Welcome

Java Programming I
CIS 325

- Week 10 -

Java Applets

Multi-Threading

- - - -

Christopher K. Burns

Agenda

Tonight's agenda

- Java Applets
- Multi-Threading

Schedule

Week		Content	
1	4/6	Chapter 1 Intro to Computers, the Internet and the Web Chapter 2 Intro to Java Applications Chapter 3 Java Classes and Objects: Part 1 *Homework 1 Assigned*	
2	4/13	Chapter 4 Control Structures: Part 1 Chapter 5 Control Structures: Part 2	
3	4/20	Chapter 6 Methods Chapter 7 Arrays *Homework 2 Assigned*	HW 1 DUE
4	4/27	Chapter 8 Java Classes and Objects: Part 2 Chapter 1-8 Review	
5	5/4	***MID-TERM EXAMINATION***	HW 2 DUE
6	5/11	Chapter 9 Object-Oriented Programming: Inheritance Chapter 10 Object-Oriented Programming: Polymorphism *Homework 3 Assigned*	PROJECT IDEA DUE
7	5/18	**No Class Tonight**	
8	5/25	Chapter 11 GUI Components: Part 1 Chapter 12 Graphics and Java2D *Homework 4 Assigned*	HW 3 DUE
9	6/1	Chapter 13 Exception Handling Chapter 29 Strings, Characters and RegEx	
10	6/8	Chapter 20: Java Applets Chapter 23 Multithreading	HW 4 DUE
11	6/15	Files, JDBC, Networking, Servlets, and JSP *Class lab time for review and assistance with final project* ***FINAL PROJECT DUE***	PROJECT DUE

Home Work

Due tonight:

Read Chapters 20 & 23

Create a Java GUI that contains at least one button, one label, and one text field.

Use the GUI to create or manipulate an instance or instances of your class from the assignment due last week.

Due next week:

Project

As Final Grades must be calculated after the next class, no assignments can be accepted after the 15th of June!

Syllabus

EVALUATION METHODS:

Homework	30%
Mid Term	30%
Final Project	30%
Class Participation	10%

GRADING SCALE:

90-100 %	A
80-89 %	B
70-79 %	C
60-69 %	D
Below 60 %	F

Java Applets

A Java Applet is essentially a JFrame on a web page.

The applet has to follow security parameters of the web browser, that is to say it cannot read/write from the user's computer. But it can communicate with the server on which it lives.

Java Applets are not as popular as they once were since animations are more easily done in flash, and pages that need to interact with a server a better done in

Java Applets are still a powerful way to create small web based applications.

Java Applets

There are two approaches that can be used

- Create an applet using controls, like creating a JFrame.
- Override the paint method of the applet and draw your text and graphics.

Neither is mutually exclusive.

We will look at both approaches.

Java Applets – `init()`

When creating a Java Application, the `main()` method is called first.

Java Applets do not have a `main()` method, but instead have an `init()` method that servers a similar purpose.

`init()` is not a static method like `main()` because an instance of the JApplet class is created by the host or web browser.

Java Applets – `paint()`

When we looked at Java graphics, we did most of our work by overriding the `paint()` method of a JPanel.

Java Applets have a `paint()` method that we can override in the exact same way.

Java Applets – built in methods

The life of a JApplet:

`init()` – called when applet first created
`paint()` – called when the host needs to redraw its container
`repaint()` – similar to paint, but not usually overridden. Used to force a paint.
`start()` – called when applet is first made visible
`stop()` – called when applet is removed from the page
`destroy()` – called shortly after stop

A Simple Applet – Applet Code

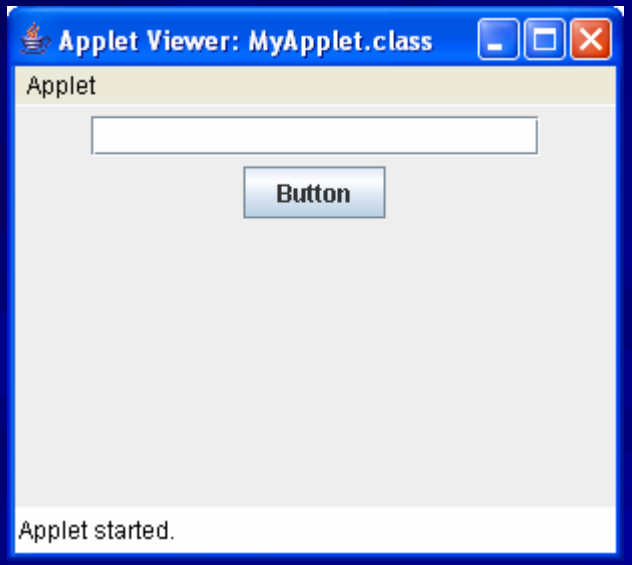
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class MyApplet extends JApplet
{
  JButton button;
  JTextField text;
  public void init()
  {
    button = new JButton( "Button" );
    button.addActionListener( new ActionListener()
      { public void actionPerformed( ActionEvent e )
        { text.setText( "Button pressed" ); } } );
    text = new JTextField( 20 );
    getContentPane().setLayout( new FlowLayout() );
    getContentPane().add( text );
    getContentPane().add( button );
  }
}
```

A Simple Applet – HTML

