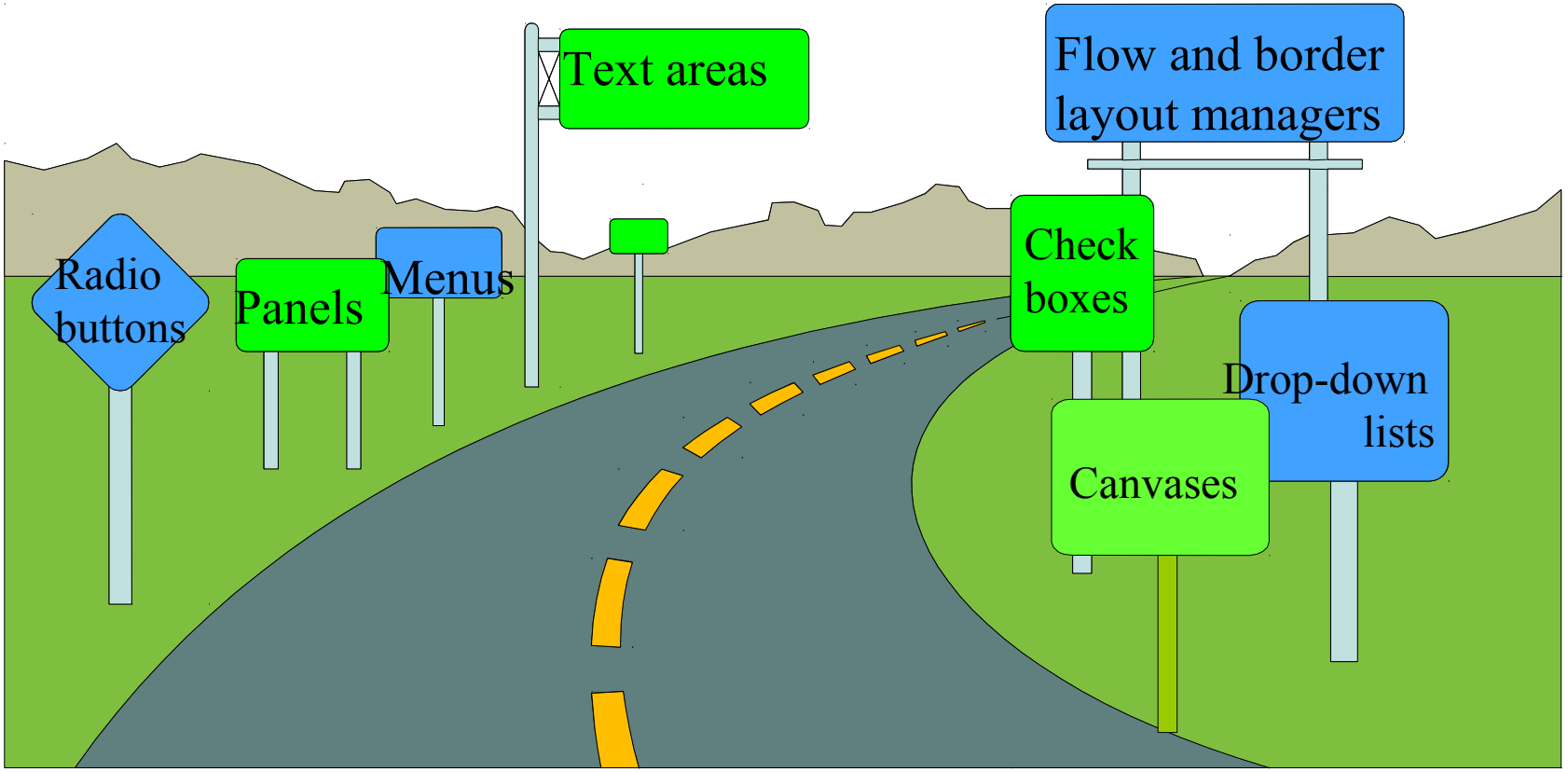
Java

GUI 1

DDOOCP

GUI
Graphical User Interface

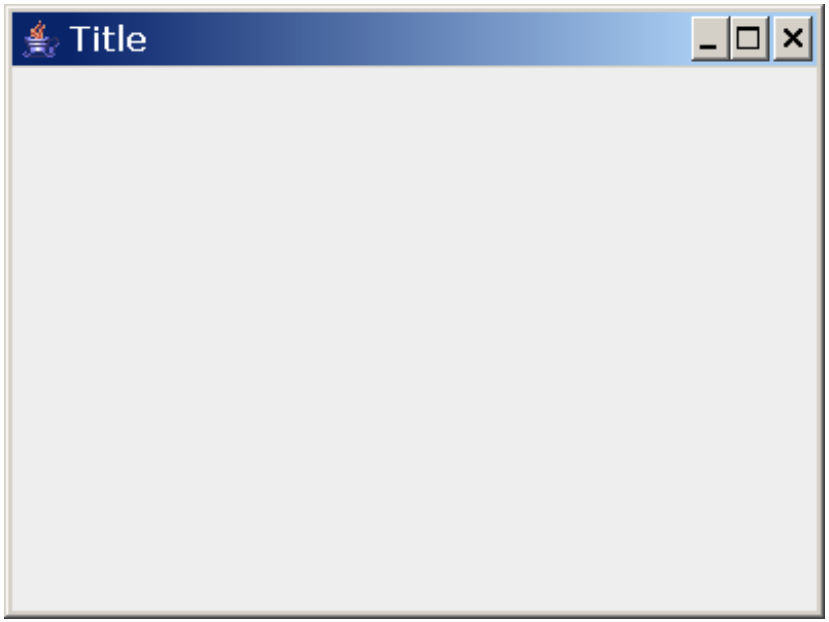


What makes up a GUI?

- This lecture shows how to write Java programs that run in windows, using *controls* such as
  - buttons,
  - text fields,
  - combo boxes,
  - check boxes,
  - radio buttons
- All of these (and many more) are defined in the *Swing* classes javax.swing.* which must be imported into your class.

Where do we start?

- With the background to our interface
- This is called the content pane
- It is part of the Java Swing library and uses a class called JFrame
- It looks very exciting!!



```
1  import java.awt.*;
2  import javax.swing.*;
3  import java.awt.event.*;
4
5  public class MyJF extends JFrame
6                implements ActionListener
7  {
8
9     public static void main(String[] args)
10    {
11       MyJF jf = new MyJF();
12    }
13
14    public MyJF()
15    {
16       setLayout(new FlowLayout());
17       setSize(400, 300);
18       setTitle("Title");
19       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20       setVisible(true);
21    }
22
23    public void actionPerformed(ActionEvent e)
24    {
25       // add your event handling code here
26    }
27 }
```

constructor

Event-Driven Programming Basics

- GUI programs usually use event-driven programming techniques.
- Basic idea behind event-driven programming:
  - The program waits for events to occur and then it responds.
- An *event* is a message that tells the program that something has happened. For example, if the user clicks a button, then an event is generated, and it tells the program that a particular button was clicked.
- More formally, when the user clicks a button, we say that the button object *fires an event*.

