color.blue);
    }
}
```

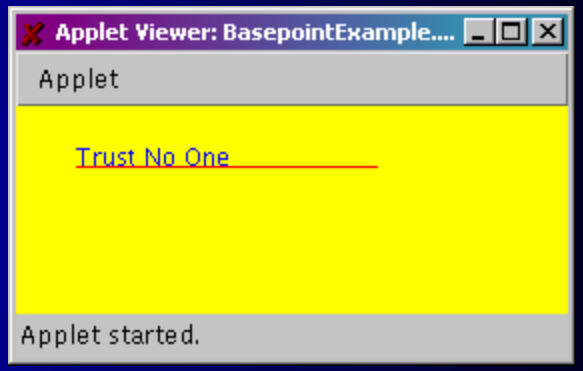
- Fonts and font metrics

Fonts

- Frequently we want to put some text into a graphical applet, for example to label a component
- To use the drawString method we have to specify the basepoint of the first character

Basepoint Example

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

public class BasepointExample extends Applet
{
    public void paint(Graphics g)
    {
        setBackground(Color.yellow);
        g.setColor(Color.red);
        g.drawLine(30,30,180,30);
        g.setColor(Color.blue);
        g.drawString("Trust No One", 30, 30);
    }
}
```



Fonts

- To output text, the characteristics of the font to use must be specified
- Three characteristics
  - Font face name
  - Style
  - Point size

Font face name

- The font face name is either a logical face name or a typeface name available on your computer
- Typeface names e.g.

  "Times New Roman"

  "Helvetica"

  "Courier New"

  "Lucida Handwriting"

Serifs

- Compare the shape of the following two fonts

"Times New Roman"

"Helvetica"

- The main difference is that Times New Roman has small cross segments at the end of the strokes
- These cross segments are called serifs
- Letters in the Helvetica font do not have serifs
- Serifs are believed to make text easier to read

Font shape

- Designing a typeface is a difficult artistic task
- In the USA, one cannot copyright the shape of a font, but can copyright the name of a font
- Resulting in, for example
  - Times - Arial
  - Times New Roman - Helvetica
  - Courier
  - Courier New
  - CourierHP

Logical font names

- Selecting fonts by typeface name is problematic if similar fonts may be called by different names on different computers
- How do you know which font will be available on the machine the applet is running?
- Java has five logical font names and the Java font mapper will search the local machine for the best matching font

Logical font names

Name	Sample	Description
Serif	Qwerty	A font with serifs
SansSerif	Qwerty	A font without serifs
Monospaced	Qwerty	Every character has the same width
Dialog	**Qwerty**	Suitable for labels
DialogInput	Qwerty	Suitable for input in a dialog box

Point size

- Letter size is specified in points
- A point is 1/72 of an inch
- Varies between fonts
- Examples
  - 8 points (small)
  - 12 points (medium)
  - 18 points (large)
  - 36 points (huge)

Point size

- Point size is the distance from the top of the ascender to the bottom of the descender

Creating fonts

- Font objects contain font formatting information

  Font(String fontType, int fontStyle, int fontSize)