```
<html>
  <applet
    code = "MyApplet.class"
    codebase = "classes/"
    width = "300" height = "200">
  </applet>
</html>
```

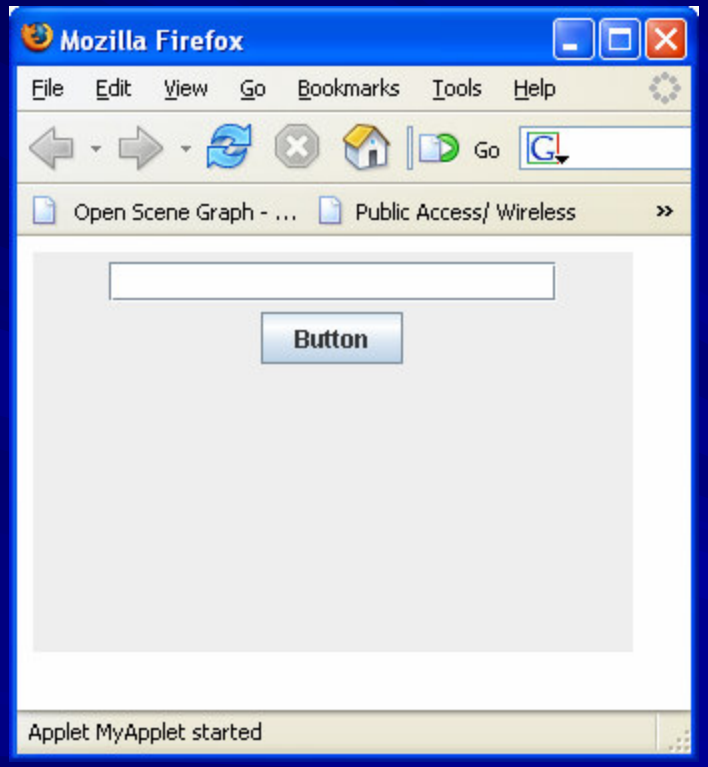
Running the Applet

AppletViewer:



Running the Applet

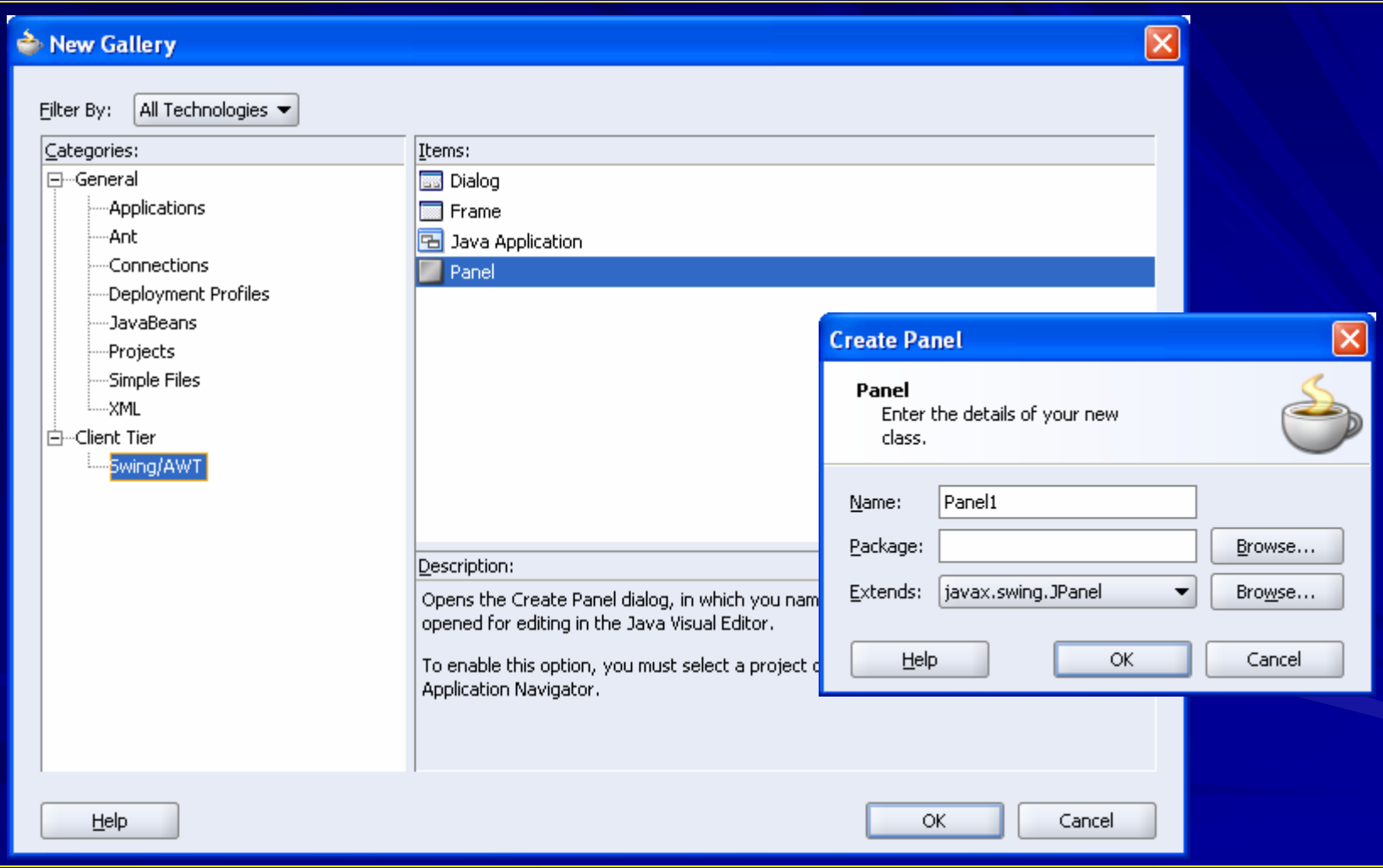
Web Browser:



Java Applets and JDeveloper

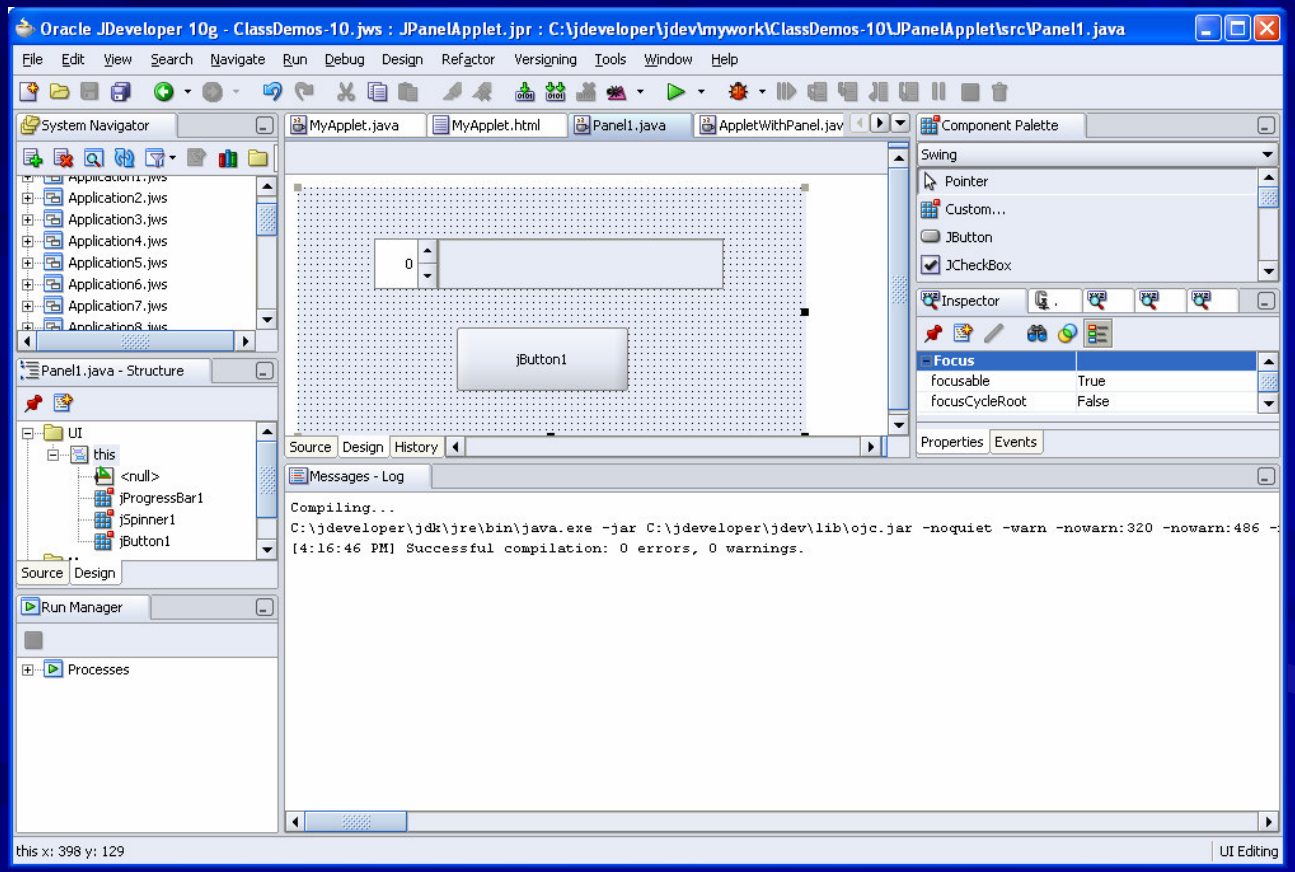
1. Create a JPanel in JDeveloper
2. Create an applet to host the JFrame
3. Create the html
4. Run it…

Java Applets and JDeveloper



