CSE115 / CSE503
Introduction to Computer Science I

Dr. Carl Alphonce

343 Davis Hall

alphonce@buffalo.edu

Office hours:

Tuesday 10:00 AM – 12:00 PM*

Wednesday 4:00 PM – 5:00 PM

Friday 11:00 AM – 12:00 PM

OR request appointment via e-mail

**Tuesday adjustments: 11:00 AM – 1:00 PM on 10/11, 11/1 and 12/6*

ROADMAP

Last time

- Interfaces
- Realization relationship

Today

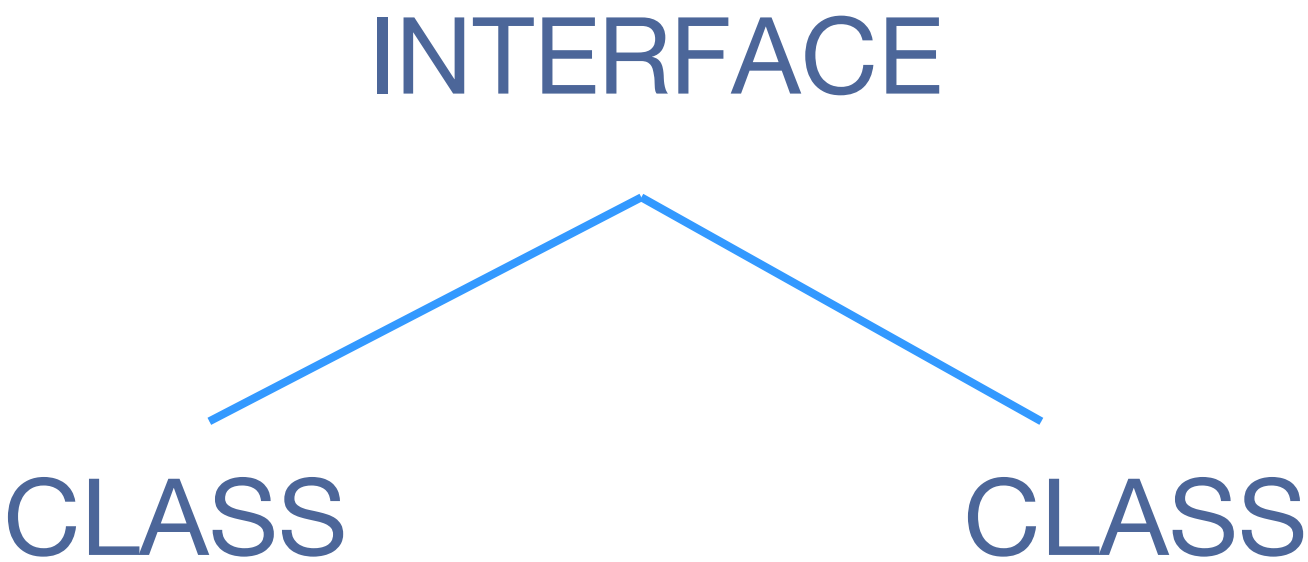
- Graphics
- Event handling

Coming up

- Primitives
- Control structures

REVIEW

POLYMORPHISM



Concrete example

```
public class EventHandler implements ActionListener {

      @Override
      public void actionPerformed(ActionEvent e) {
            System.out.println("Button clicked");
      }

}
```

Types

When you define a class, you are defining a type.

When you define an interface, you are also defining a type.

A class which implements an interface is a SUBTYPE of the interface type.

an instance of the class belongs to both types

Assignment

If a variable is declared to be of an interface type (e.g. IType), it can be assigned an instance of any subtype class (e.g. C1):

```
public class C1 implements IType {…}
public class C2 implements IType {…}

IType var;
var = new C1 (); // C1 is a subtype of IType
var = new C2 (); // C2 is a subtype of IType
```

MOVING ON

Point of variation

If a variable is declared to be of an interface type (e.g. IType), it can be assigned an instance of any subtype class (e.g. C1):

```
public class C1 implements IType {…}
public class C2 implements IType {…}

IType var;
var = new C1 (); // C1 is a subtype of IType
var = new C2 (); // C2 is a subtype of IType
```

var now is a point of variation in the code.

A method call on var is polymorphic: the outcome depends on definition of method in the subtypes of IType (i.e. C1 and C2)

Method restrictions

The declared type of a variable, not the actual type of the object the variable refers to, determines WHICH methods can be called on the object.

The actual type of the object on which a method is called, rather than the declared type of the variable, determines the behavior (the code executed).

We'll have more to say about this when we discuss the inheritance relationship.

Graphical User Interface (GUI)

Using the Java graphics classes

In these slides we will explain the basics of how to create graphical programs.