  where fontType can be

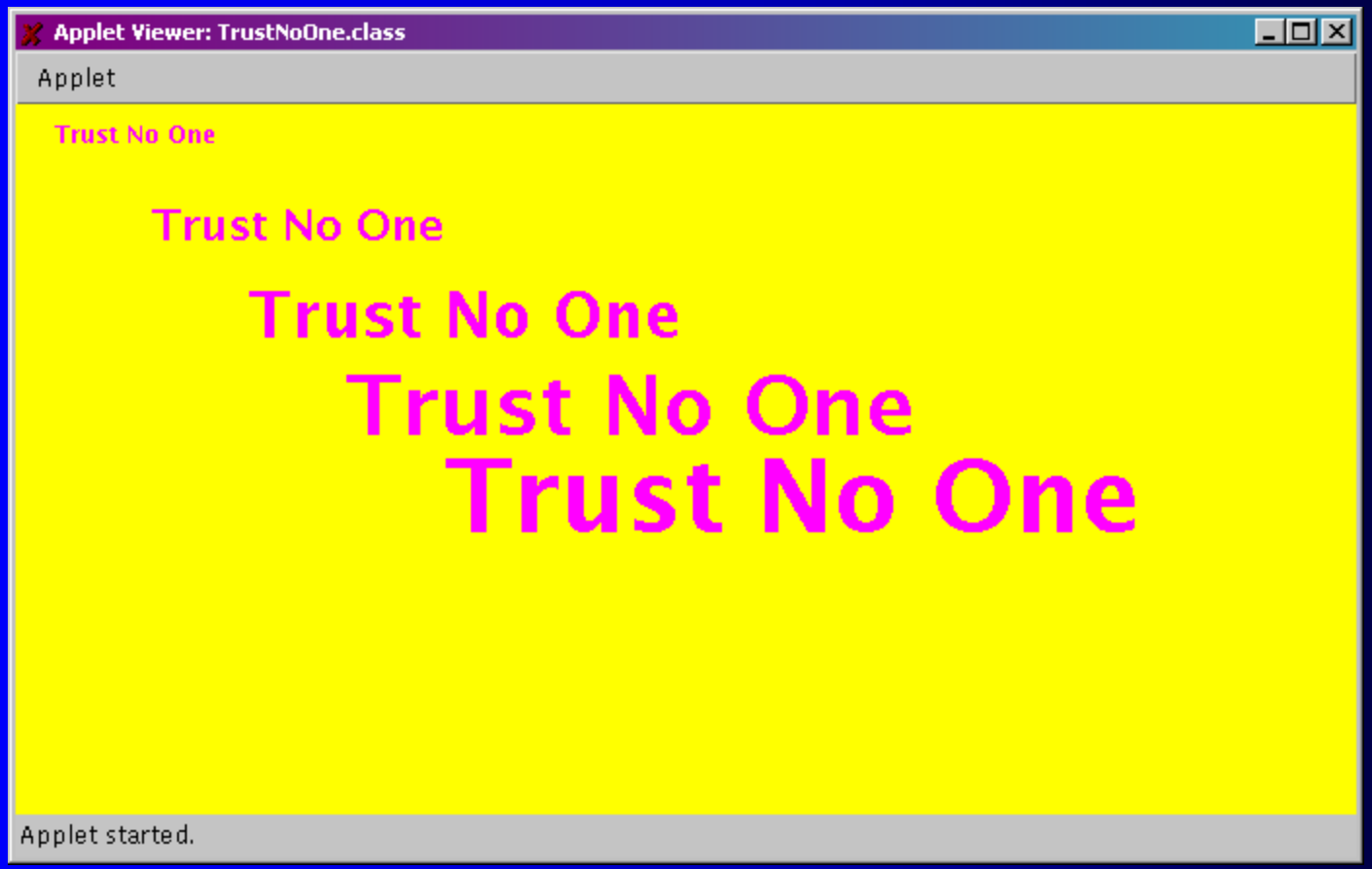
  "Serif" "SansSerif" "MonoSpaced"
  "Dialog" "DialogInput"

  and fontStyle can be

  Font.PLAIN Font.BOLD
  Font.ITALIC Font.BOLD + Font.ITALIC

Font example

- Write an applet to display the string “Trust No One” in increasingly larger text

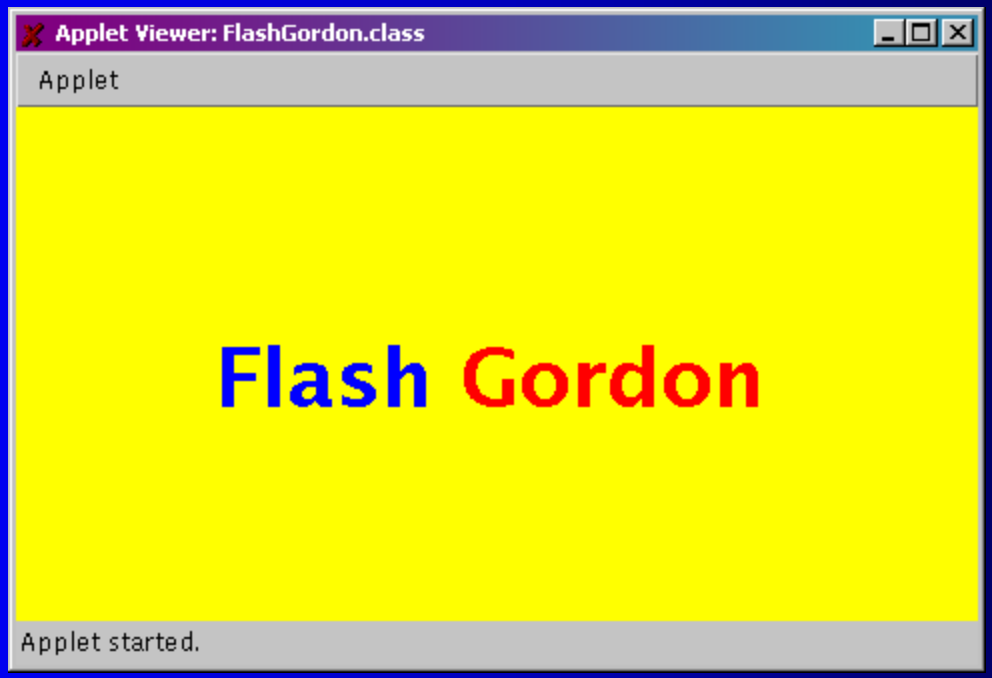


Font example

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

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

        for (int i = 0; i < 5; ++i)
        {
            Font f = new Font("SansSerif", Font.BOLD, 12 + (i * 10));
            g.setFont(f);
            g.drawString("Trust No One", 20 + (i * 50), 20 + (i * 50));
        }
    }
}
```

Font metrics – example 1

- Write an applet to display “Flash Gordon”, with the word “Flash” in one colour and “Gordon” in another colour



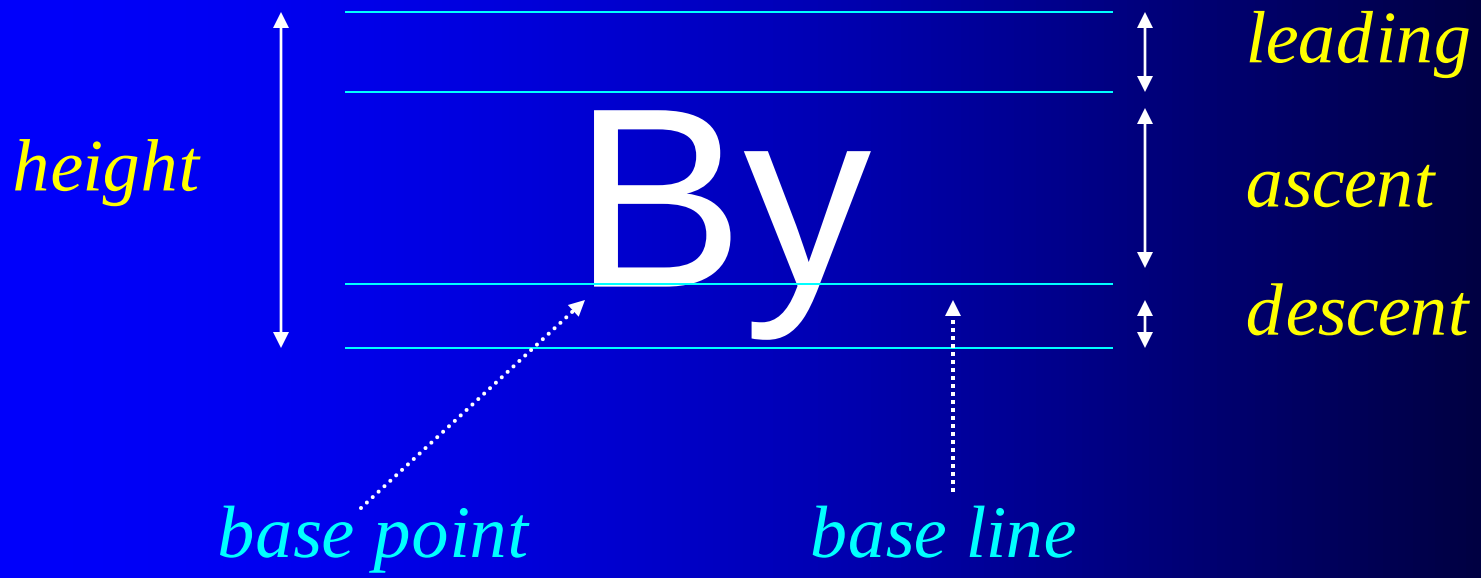
Font metrics – example 1

- This example requires us to position the text accurately
- To do that, we need to know about font metrics

Basics of font metrics

- In printing, a font is a set of characters of the same style and size
- Font metrics are various measurements about a font
- Basic font metrics are
  - Ascent
  - Descent
  - Leading (pronounced “ledding”)
  - Height

Basics of font metrics



Getting fonts metrics

- To get font metrics, we first get an instance of the FontMetrics class, and then get the required metrics from that instance
- To get a FontMetrics instance, use the method

  getFontMetrics( )

  of the Graphics class

FontMetrics

- Various useful metrics can be obtained by the following methods of the FontMetrics class

  int getAscent( )

  int getDescent( )

  int getLeading( )

  int getHeight( )

  int stringWidth(String str)

Example 1 revisited

- Display “Flash” in blue, “Gordon” in red
- Yellow background
- Use bold sans serif font of size 40
- Base point at (100, 150)

Example 1 revisited

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

public class FlashGordon extends Applet
{
    private String message1, message2;

    public void init( )
    {
        setSize(520, 300);
        setBackground(Color.yellow);
        setForeground(Color.blue);

        message1 = "Flash ";
        message2 = "Gordon";
    }
```