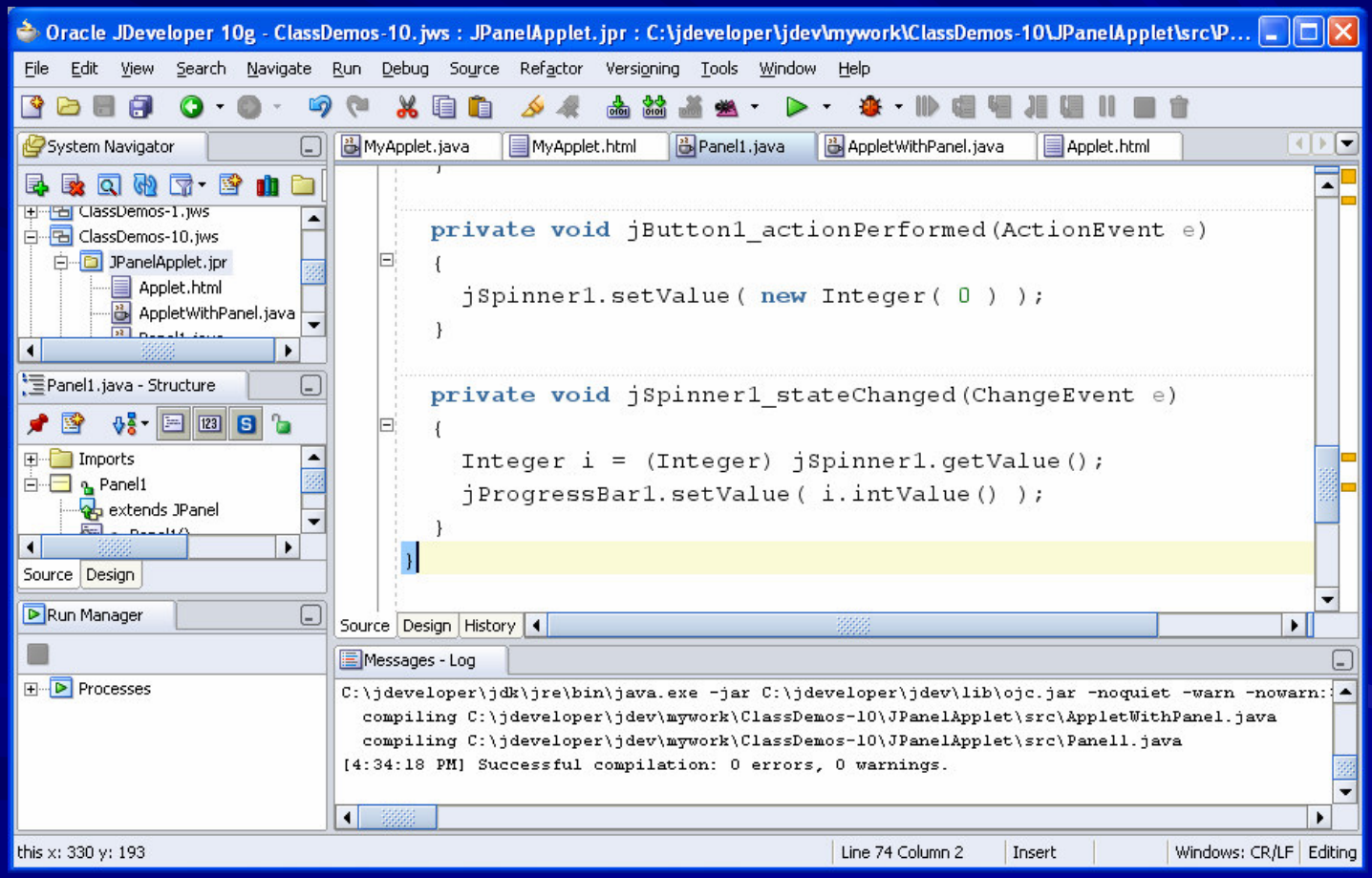
Java Applets and JDeveloper

Add some controls and code to the panel



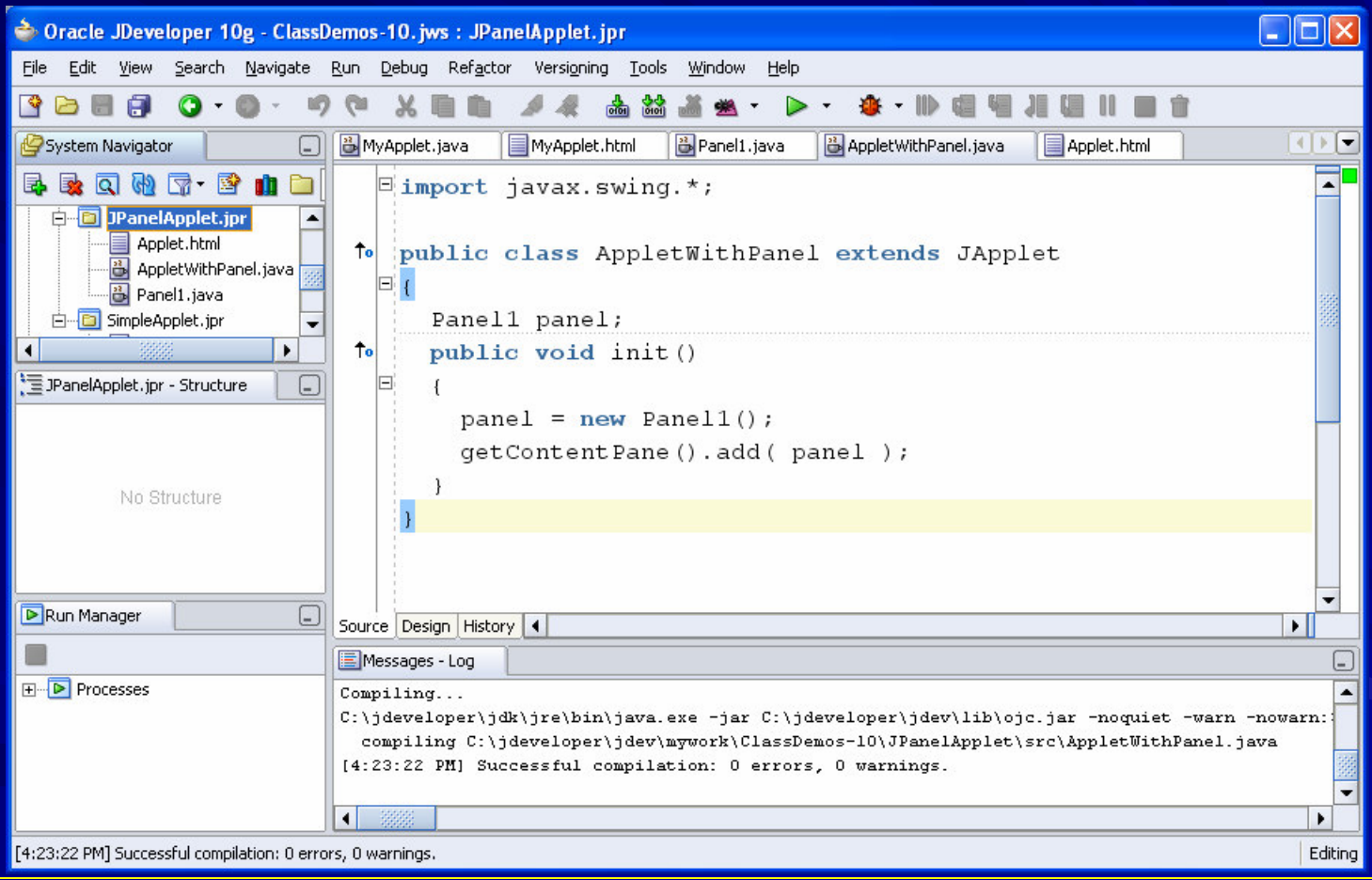
Java Applets and JDeveloper

Put some code behind the controls



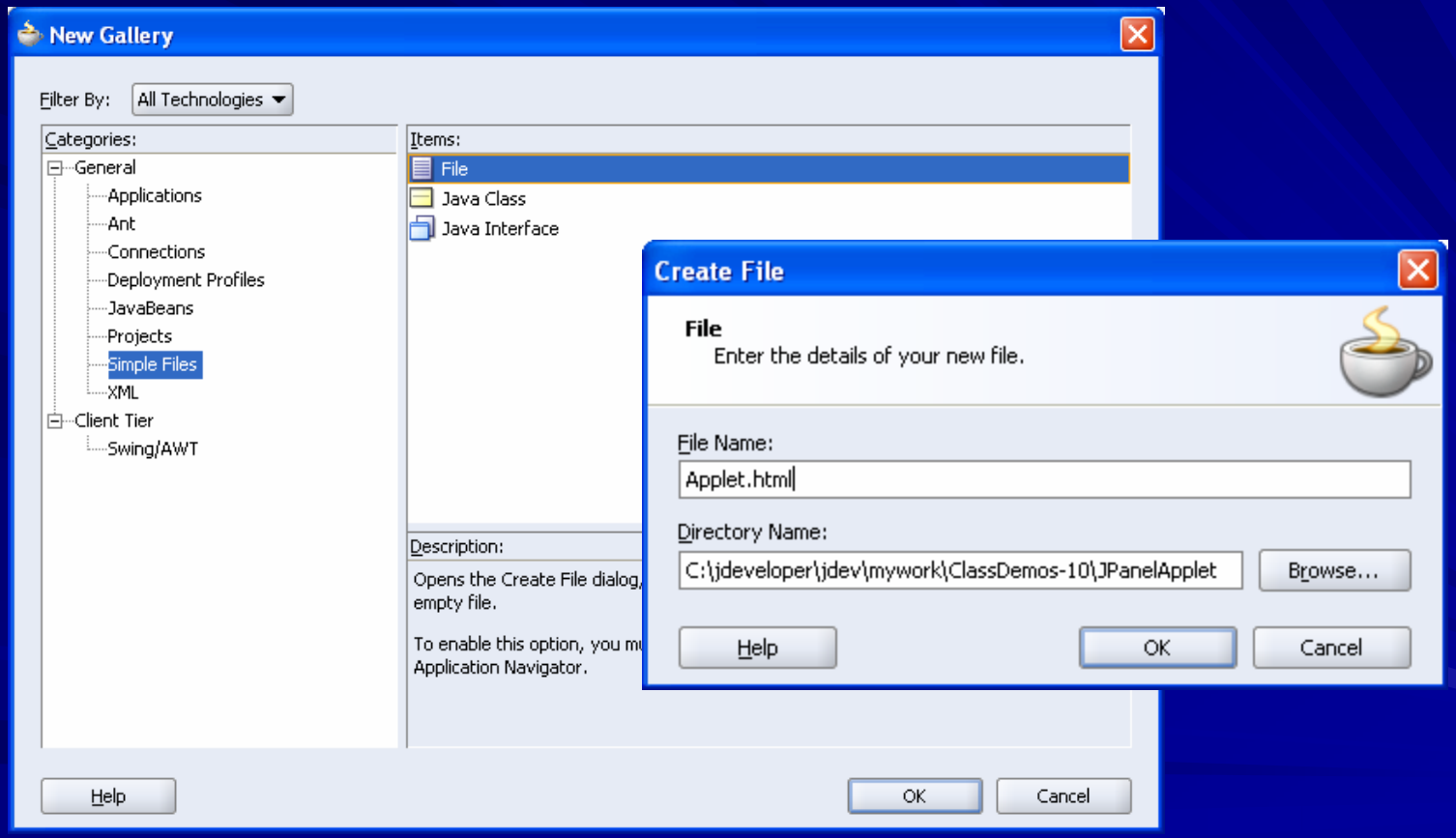
Java Applets and JDeveloper

Create a JApplet (just like a regular java class)



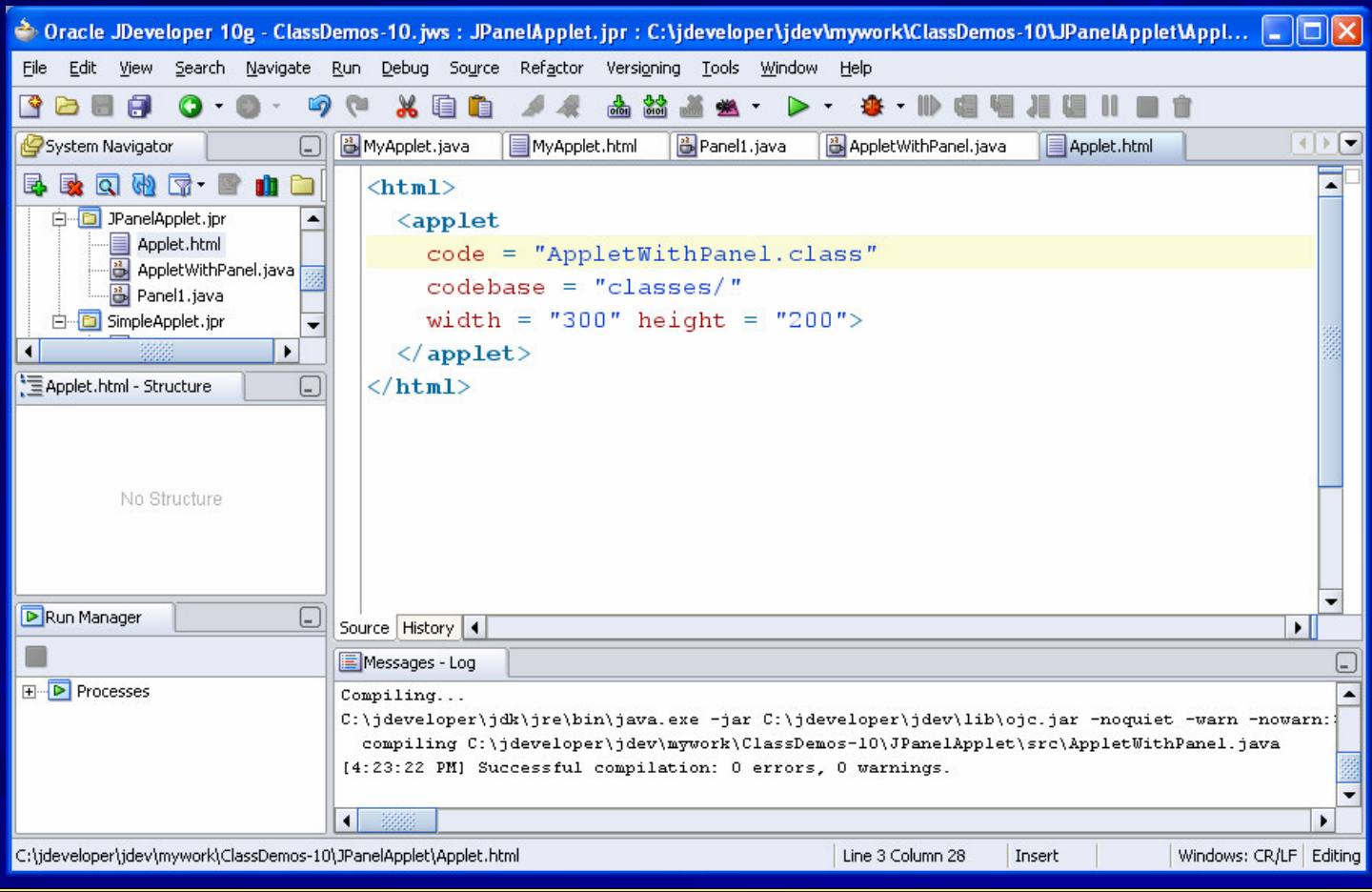
Java Applets and JDeveloper

Add the html as a regular file



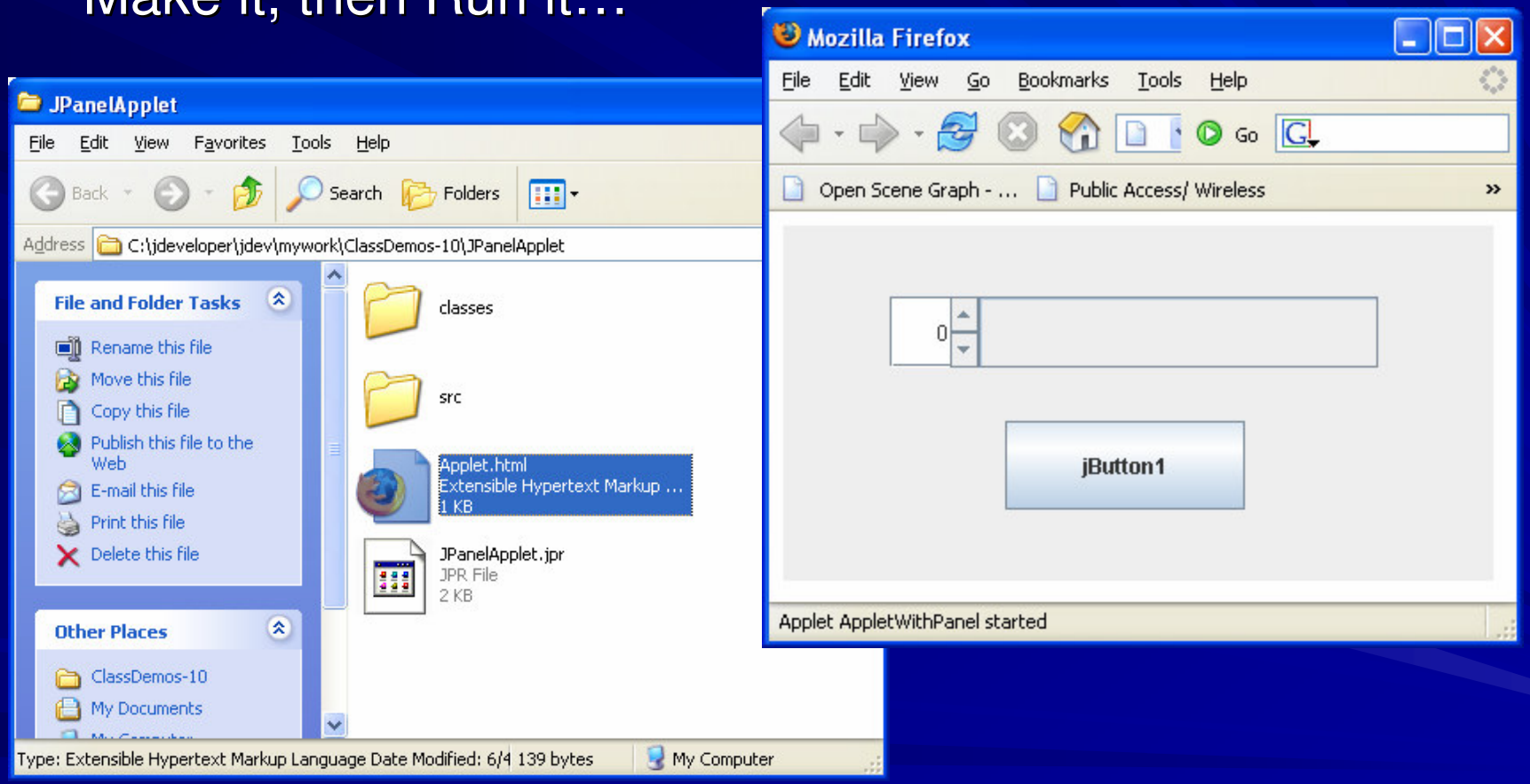
Java Applets and JDeveloper

Add the html code



Java Applets and JDeveloper

Make it, then Run it…



Java Applets and Paint

Let's look at an example using that override's the paint method of the Java Applet and draws text and inserts an image.

Java Applets and Paint

```
import java.io.*;
import java.awt.*;
import javax.swing.*;

public class PaintApplet extends JApplet
{
  ImageIcon image;

  public void init()
  {
    image = readImage( "face1.JPG" );
  }

  public void paint(Graphics g)
  {
    super.paint(g);
    g.drawString("This applets shows the paint method!", 10, 25);
    image.paintIcon( this, g, 30, 30 );
  }
```

Java Applets and Paint

```
public ImageIcon readImage( String imageName )
{
  int count = 0;
  BufferedInputStream imgStream =
    new BufferedInputStream(this.getClass().getResourceAsStream(imageName));
  if (imgStream != null)
  {
    byte buf[] = new byte[30000];
    try
    {
      count = imgStream.read(buf);
      imgStream.close();
    }
    catch (java.io.IOException ioe)
    {
      System.err.println("Couldn't read stream from file: " + imageName);
      return null;
    }
    return new ImageIcon(Toolkit.getDefaultToolkit().createImage(buf));
  }
  return null;
}
}
```

Threads

We'll return to Applets and show how to create a simple animation, but first we need to understand threading in Java

Threads and Processes

What is a process?

Your program is a process.

Other programs can run along side your program, these are other processes.

Processes run simultaneously, but exist separately

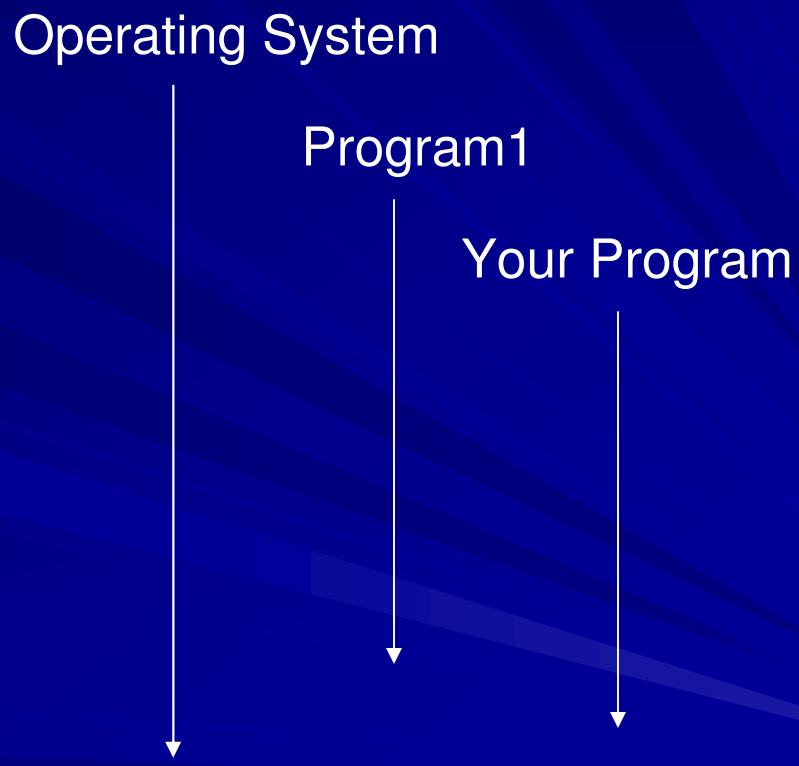
Threads and Processes

What is a process?

Your program is a process.

Other programs can run along side your program, these are other processes.

Processes run simultaneously, but exist separately



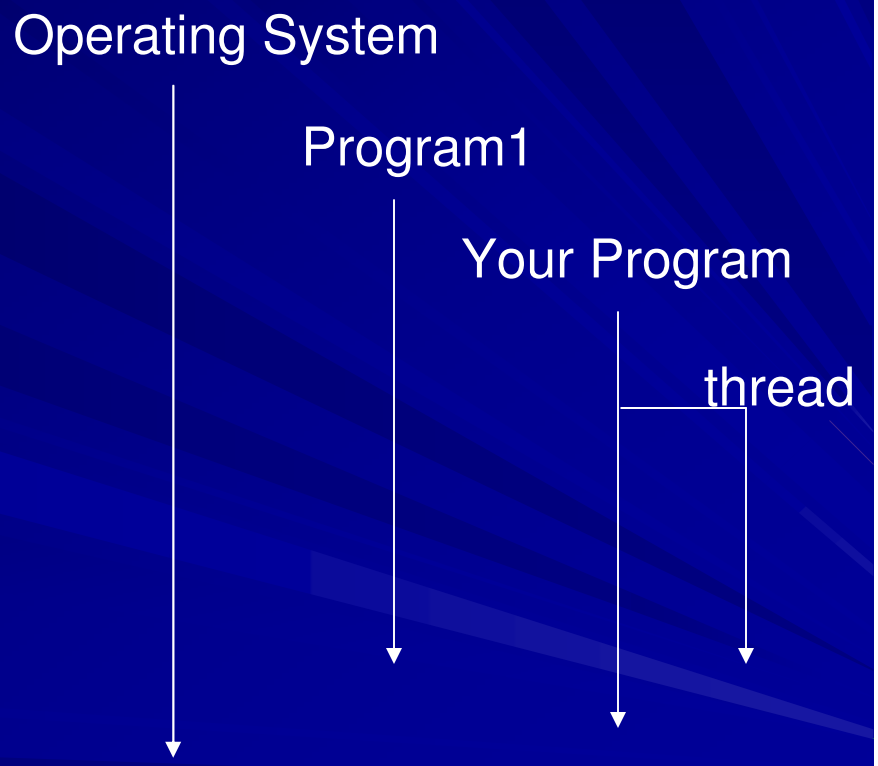
Threads

What is a thread?

Your program can spawn threads that, like other processes run simultaneous to your program

Unlike processes, they exist within you program

This allows you to perform multiple tasks at the same time



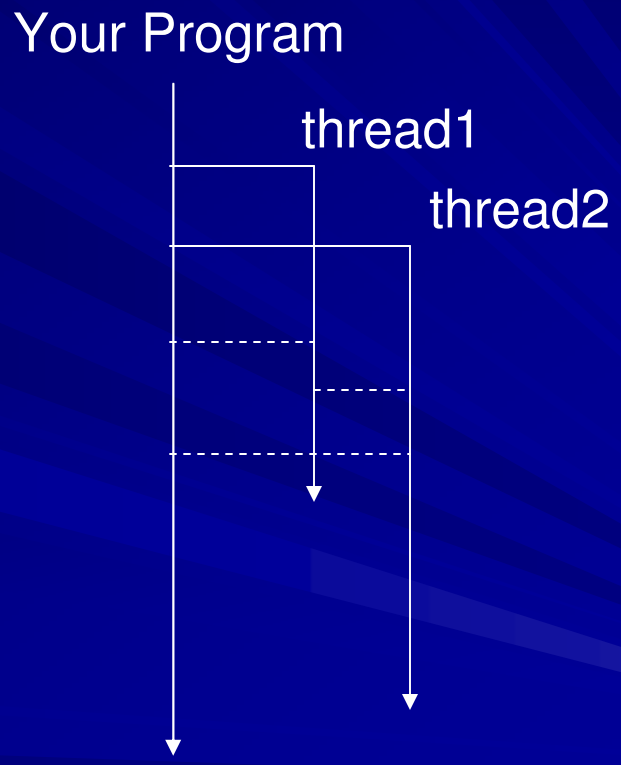
Threads

What is a thread?

A thread shares the same variables as the process that spawned it

A thread can communicate with other threads

Special communication mechanisms like monitors exist so threads can safely use the same resources



Threads

What are some multithreaded applications that you use?

Word Processor

- A modern word processor has a thread in the background that checks spell and grammer

Web Browser

- A web browser allows you to view a page and interact with it while a thread continues to load images and data.

Threads

Creating threads of your own:

Any Java class may implement the runnable interface and have a run method that can be spawned as a thread.

We can do this explicitly, but an anonymous inner class definition makes this easier.

Threads using class implementing Runnable

```
public class BasicThreading implements Runnable
{
  public void run() // run is the thread method for Runnable
  {
    for( int i = 1; i <= 10; i++ )
    {
      System.out.println( "thread - " + i );
      try {Thread.sleep(100);} catch (InterruptedException e){ }
    }
  }

  public static void main(String[] args)
  {
    BasicThreading me = new BasicThreading();
    Thread myThread = new Thread( me ); // create the thread
    myThread.start(); // start the thread (calls the run method)
    for( int i = 1; i <= 10; i++ )
    {
      System.out.println( "main - " + i );
      try {Thread.sleep(100);} catch (InterruptedException e){ }
    }
  }
}
```

Threads using anonymous inner class

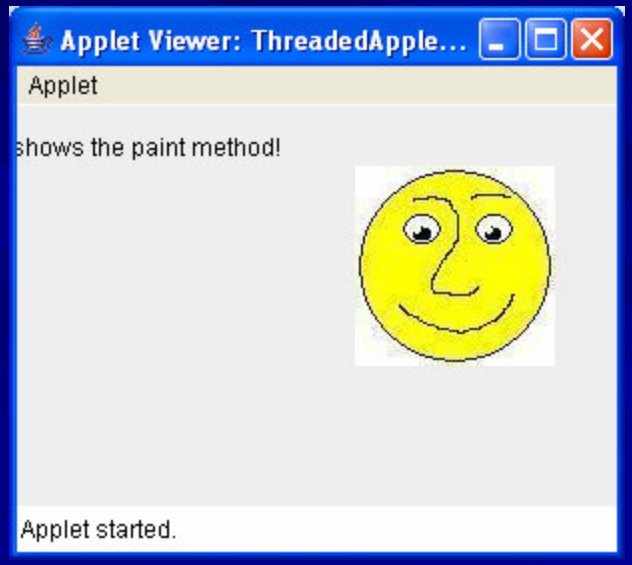
```
Thread myThread = new Thread(
      new Runnable()
      {
        public void run()
        {
          me.printNumbers( "thread", 100 );
        }
      }
    );
```

Threads using anonymous inner class

```
public class BasicThreading2
{
  static BasicThreading2 me;
  public void printNumbers( String name, long sleepInterval )
  {
    for( int i = 1; i <= 10; i++ )
    {
      System.out.println( name + " - " + i );
      try {Thread.sleep(sleepInterval);}  catch (InterruptedException e) {return;}
    }
  }

  public static void main(String[] args)
  {
    me = new BasicThreading2();
    Thread myThread = new Thread( new Runnable() {
      public void run() {
        me.printNumbers( "thread", 100 );
      } } ); // create the thread
    myThread.start(); // start the thread (calls the run method)
    me.printNumbers( "main", 50 );
    myThread.interrupt(); // interrupt the thread
  }
}
```

Java Applets, Threads, and simple animation

Let's create a simple animation on a JApplet using the paint method.



```
import java.io.*;
import java.awt.*;
import javax.swing.*;

public class ThreadedApplet extends JApplet
{
	ImageIcon[] images;
	int textX, textY, faceX, faceY;
	int activeFace;
	boolean animate;
	long frameInterval;
	Thread animationThread;

	public void animation()
	{
		while (animate)
		{
			try { Thread.sleep(frameInterval); }
			catch (InterruptedException e)
			  { System.out.println("animation thread sleep interrupted!"); }
			activeFace = (activeFace + 1) % 10;
			textX += 3;
			if (textX > 300)
				textX = -200;
			faceX++;
			if (faceX > 300)
				faceX = -100;
			this.repaint();
		}
	}
```

```
public void init()
{
      setBackground(Color.white);

      frameInterval = 100;
      textX = 10;
      textY = 25;
      faceX = 30;
      faceY = 30;
      activeFace = 0;

      images = new ImageIcon[10];
      for (int i = 0; i < images.length; i++)
      {
            images[i] = readImage("face" + (i + 1) + ".JPG");
      }

      animate = true;
      Thread animationThread = new Thread(
        new Runnable()
      {
            public void run()
            {
                  animation();
            }
      }
        );
      animationThread.start();
}
```

```
public void stop()
{
	super.stop();
	animate = false;
}

public void paint(Graphics g)
{
	super.paint(g);
	g.drawString("This applets shows the paint method!", textX, textY);
	ImageIcon ii = (ImageIcon)images[activeFace];
	ii.paintIcon(this, g, faceX, faceY);
}
```

```
    public ImageIcon readImage(String imageName)
    {
        int count = 0;
        BufferedInputStream imgStream =
          new BufferedInputStream(this.getClass().getResourceAsStream(imageName));
        if (imgStream != null)
        {
            byte buf[] = new byte[30000];
            try
            {
                count = imgStream.read(buf);
                imgStream.close();
            }
            catch (java.io.IOException ioe)
            {
                System.err.println("Couldn't read stream from file: " + imageName);
                return null;
            }
            return new ImageIcon(Toolkit.getDefaultToolkit().createImage(buf));
        }
        return null;
    }
}
```

More on Applets and Threads

- Sun Java Tutorial on How to Make Applets – includes good thread animation demo

  http://java.sun.com/docs/books/tutorial/uiswing/components/applet.html