Event-Driven Programming Basics

- Note these additional event examples:

User Action	What Happens
Pressing the enter key while the cursor is inside a text box.	The text box object fires an event, and it tells the program that enter was pressed within the text box.
Clicking a menu item.	The menu item object fires an event, and it tells the program that the menu item was selected.
Closing a window (clicking on the window's top-right corner "X" button)	The window object fires an event, and it tells the program that the window's close button

Event-Driven Programming Basics

- If an event is fired, and you want your program to handle the fired event, then you need to create a *listener* for the event.
- A listener is actually an object that you instantiate, but don't worry about such details just yet.
- For now, think of a listener as an ear.
- For example, if you want your program to do something when the user clicks a particular button, you need to create a listener for the button.
- If an event is fired and there's no listener listening to it, then the fired event is never "heard" and there's no response to it.
- On the other hand, if there is a listener listening to a fired event, then the listener "hears" the event and the program then responds to the fired event.
- The way the program responds is by executing a chunk of code known as an *event handler*.

Event-Driven Programming Basics

- What happens when a button is pressed:

a button is pressed

an event being fired

listener hearing the event

event handler being executed

```
1   package ch04_gui1;
2
3   import java.awt.*;
4   import javax.swing.*;
5   import java.awt.event.*;
6
7   public class JFrameBasic
8           extends JFrame implements ActionListener
9
10      public static void main(String[] args) {
11          JFrameBasic jf = new JFrameBasic();
12      }
13
14      public JFrameBasic() {
15          setLayout(new FlowLayout());
16          setSize(400, 300);
17          setTitle("Title");
18          setDefaultCloseOperation(JFrame.EXIT_ON_
19          setLocationRelativeTo(null);
20          setVisible(true);
21      }
22
23      public void actionPerformed(ActionEvent e) {
24          // add your event handling code here
25      }
26  }
```

these are the Java libraries you will be using

awt is the abstract windows toolkit and adding **event** means it can interact with the user clicking buttons etc

```
1   package ch04_gui1;
2
3   import java.awt.*;
4   import javax.swing.*;
5   import java.awt.event.*;
6
7   public class JFrameBasic
8           extends JFrame implements ActionListener {
9
10      public static void main(String[] args) {
11          JFrameBasic jf = new JFrameBasic();
12      }
13
14      public JFrameBasic() {
15          setLayout(new FlowLayout());
16          setSize(400, 300);
17          setTitle("Title");
18          setDefaultCloseOperation(JFrame.EXIT_
19          setLocationRelativeTo(null);
20          setVisible(true);
21      }
22
23      public void actionPerformed(ActionEvent e
24          // add your event handling code here
25      }
26  }
```

means we can use the features of the class JFrame as this is its subclass

implements ActionListener says that we want our application to "listen" for *events* such as the user pressing a button (writing this means that the class must have an actionPerformed method)

```
1   package ch04_gui1;
2
3   import java.awt.*;
4   import javax.swing.*;
5   import java.awt.event.*;
6
7   public class JFrameBasic
8           extends JFrame implements ActionListener {
9
10      public static void main(String[] args) {
11          JFrameBasic jf = new JFrameBasic();
12      }
13
14      public JFrameBasic() {
15          setLayout(new FlowLayout());
16          setSize(400, 300);
17          setTitle("Title");
18          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
19          setLocationRelativeTo(null);
20          setVisible(true);
21      }
22
23      public void actionPerformed(ActionEvent e) {
24          // add your event handling code here
25      }
26  }
```

main method as this is not just a class but an application

just creates a new object of the JFrameBasic class

```
1   package ch04_gui1;
2
3   import java.awt.*;
4   import javax.swing.*;
5   import java.awt.event.*;
6
7   public class JFrameBasic
8           extends JFrame implements      nListener {
9
10      public static void main(St     g[] args) {
11          JFrameBasic jf = ne     rameBasic();
12      }
13
14      public JFrameBasic() {
15          setLayout(new FlowLayout());
16          setSize(400, 300);
17          setTitle("Title");
18          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE
19          setLocationRelativeTo(null);
20          setVisible(true);
21      }
22
23      public void actionPerformed(ActionEvent e) {
24          // add your event handling code here
25      }
26  }
```

the instructions to make had no parameters () so we need the constructor with no parameters

the constructor (with no parameters) just sets up the appearance and behaviour of the object using 6 basic methods inherited from JFrame

```
1   package ch04_gui1;
2
3   import java.awt.*;
4   import javax.swing.*;
5   import java.awt.event.*;
6
7   public class JFrameBasic
8           extends JFrame implements ActionListener {
9
10      public static void main(String[] args) {
11          JFrameBasic jf = new JFrameBasic();
12      }
13
14      public JFrameBasic() {
15          setLayout(new FlowLayout());
16          setSize(400, 300);
17          setTitle("Title");
18          setDefaultCloseOperation(JFrame.EXIT_ON_CL
19          setLocationRelativeTo(null);
20          setVisible(true);
21      }
22
23      public void actionPerformed(ActionEvent e) {
24          // add your event handling code here
25      }
26  }
```

The code to be executed when the event occurs goes in this method

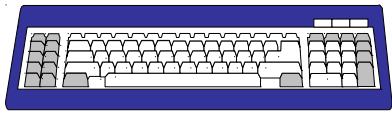
JFrame

…..

+setLayout(LayoutManager)
+setSize(int,int)
+setTitle (String)
+setDefaultCloseOperation(int)
+setVisible(boolean)
……..

can set different Layout Managers as parameter

can set different specified operations as parameter, e.g. DISPOSE_ON_CLOSE

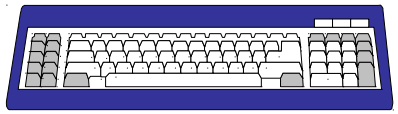


```
14      public JFrameBasic() {
15          setLayout(new FlowLay
16          setSize(400, 300);
17          setTitle("Title");
18          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
19          setLocationRelativeTo(null);
20          setVisible(true);
21
```

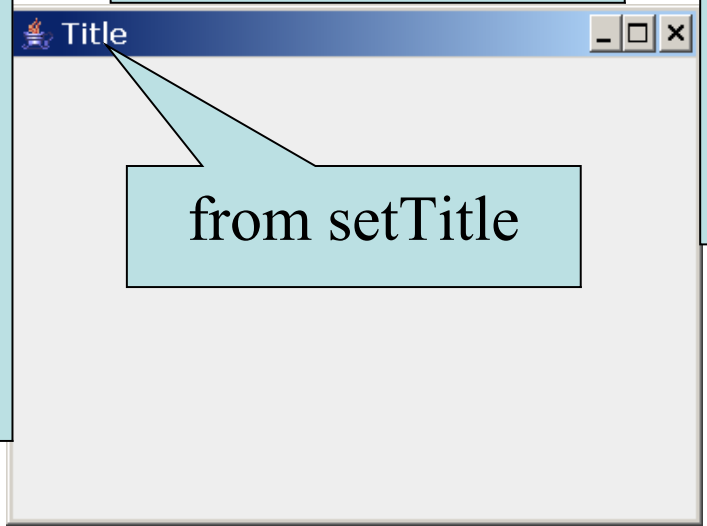
the content pane of the frame is 392 by 265 pixels.
(we lose 8 sideways for the border and 35 for the top title bar)

needed to make the frame visible.

allows the user to close the application by clicking the close window button



FlowLayout adds controls from top to bottom and left to right in a manner similar to writing text in Word (centrally justified)



from setTitle

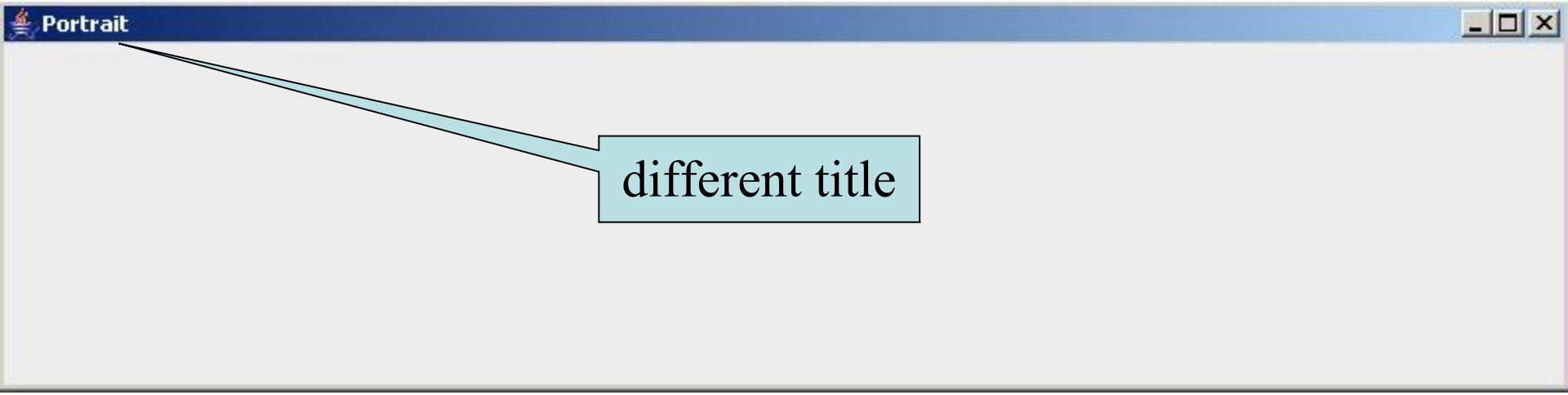
A different example using the template with class renamed FirstFrame

```
public FirstFrame()
{
   setLayout(new FlowLayout());
   setSize(800, 200);
   setTitle("Portrait");
   setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   setVisible(true);
}
```

different size

different title

different size



different title

Parts of 3

Buttons

- Creating our own buttons gives us more flexibility, by using

```
JButton myButton = new JButton("Text");
```

Adding a button- 3 parts

1

```
JButton myButton = new JButton("Press");
```

2

```
add(myButton);

myButton.addActionListener(this);
```

3

```
public void actionPerformed(ActionEvent e)
    {
      setTitle("pressed");
    }
```

Adding a button- 3 parts

```
JButton myButton = new JButton("Press");
```

Make a new JButton object called myButton using the constructor from javax.swing which takes a String parameter and puts it on the button

Adding a button- 3 parts

adds the button to the content pane

```
add(myButton);

myButton.addActionListener(this);
```

It *registers* the button with the action listener for the frame (identified by this). When the user clicks the button, the actionPerformed method is called automatically

Adding a button- 3 parts

this is the method which is automatically called when the ActionListener 'hears' that the button has been clicked (pressed by the mouse)

```
public void actionPerformed(ActionEvent e)
    {
      setTitle("pressed");
    }
```

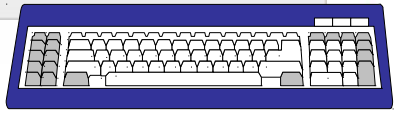
this is what will happen

```
1  import java.awt.*;
2  import javax.swing.*;
3  import java.awt.event.*;
4
5  public class FirstFrame2 extends JFrame
6                implements ActionListener
7  {
8
9    JButton myButton = new JButton("Press");
10
11    public static void main(String[] args)
12    {
13       FirstFrame2 ff = new FirstFrame2();
14    }
15
16    public FirstFrame2()
17    {
18       setLayout(new FlowLayout());
19       setSize(400, 400);
20       setTitle("Button");
21       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
22       add(myButton);
23       myButton.addActionListener(this);
24
25       setVisible(true);
26    }
27
28    public void actionPerformed(ActionEvent e)
29    {
30      setTitle("pressed");
31    }
32 }
```

FirstFrame2.java

Spot the difference

```
1  import java.awt.*;
2  import javax.swing.*;
3  import java.awt.event.*;
4
5  public class FirstFrame3 extends JFrame
6              implements ActionListener
7  {
8    ImageIcon ic = new ImageIcon("ctree.gif");
9    JButton myButton = new JButton("Press", ic);
10
11    public static void main(String[] args)
12    {
13       FirstFrame3 ff = new FirstFrame3();
14    }
15
16    public FirstFrame3()
17    {
18       setLayout(new FlowLayout());
19       setSize(400, 400);
20       setTitle("Button");
21       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
22       add(myButton);
23       myButton.addActionListener(this);
24
25       setVisible(true);
26    }
27
28    public void actionPerformed(ActionEvent e)
29    {
30      setTitle("Happy Christmas");
31    }
32 }
```



FirstFrame3.java

Text Field

- It is not very sophisticated to send a message to the user via the title bar of the frame
- We need a way to display text and amongst others Java uses textfields

```
JTextField myText = new JTextField(10);
```

Make a new JTextField object called myText using the constructor from javax.swing which takes an int parameter and makes a text area with that width (here 10 characters)

JTextField

…..

+setText(s:String)
+getText(): String
……..

Package javax.swing it has many different methods including those to read the String from the box and display a String in the box

Adding a TextField- 2 parts

1

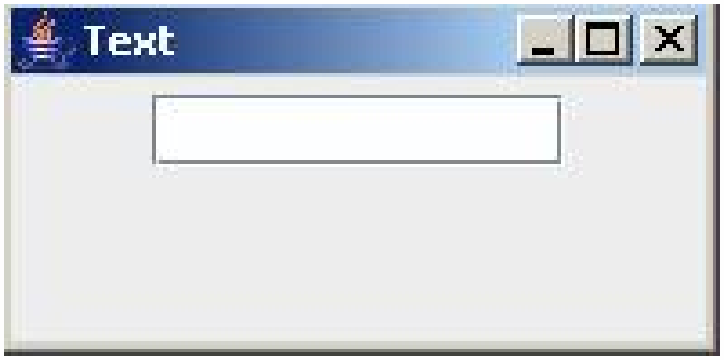
```
JTextField myText = new JTextField(10);
```

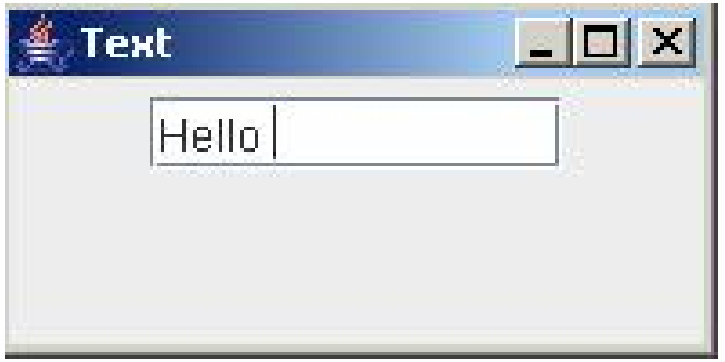
2

```
add(myText);
```

HelloText.java

```
1  import java.awt.*;
2  import javax.swing.*;
3  import java.awt.event.*;
4
5  public class HelloText extends JFrame
6              implements ActionListener
7  {
8     JTextField myText = new JTextField(10);
9
10
11    public static void main(String[] args)
12    {
13       HelloText jt = new HelloText();
14    }
15
16    public HelloText()
17    {
18       setLayout(new FlowLayout());
19       setSize(200, 100);
20       setTitle("Text");
21       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
22
23       add(myText);
24
25
26       setVisible(true);
27    }
28
29    public void actionPerformed(ActionEvent e)
30    {
31
32    }
33 }
```

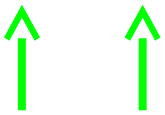




Now both together!

- Often we want the user to positively press a submit button rather than just press a return which they might do automatically at the end of typing
- In the program HelloWorld (see Java workbook) the text is changed in response to a button action so we have both button and text field added to our content pane

```
1  package gui1;
2
3  import java.awt.*;
4  import javax.swing.*;
5  import java.awt.event.*;
6
7  public class HelloWorldDone
8        extends JFrame implements ActionListener {
9
10    JTextField helloText = new JTextField(10);
11    JButton helloButton = new JButton("Press Me");
12
13    public static void main(String[] args) {
14       HelloWorldDone jf = new HelloWorldDone();
15    }
16
17    public HelloWorldDone() {
18       setLayout(new FlowLayout());
19       setSize(200, 100);
20       setTitle("Hello");
21       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
22       setLocationRelativeTo(null);
23
24       add(helloText);
25       add(helloButton);
26       helloButton.addActionListener(this);
27
28       setVisible(true);
29    }
30
31    public void actionPerformed(ActionEvent e) {
32       helloText.setText("Hello World!");
33    }
34 }
```



```
JTextField helloText = new JTextField(10);
JButton helloButton = new JButton("Press Me");
```

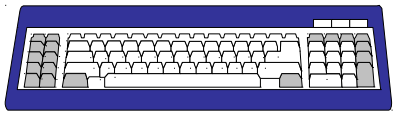
setting up the text field and button and making sure the button is listened to

```
add(helloText);
add(helloButton);
```

```
helloButton.addActionListener(this);
```

```
public void actionPerformed(ActionEvent e)
    {
     helloText.setText("Hello World !");
    }
```

setting the text to read Hello World! when the button is pressed



HelloWorld.java

```
JTextField helloText = new JTe
JButton helloButton = ne
```

JTextField

.....

+setText(s:String)

+getText(): String

.....

```
a
a
he                                    ener(this);
```

```
public void actionPerf    med(ActionEvent e)
    {
     helloText.setText("Hello World !");
    }
```

Check Boxes

- Check boxes give you a quick way of getting input from the user when there is a choice of 2 options.
- They can easily be used for
  - yes/no
  - male/female
- The code for making a new check box object is

```
JCheckBox myCheckBox = new JCheckBox("Text");
```

Adding a CheckBox- 3 parts

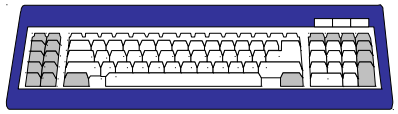
```
JCheckBox myChBx = new JCheckBox("Try checking it");
```

create a checkbox object called myChBx with the text **Try checking it** at its side

```
add(myChBx);

myChBx.addActionListener(this);
```

set the action to be triggered if someone checks it

```
public void actionPerformed(ActionEvent e)
    {
        myTxt.setText("you did it!");
    }
```



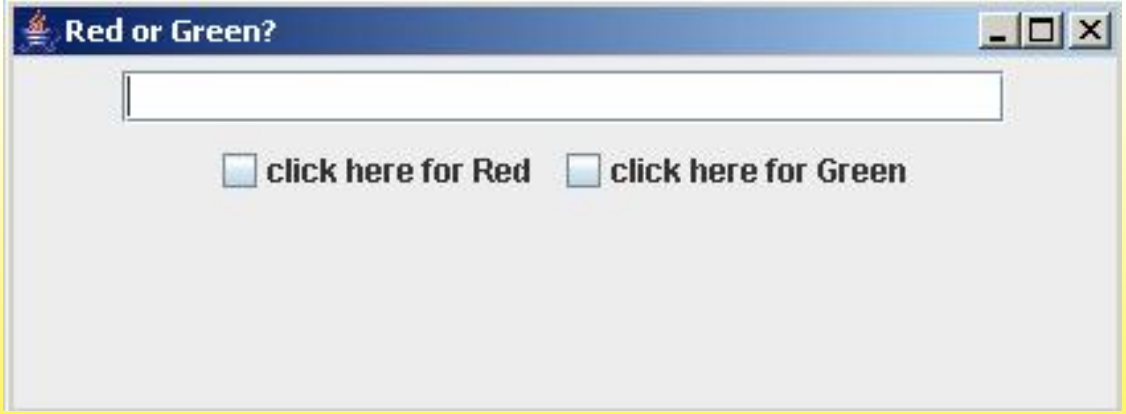
message to appear in a text field if box is ticked

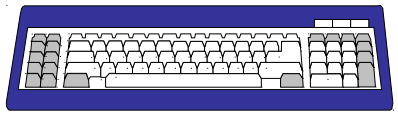
CheckText.java

```
1  import java.awt.*;
2  import javax.swing.*;
3  import java.awt.event.*;
4
5  public class CheckText extends JFrame
6                implements ActionListener
7  {
8     JTextField myTxt = new JTextField(10);
9     JCheckBox myChBx = new JCheckBox("Try checking it");
10
11    public static void main(String[] args)
12    {
13       CheckText jf = new CheckText();
14    }
15
16    public CheckText()
17    {
18       setLayout(new FlowLayout());
19       setSize(600, 120);
20       setTitle("Check Box Example");
21       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
22
23       add(myTxt);
24       add(myChBx);
25       myChBx.addActionListener(this);
26       setVisible(true);
27    }
28
29    public void actionPerformed(ActionEvent e)
30    {
31      myTxt.setText("you did it!");
32     }
33 }
```

Slightly more complex

- If we extend this example to give us **2 check boxes** we come across several problems
- How do we know **which one** is checked?
- Can we check **both**?
- What happens if we **resize**?
- Can we get the **layout** to look better.
- Each of these problems leads us to more sophisticated coding