Example 1 revisited

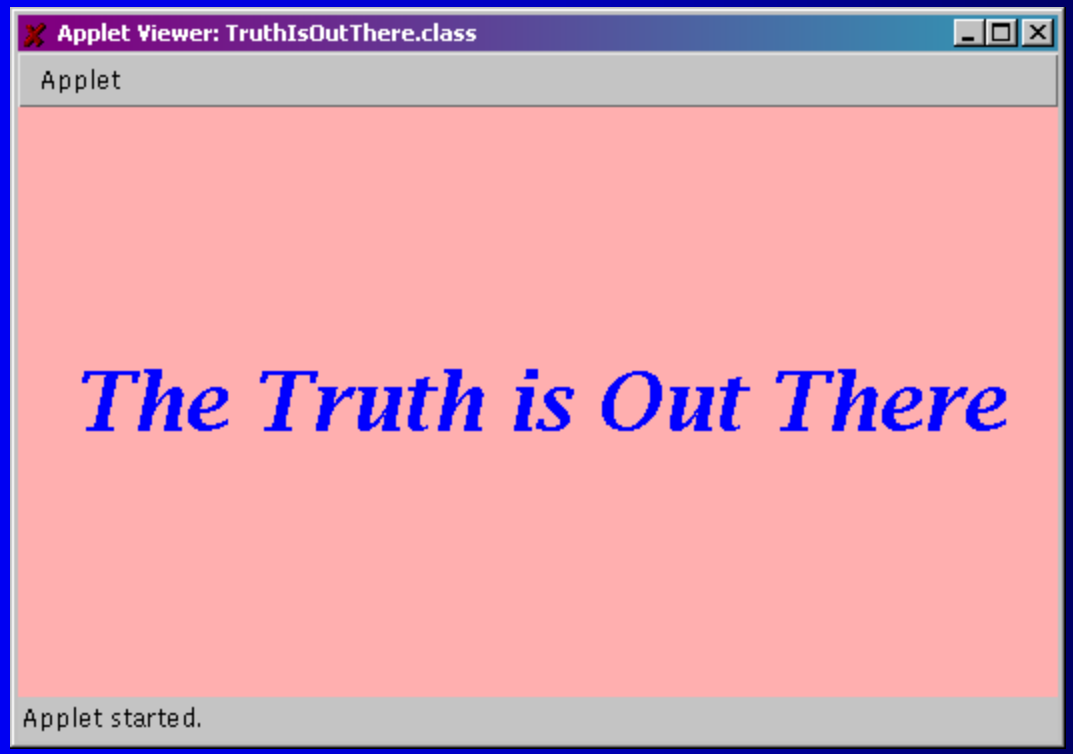
```
    public void paint(Graphics g)
    {
        g.setFont(new Font("SansSerif", Font.BOLD, 40 ));

        int xBase = 100;
        int yBase = 150;
        FontMetrics metrics = g.getFontMetrics( );
        int widthOfMessage1 = metrics.stringWidth(message1);

        g.setColor(Color.blue);
        g.drawString(message1, xBase, yBase);

        g.setColor(Color.red);
        g.drawString(message2, xBase + widthOfMessage1, yBase);
    }
}
```

FontMetrics

- getFontMetrics( )
  - Is a method of the Graphics class
  - Returns a FontMetrics object containing information about the currentFont
- stringWidth(String message)
  - Is a method of the FontMetrics object
  - Returns the length of the text message were it rendered with a font with the metrics described by the FontMetrics object

Font metrics - example 2

- Display the message "The Truth is Out There" (in a fixed font of your choice) so that the message is always in the middle of the applet, vertically and horizontally



Font metrics - example 2

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

public class TruthIsOutThere extends Applet
{
    private String message;

    public void init( )
    {
        setSize(520, 300);
        setBackground(Color.pink);
        setForeground(Color.blue);

        message = "The Truth is Out There";
    }
```

Font metrics - example 2

```
    public void paint(Graphics g)
    {
        g.setFont(new Font("Serif", Font.BOLD + Font.ITALIC, 40 ));

        FontMetrics metrics = g.getFontMetrics( );
        int widthOfMessage = metrics.stringWidth(message);
        int heightOfMessage = metrics.getAscent( ) + metrics.getDescent( );

        int xCoord = (getWidth( ) - widthOfMessage) / 2;
        int yCoord = (getHeight( ) - heightOfMessage) / 2 + metrics.getAscent( );

        g.drawString(message, xCoord, yCoord);
    }
}
```

Next lecture

- Arrays and applets