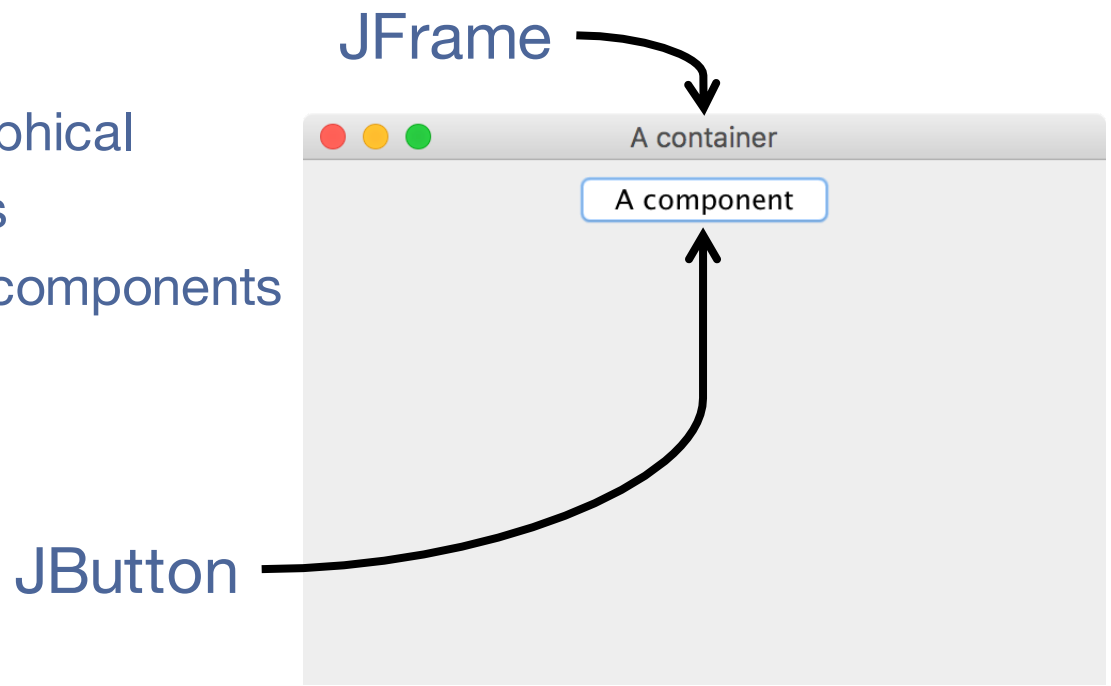
Some advanced issues will be glossed over (e.g. thread safety, graphics library design).

Graphical elements

There are two basic types of graphical elements:

Containers

able to hold graphical objects, such as containers and components



Components

must be put into containers

able to generate events when manipulated

Containers

Top-level containers

- some containers are called “top-level” because they do not need to be placed inside any other containers

- JFrame is a top-level container, meaning it can exist independently; a JFrame draws a window, complete with a title bar, scroll-bar, resize controls, etc.

Other containers (not top-level)

- most containers must be placed inside some other container
- javax.swing.JPanel is an example

javax.swing.JFrame

Top-level containers have multiple panes

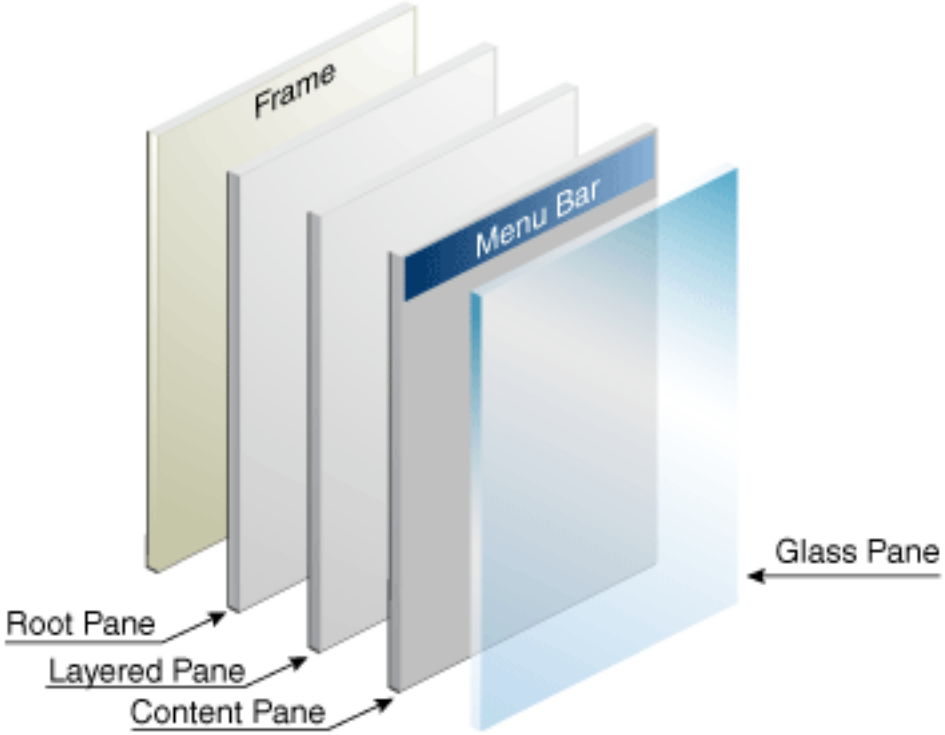


image credit: http://docs.oracle.com/javase/tutorial/uiswing/components/toplevel.html

Adding elements to a JFrame

We will add components to the content pane.

With javax.swing.JFrame, two ways:

- call `getContentPane()` on frame to get frame's content pane, then call `add(…)` on content pane to add a component
- call `add(…)` directly on the JFrame object

Second approach is just a convenience method, does the same thing the first approach.

Example

Creating just a frame

new javax.swing.JFrame()

Creating a frame with a title

new javax.swing.JFrame("My title")

Making the frame visible

call setVisible(true) on the frame

Making application close when window is closed:

call setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE) on the frame

Example

Creating just a frame

new javax.swing.JFrame()

Creating a frame with a title

new javax.swing.JFrame("My title")

Making the frame visible

call setVisible(true) on the frame

true is a value of the (primitive) type boolean.

Making application close when window is closed:

call setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE) on the frame

Live coding

See code in graphics package of LectureCode project: