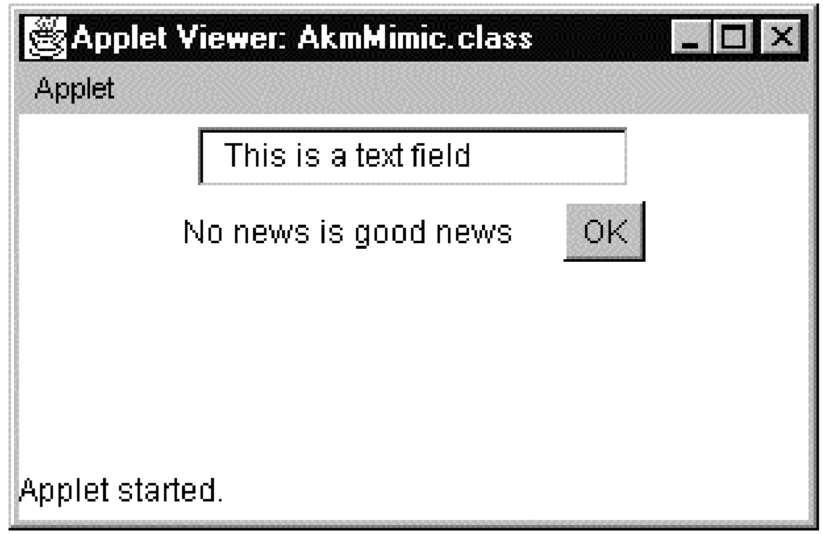
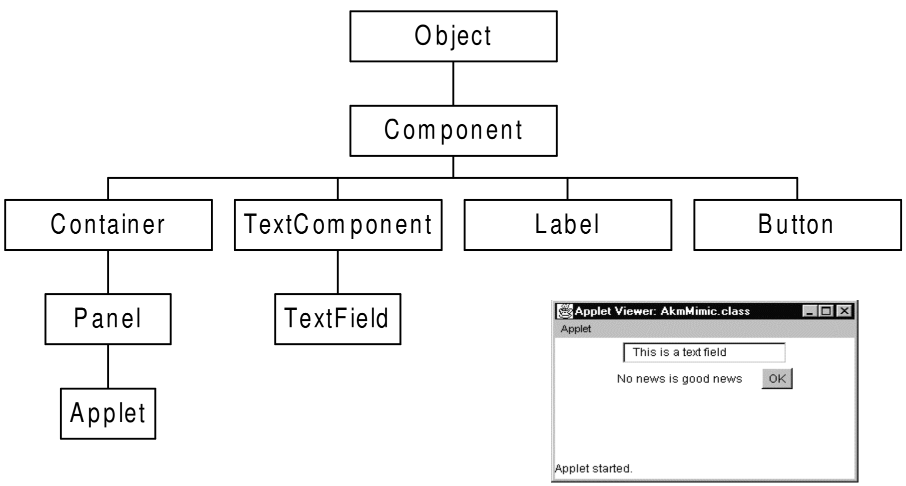
Programming in Java

Graphics and Graphical User Interfaces

Display of AkmMimic



Classes used in Mimic



Display Components

- Basic code to display components

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

public class AkmMimic extends Applet {
  public void init () {
    this.add (new TextField(20));
    this.add (new Label ("No news is good news"));
    this.add (new Button ("OK"));
  }
}
```

- Applet is a type of Container
  - Can add components to containers
  - Container takes care of "laying out the components"
  - Default behavior for Applets is "Flow Layout"
    – Left to right, goes to new rows as necessary

Events

- Events can occur on Components
  - There are many types of events
    - Action, Mouse, Key, Focus, Paint, Window
- Listeners watch for events on component
  - Listeners in Java are interfaces
    - (must be implemented by a class)
  - Examples:
    - ActionListener, MouseListener, MouseMotionListener
- Listeners are passed Event objects
  - Event objects contain information (e.g. x,y position of mouse)
  - Examples
    - ActionEvent, MouseEvent, KeyEvent

Events

- Different components can generate different events
  - Button, List, TextField, MenuItem
    - generate ActionEvents
  - Any component can generate MouseEvents
- To handle events, have to add listeners

```
Button b = new Button ("OK");
ActionListener al = … create an action listener …
b.addActionListener (al);
```

- ActionListener is an interface
  - Need a class that implements the interface

```
public abstract interface ActionListener
                          extends EventListener {
  public abstract void actionPerformed (ActionEvent e);
}
```


Listening to the OK button

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

public class AkmMimic extends Applet {
  public void init () {
    this.add (new TextField(20));
    this.add (new Label ("No news is good news"));
    Button b = new Button ("OK");
    ActionListener al = new OK_Handler();
    b.addActionListener (al);
    this.add (b);
  }
}

class OK_Handler implements ActionListener {
  public void actionPerformed (ActionEvent e) {
    System.out.println ("You pressed OK");
  }
}
```


Applets can also Listen

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

public class AkmMimic extends Applet
implements ActionListener {
  public void init () {
    this.add (new TextField(20));
    this.add (new Label ("No news is good news"));
    Button b = new Button ("OK");
    b.addActionListener (this);
    this.add (b);
  }

  public void actionPerformed (ActionEvent e) {
    System.out.println ("You pressed OK");
  }
}
```

Copying from TextField to Label

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

public class AkmMimic extends Applet
implements ActionListener {
  TextField textfield = new TextField(20);
  Label label = new Label ("No news is good news");
  public void init () {
    this.add (textfield);
    this.add (label));
    Button b = new Button ("OK");
    b.addActionListener (this);
    this.add (b);
  }

  public void actionPerformed (ActionEvent e) {
    label.setText (textfield.getText());
  }
}
```


Listeners

- Some listeners have multiple methods

```
public abstract interface MouseListener
extends EventListener {
  public abstract void mouseClicked(MouseEvent e);
  public abstract void mouseEntered(MouseEvent e);
  public abstract void mouseExited(MouseEvent e);
  public abstract void mousePressed(MouseEvent e);
  public abstract void mouseReleased(MouseEvent e);
};
```

- Classes that implement interfaces
  - MUST define code for all methods of the interface
    - If you only need code in one method, solution isn't clean

Handling Mouse Events

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

public class AkmMimic extends Applet {
  public void init () {
    this.add (new TextField(20));
    this.add (new Label ("No news is good news"));
    this.add (new Button("OK"));
    this.addMouseListener (new MyMouseHandler());
  }
}

class MyMouseHandler implements MouseListener {
  public void mouseClicked(MouseEvent e) { }
  public void mouseEntered(MouseEvent e) { }
  public void mouseExited(MouseEvent e) { }
  public void mousePressed(MouseEvent e) {
    System.out.println ("You pressed a mouse button");
  }
  public void mouseReleased(MouseEvent e) { }
}
```


Using Adapters to Listen

- For most listeners, there is a corresponding adapter

```
class MouseAdapter implements MouseListener {
  public void mouseClicked(MouseEvent e) { }
  public void mouseEntered(MouseEvent e) { }
  public void mouseExited(MouseEvent e) { }
  public void mousePressed(MouseEvent e) { }
  public void mouseReleased(MouseEvent e) { }
}
```

- You can reuse these, and only implement necessary methods

```
class MyMouseHandler extends MouseAdapter {
  public void mousePressed (MouseEvent e) {
    System.out.println ("You pressed a mouse button");
  }
}
```

Using MouseAdapter

```
mport java.applet.Applet;
import java.awt.*;
import java.awt.event.*;

public class AkmMimic extends Applet {
  public void init () {
    this.add (new TextField(20));
    this.add (new Label ("No news is good news"));
    this.add (new Button("OK"));
    this.addMouseListener (new MyMouseHandler());
  }
}

class MyMouseHandler extends MouseAdapter {
  public void mousePressed(MouseEvent e) {
    System.out.println ("You pressed a mouse button");
  }
}
```


Too many classes

- A typical GUI has many buttons, components, menu items, etc.
- Need a different class for each listener
  - Too many classes
  - Confuses name space
- Java's solution
  - Anonymous, inner classes
    - Class is embedded in another class
    - Class does not have to have a name provided by developer

Anonymous Classes

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

public class AkmMimic extends Applet {
  public void init () {
    this.add (new TextField(20));
    this.add (new Label ("No news is good news"));
    this.add (new Button("OK"));
    this.addMouseListener (new MouseAdapter () {
      public void mousePressed(MouseEvent e) {
        System.out.println ("You pressed a mouse button");
      }});
  }
}
```


Left, Right, or Middle Button?

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

public class AkmMimic extends Applet {
  public void init () {
    this.add (new TextField(20));
    this.add (new Label ("No news is good news"));
    this.add (new Button("OK"));
    this.addMouseListener (new MouseAdapter () {
      public void mousePressed(MouseEvent e) {
        if (e.isMetaDown()) {
          System.out.println ("Right mouse button pressed");
        } else if (e.isAltDown()) {
          System.out.println ("Middle mouse button pressed");
        } else {
          System.out.println ("Left mouse button pressed");
        }
      }});
  }
}
```