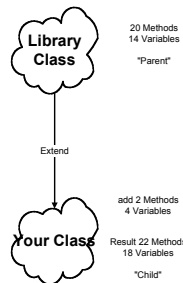


Introducing, the JFrame

Gives us a work area beside
System.out.println

Extension, Hierarchy, and Inheritance



Extension, Hierarchy, and Inheritance

- Extends – in the class statement, tells which larger class serves as the “parent”.
- Inheritance – the child class inherits methods and variables from the parent class
- Hierarchy - parent to child relationship

Extend keyword

```
public class myClass extends someOtherClass
{
  // I can use all the methods and
  // variables from someOtherClass in myClass.
}
```

Other terms

- Parent / Child
- Base class / Sub class
- Super Class / Sub class
- Any class (not just an imported library class) can be a parent or base class.

- Write classes – then write new inherited classes for targeted capabilities.

Java common inherited classes

- `public class myClass extends JFrame`
- myClass gets a whole bunch of methods for displaying windows on-screen
- `public class myClass extends Applet`
- applet and browser graphics
- `public class myClass extends Object`
the parent of ALL classes, usually omitted

Currently...

- If you don't specify an "extend" source, a class extends the Java base class "Object"
- Only mildly capable
- Better to extend a useful class

```
import javax.swing.*;
public class MainClass extends JFrame
{

    // stuff here

    public static void main (String [ ] args)
    {
        MainClass application = new MainClass( );
    }
}
```

What is this?

- Main method instantiates the class that it's in? Heavens why?
- Because main is not considered a member of the class that contains it.
- The Main class serves a place-holding function for the computer, by keeping main in a usable place.
- Instantiating the Main class causes the constructor (if there is one) to be run.
- What other reasons?

JFrame

- Puts graphics-capable windows on screen.
- JOptionPane uses JFrame
- Colors
- Fonts
- Drawings
- A frame for intuitive GUI

helper libraries

```
import javax.swing.* ; // for JFrame  
  
import java.awt.Graphics ; // for painting and  
animation  
  
import java.awt.event.* ; // for event handling
```

some JFrame methods

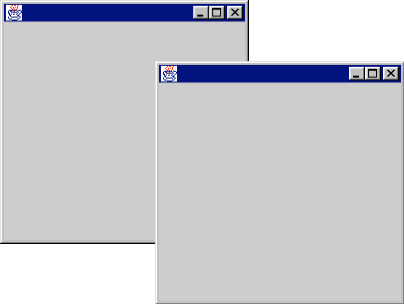
- setLocation(100,200);
- setSize(250,250);
- setVisible(true);
- getContentPane();

```
public class MainClass extends JFrame
{
public MainClass ( )
{
    setLocation(100,200);
    setSize(250,250);
    setVisible(true);
} // end MainClass constructor
public static void main (String[] args)
{
    MainClass application = new MainClass( );
} // end main method
} // end class
```

```
public class MainClass extends JFrame
{
public MainClass ( int x, int y )
{
    setLocation( x, y );
    setSize(250,250);
    setVisible(true);
} // end MainClass constructor
public static void main (String[] args)
{
    MainClass application = new MainClass(100, 200);
} // end main method
} // end class
```

```
public class MainClass extends JFrame
{
public MainClass ( int x, int y )
{
    setLocation( x, y );
    setSize(250,250);
    setVisible(true);
} // end MainClass constructor
public static void main (String[] args)
{
    MainClass appl1 = new MainClass(100, 200);
    MainClass appl2 = new MainClass(200, 300);
} // end main method
} // end class
```

Demo



Objects As Containers of Information

Objects don't always DO things

- Sometimes they just HOLD things (information)
- The Color class can hold color information
- The Point class can hold cartesian coordinate information
- The Container class, well...

Color class

- Like the String class, and Integer class, does not use the `new` keyword.

- holds a color value (nothing more)

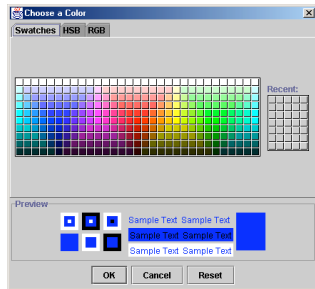
e.g `Color boxColor = new Color();`
`boxColor = Color.blue;`

or

`Color boxColor = Color.blue`

then `boxColor` can be used to set System properties in Classes/Objects that need color (more later).

JColorChooser – returns a Color object to the caller



Returns an Object?

- JColorChooser fills in all of the information in a blank object of the Color class, and copies it to the Color object in the calling statement:

```
boxColor = JColorChooser.showDialog(  
    null, Greeting, default color );
```

the Point class

- holds an x,y value (nothing more)

```
Point screenLocation = new Point( );
```

```
screenLocation.x = 100;  
screenLocation.y = 200;
```

the `this` qualifier

- Means “*this object*, the one that we’re in”.
- Used when a call to a method, needs to specify which object the method should act upon.
- `this` always refers to an object of some type.

the `super` qualifier

- refers to the parent class
- the command `super` calls the parent’s constructor
- `super.MethodName` can call any method in the parent.

graphics on “your computer”

- Since there are many computers (types, instances)....
- and since the programs that you write should run anywhere....
- then the programs that you write, should retrieve and use information native to the computer that it is running on.
- This is called “context awareness”

the Container class

- Holds a whole window “context”: frames, scroll bars, buttons, menus, pop-ups, and operator actions: mouse clicks, dragging & dropping.
- Operator actions are called *events*.
- JFrame objects must be placed in a container, to be of any use.
- ContentPane – is a special container that contains the current state of the computer.

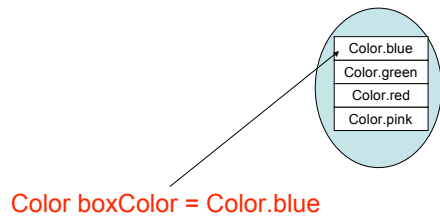
Creating a ContentPane container

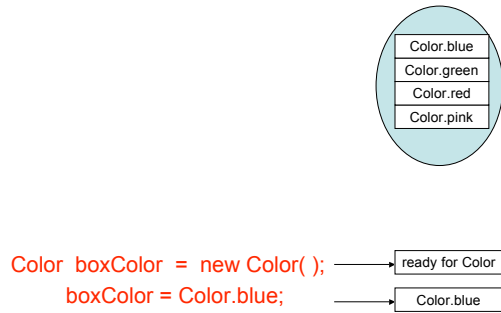
```
Container frameContainer = new Container();  
frameContainer = getContentPane( );
```

frameContainer is now an “event environment”, which is an object that *holds* the current details of all screen, user, keyboard, mouse events and conditions, *and further... is a place where we can build a GUI, compatible with the computer’s configuration (called a “layered glass pane”).*

a clarification

e.g `Color boxColor = new Color();`
`boxColor = Color.blue;`
or
`Color boxColor = Color.blue`



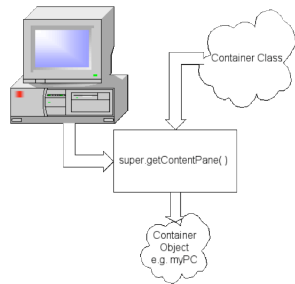


what does this do?

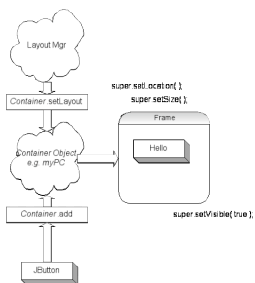
```
FlowLayout layout = new FlowLayout( )  
Container frameContainer = new Container();  
frameContainer = getContentPane( );  
frameContainer.setLayout( layout );
```

JFrame Container

create a Container to hold local computer
"contents" e.g. a ContentPane



the Container controls everything



more `JFrame` and `Container` methods

```
setSize( w, h );  
setLocation( x, y );  
setDefaultCloseOperation( EXIT_ON_CLOSE );  
  
Container myPC = new Container();  
myPC = getContentPane();  
FlowLayout layout = new FlowLayout();  
myPC.setLayout( layout );  
  
JButton helloButton = new JButton( "Hello" );  
myPC.add( helloButton );  
  
setVisible( true );
```