CheckText2 (closing)

How do we know which one is chosen?

```
public void actionPerformed(ActionEvent e)
   {
        if (e.getSource() == myRedChBx)
        {
                myTxt.setText("you have chosen Red");
        }
        else if (e.getSource() == myGreenChBx)
        {
                myTxt.setText("you have chosen Green");
        }
  }
```

we can use the statement `if (e.getSource() == myChBx)`

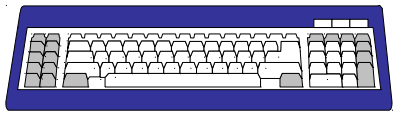
to test for the source of the event e that has triggered the actionPerformed method

How do we know which one is chosen?

```
public void actionPerformed(ActionEvent e)
{
  if (myRedChBx.isSelected())
  {
        myTxt.setText("you have chosen Red");
  }
  else if (myGreenChBx.isSelected())
  {
        myTxt.setText("you have chosen Green");
  }
}
```

Alternatively we can use the test if (myChBx.isSelected())

to test for the source of the event e that has triggered the actionPerformed method

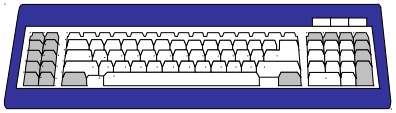


CheckText2A

Can we check both?

- Oops yes! It will change the text according to the last one to be clicked (even if that un-checks it)
- If you want to only have one checked at a time you have to put them in a button group
- Although the group can contain checkboxes, it is more common to use radio buttons
- The code for making a group object is

ButtonGroup myBG = new ButtonGroup();



```
8     JTextField myTxt = new JTextField(30);
9     JRadioButton myRedRB = new JRadioButton("click here for Red", true);
10    JRadioButton myGreenRB = new JRadioButton("click here for Green", false);
11    ButtonGroup colours = new ButtonGroup();
```

```
public RadioText()
{
    setLayout(new FlowLayout());
    setSize(600, 120);
    setTitle("Red or Green?");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    colours.add(myRedRB);
    colours.add(myGreenRB);
    add(myTxt);
    add(myRedRB);
    add(myGreenRB);

    myRedRB.addActionListener(this);
    myGreenRB.addActionListener(this);
    setVisible(true);
}
```

sets up the ButtonGroup object and calls it colours

creates the two buttons objects and sets the red one 'on' at start

adds both the buttons to the group to ensure only one can be clicked at a time

adds both the text field and buttons to the content pane

registers both buttons

```
public void actionPerformed(ActionEvent e)
{
  if (e.getSource() == myRedRB)
  {
      myTxt.setText("you have chosen Red");
  }
  else if (e.getSource() == myGreenRB)
  {
      myTxt.setText("you have chosen Green");
  }
}
```

Resizing and Layout

- These p ys
- The statement

  setResizable(false);

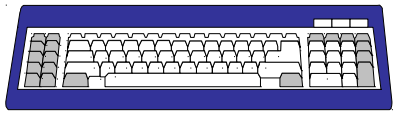
  in the constructor of the class makes the JFrame a fixed size.
- The examples later and in the notes show the use of different layout managers

  setLayout(new FlowLayout());

  setLayout(new GridLayout(int, int));

  setLayout(new BorderLayout());

  in the design of the frame



add line to RadioText

Using text

- So far we have seen text in one of 3 ways
- On a button
  - (here **Press** will appear on the button)

```
JButton myButton = new JButton("Press");
```

- With a checkbox or radio button
  - (here **Try checking it** will at the side of the check box)

```
JCheckBox myChBx = new JCheckBox("Try checking it");
```

- In a text field
  - (here **Hello** will appear in the textfield)

```
myText.setText("Hello");
```

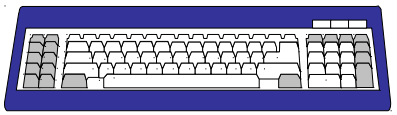
Labels

- These are used to make the user interface more readable and give instructions but are limited as they are intended to be fixed once set
- They are added using the Java

```
add(new Label("String"));
```

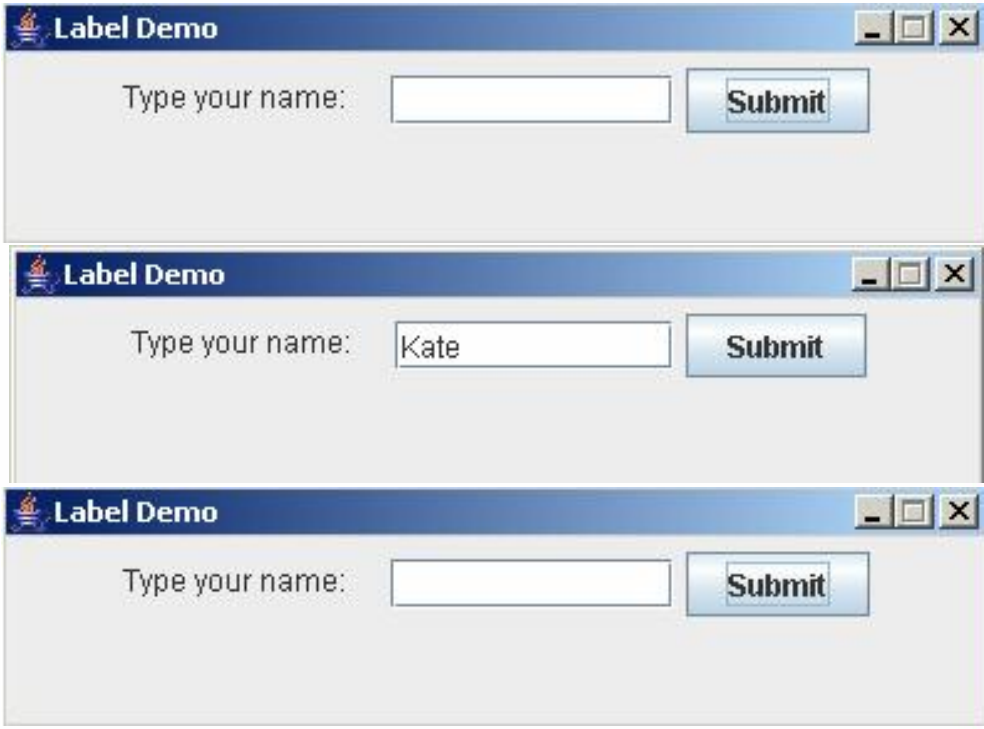
e.g.

```
add(new Label ("first label :"));
add(new Label ("second label :"));
add(new Label ("third label :"));
add(new Label ("fourth label :"));
add(new Label ("fifth label :"));
```



LabelDemo2 first then LabelDemo

```
1  import java.awt.*;
2  import javax.swing.*;
3  import java.awt.event.*;
4
5  public class LabelDemo extends JFrame
6              implements ActionListener
7  {
8     JTextField nameTxt = new JTextField(10);
9     JButton sub = new JButton("Submit");
10
11    public static void main(String[] args)
12    {
13       LabelDemo jf = new LabelDemo();
14    }
15
16    public LabelDemo()
17    {
18       setLayout(new FlowLayout());
19       setSize(400, 100);
20       setTitle("Label Demo");
21       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
22
23       add(new Label("Type your name:"));
24       add(nameTxt);
25       add(sub);
26       sub.addActionListener(this);
27       setVisible(true);
28       setResizable(false);
29    }
30
31    public void actionPerformed(ActionEvent e)
32    {
33       nameTxt.setText("");
34    }
```



label added here

only listening for the button using keyboard return in the text field has no effect

pressing submit clears the text field

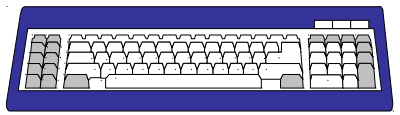
Text Area

- A text area is less restrictive than a text field and allows us to have quite a bit of text. It is instantiated using

  JTextArea myArea = new JTextArea(int, int);

- The two integers in this constructor control the number of lines and the width in characters of the text area

- In this next program we will also demonstrate the **getText** method so that we can use some text the user has put into the text field



TextAreaDemo

```
1  import java.awt.*;
2  import javax.swing.*;
3  import java.awt.event.*;
4
5  public class TextAreaDemo extends JFrame
6              implements ActionListener
7  {
8     JTextField nameTxt = new JTextField(10);
9     JTextArea output = new JTextArea(2, 30);
10    JButton sub = new JButton("Submit");
11
12    public static void main(String[] args)
13    {
14       TextAreaDemo jf = new TextAreaDemo();
15    }
16
17    public TextAreaDemo()
18    {
19       setLayout(new FlowLayout());
20       setSize(400, 120);
21       setTitle("Text Area Demo");
22       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
23
24       add(new Label("Type your name:"));
25       add(nameTxt);
26       add(sub);
27       sub.addActionListener(this);
28       add(output);
29       output.setEditable(false);
30       setVisible(true);
31    }
32
33    public void actionPerformed(ActionEvent e)
34    {
35       String name = nameTxt.getText();
36       String message = "Hello " + name + " \nEnjoy your programming ";
37       output.setText(message);
38    }
39 }
```

creating 3 objects
a text field, a text area
and a button

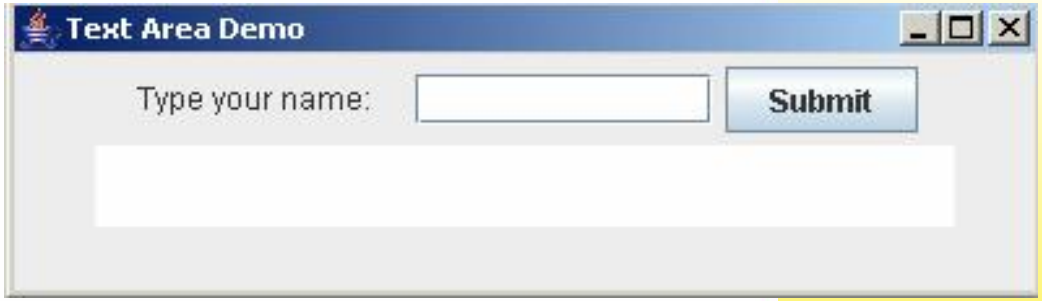
adding the objects to the content pane and registering the button with the ActionListener

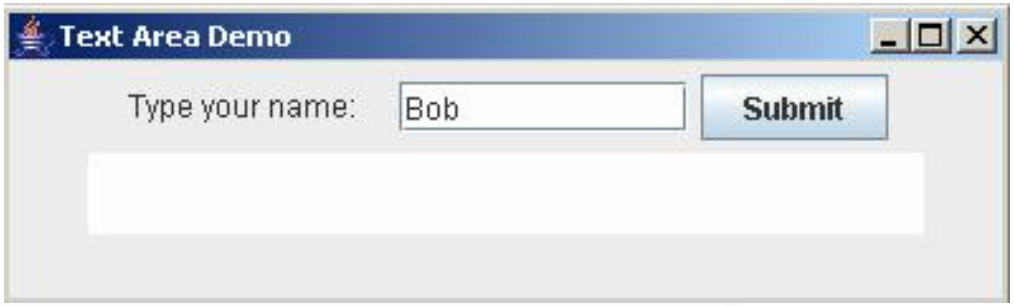
using the getText methods form the JTextField class to give the String in the text field to the variable name

creating the message to send to the text area output

using the setText method of JTextArea class

```
1  import java.awt.*;
2  import javax.swing.*;
3  import java.awt.event.*;
4
5  public class TextAreaDemo extends JFrame
6              implements ActionListener
7  {
8     JTextField nameTxt = new JTextField(10);
9     JTextArea output = new JTextArea(2, 30);
10    JButton sub = new JButton("Submit");
11
12    public static void main(String[] args)
13    {
14       TextAreaDemo jf = new TextAreaDemo();
15    }
16
17    public TextAreaDemo()
18    {
19       setLayout(new FlowLayout());
20       setSize(400, 120);
21       setTitle("Text Area Demo");
22       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
23
24       add(new Label("Type your name:"));
25       add(nameTxt);
26       add(sub);
27       sub.addActionListener(this);
28       add(output);
29       output.setEditable(false);
30       setVisible(true);
31    }
32
33    public void actionPerformed(ActionEvent e)
34    {
35       String name = nameTxt.getText();
36       String message = "Hello " + name + " \nEnjoy your programming ";
37       output.setText(message);
38    }
39 }
```





Combo Boxes

- When you have too many choices for radio buttons or you want to get the user to select a choice from a given list you can use a drop down list or combo box
- This is done using the statement

```
JComboBox myCBox = new JComboBox();
```

How do we put items in the box?

We need to add **Strings** to the drop down box using the Java method **addItem**

if we have the definition

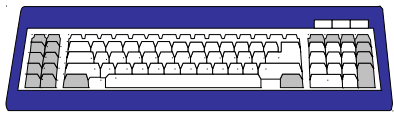
```
JComboBox myCBox = new JComboBox();
```

then we have statements like

```
myCBox.addItem("Item1");
myCBox.addItem("Item2");
myCBox.addItem("Item3");
```

we use the method getSelected to find out which one has been chosen

```
if (myCBox.getSelectedItem()=="Item2")
```



SimpleCombo.java

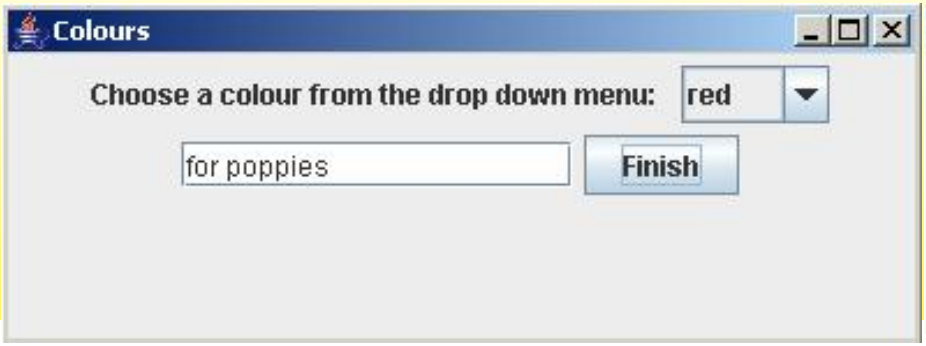
Adding a Combo Box 3 parts

```
1  import java.awt.*;
2  import javax.swing.*;
3  import java.awt.event.*;
4
5  public class SimpleCombo extends JFrame
6            implements ActionListener
7  {
8    JComboBox colour = new JComboBox();
9    JLabel instrLabel = new JLabel("Choose a colour from the drop down menu:  ");
10   JTextField colourTxt = new JTextField(15);
11    JButton doneBtn = new JButton("Finish");
12
13    public static void main(String[] args)
14    {
15       SimpleCombo sc = new SimpleCombo();
16    }
17
18    public SimpleCombo()
19    {
20       setLayout(new FlowLayout());
21       setSize(400, 150);
22       setTitle("Colours");
23       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
24       colour.addItem("red");
25       colour.addItem("yellow");
26       colour.addItem("blue");
27       colour.addItem("green");
28       add(instrLabel);
29       add(colour);
30       add(colourTxt);
31       add(doneBtn);
32       doneBtn.addActionListener(this);
33       setVisible(true);
34    }
35
36    public void actionPerformed(ActionEvent e)
37    {
38       if (colour.getSelectedItem()=="red") colourTxt.setText("for poppies");
39       else if (colour.getSelectedItem()=="yellow") colourTxt.setText (" like the sun");
40       else if (colour.getSelectedItem()=="blue") colourTxt.setText( " like the sky ");
41       else  colourTxt.setText(" like the grass");
42    }
43 }
```

make a ComboBox

add the items to the ComboBox and the ComboBox to the content pane

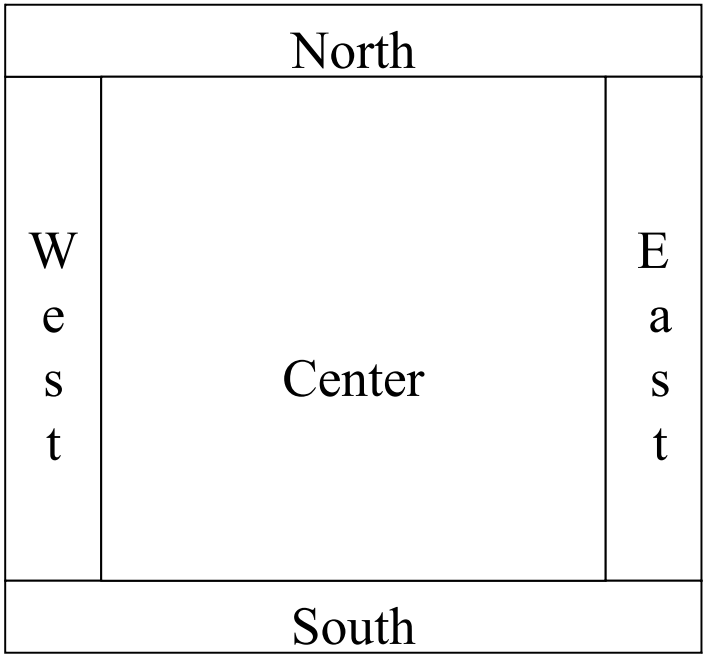
have actions triggered by button and dependent on the choice made

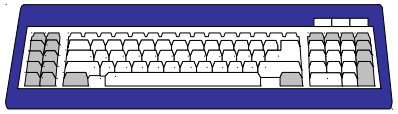


different text displayed
in the text field according
to the selection made

Layout Managers

- This example using Combo boxes also has some layout managing
- Border layout splits the content pane (or other container such as a *panel*) into up to five regions North, South, East, West and Center (note the American spelling)





ComboText.java

```
1  import java.awt.*;
2  import javax.swing.*;
3  import java.awt.event.*;
4
5  public class ComboText extends JFrame
6                  implements ActionListener
7  {
8     JComboBox tut = new JComboBox();
9     JTextArea commentTxt = new JTextArea(2,15);
10    JButton doneBtn = new JButton("Finish");
11
12    public static void main(String[] args) { new ComboText(); }
13
14    public ComboText()
15    {
16       setLayout(new BorderLayout());
17       setSize(400, 150);
18       setTitle("ComboText Demo");
19       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20      JPanel top = new JPanel();
21       top.setLayout(new FlowLayout());
22       tut.addItem("Don");
23       tut.addItem("Kate");
24       tut.addItem("Chris");
25       top.add (new Label("choose your favourite tutor"));
26       top.add (tut);
27       add("North", top);
28       add("Center", commentTxt);
29       add("South", doneBtn);
30       doneBtn.addActionListener(this);
31       setVisible(true);
32    }
33
34    public void actionPerformed(ActionEvent e)
35    {
36       String com;
37       if (tut.getSelectedItem()=="Don") com = " you must like programming";
38       else if (tut.getSelectedItem()=="Kate") com = " you think that will get you e
39       else  com = " you did not say which Chris";
40       commentTxt.setText(" I see that "+ com);
41    }
42 }
```

making 3 objects
a combo box , a text area and a button

note short version of method

giving the frame borders

making a panel with FlowLayout for the North border to organise the label & combo box

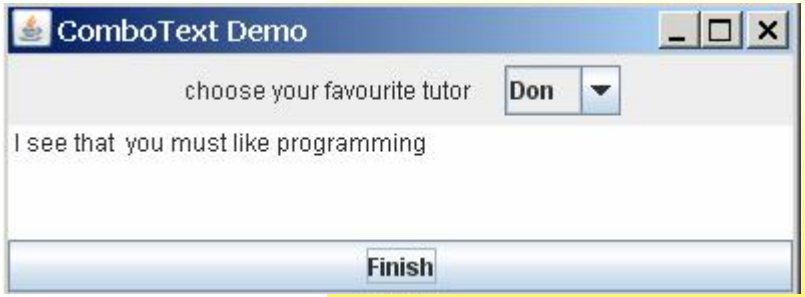
adding the items to the combo box

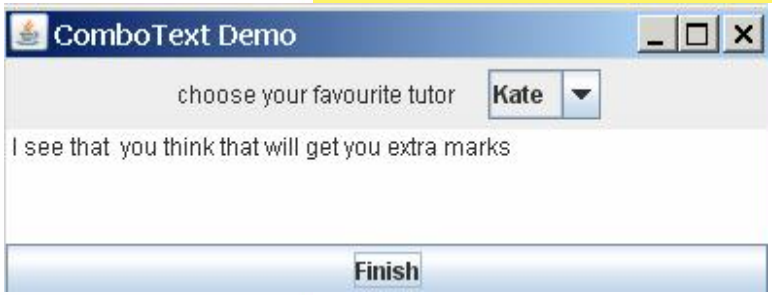
putting the label & the combo box on the panel then adding the panel to the North border

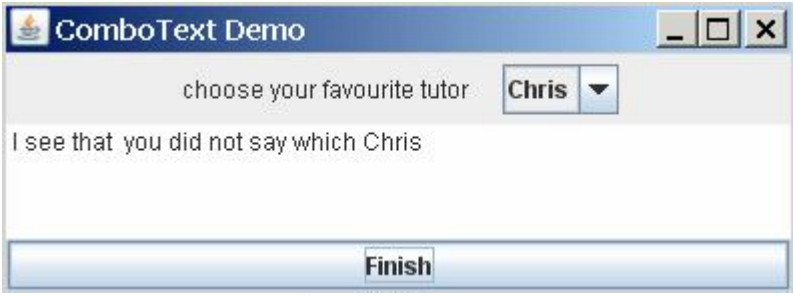
putting the text area in the Center border and the button on the South border

in response to a click on the Finish button a different comment is put depending on the combo box selection

```
1  import java.awt.*;
2   import javax.swing.*;
3   import java.awt.event.*;
4
5   public class ComboText extends JFrame
6                   implements ActionListener
7   {
8      JComboBox tut = new JComboBox();
9      JTextArea commentTxt = new JTextArea(2,15);
10     JButton doneBtn = new JButton("Finish");
11
12     public static void main(String[] args) { new ComboText(); }
13
14     public ComboText()
15     {
16        setLayout(new BorderLayout());
17        setSize(400, 150);
18        setTitle("ComboText Demo");
19        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20       JPanel top = new JPanel();
21        top.setLayout(new FlowLayout());
22        tut.addItem("Don");
23        tut.addItem("Kate");
24        tut.addItem("Chris");
25        top.add (new Label("choose your favourite tutor"));
26        top.add (tut);
27        add("North", top);
28        add("Center", commentTxt);
29        add("South", doneBtn);
30        doneBtn.addActionListener(this);
31        setVisible(true);
32     }
33
34     public void actionPerformed(ActionEvent e)
35     {
36        String com;
37        if (tut.getSelectedItem()=="Don") com = " you must like programming";
38        else if (tut.getSelectedItem()=="Kate") com = " you think that will get you extra marks";
39        else  com = " you did not say which Chris";
40        commentTxt.setText(" I see that "+ com);
41     }
42  }
```



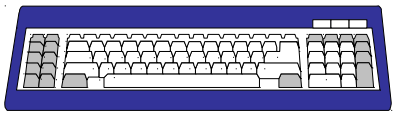




Another example of Layout manager

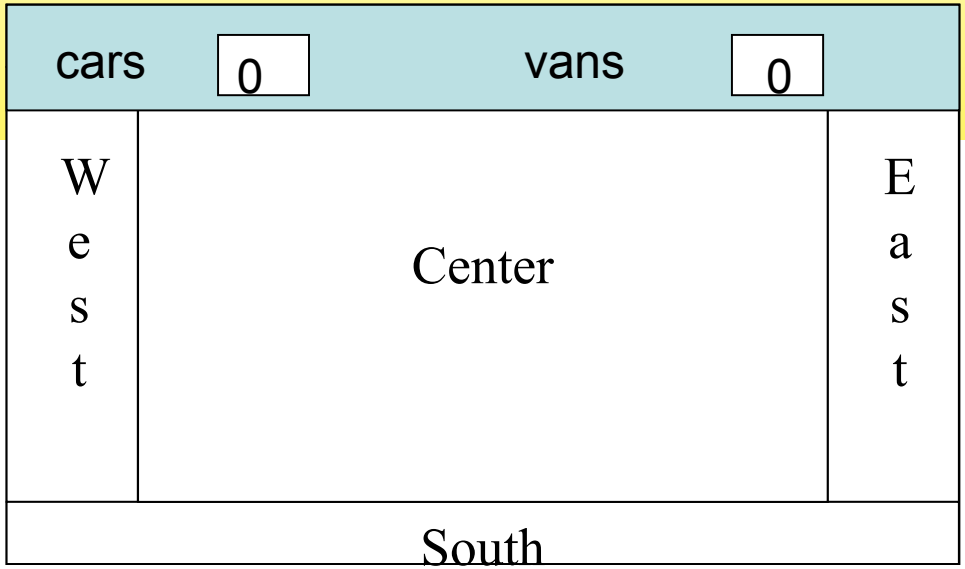
- In the Vehicles example (Java workbook) we can see again the use of different areas of the GUI being assigned various elements
- This can get very complicated and so it is useful to keep a few sample programs and just adapt them

Vehicles.java



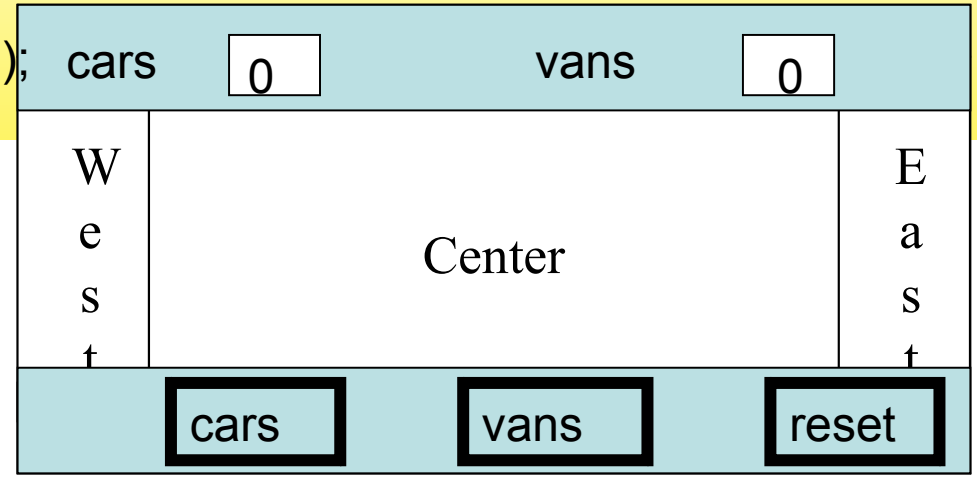
Layout of Cars & Vans

```
setLayout(new BorderLayout());

JPanel top = new JPanel();
top.setLayout(new FlowLayout());
top.add(new Label("Cars:"));
top.add(carsTxt);
carsTxt.setEditable(false);
top.add(new Label("          Vans:"));
top.add(vansTxt);
vansTxt.setEditable(false);
carsTxt.setText("0");
vansTxt.setText("0");


add("North", top);
```



Layout of Cars & Vans

```
JPanel bottom = new JPanel();
bottom.setLayout(new FlowLayout());
      bottom.add(carsBtn);
bottom.add(vansBtn);
bottom.add(reset);
carsBtn.addActionListener(this);
vansBtn.addActionListener(this);
reset.addActionListener(this);
```




```
add("South", bottom);
```

```
1  import java.awt.*;
2  import javax.swing.*;
3  import java.awt.event.*;
4
5  public class Vehicles extends JFrame
6            implements ActionListener
7  {
8     JTextField carsTxt = new JTextField(3);
9     JTextField vansTxt = new JTextField(3);
10    JButton carsBtn = new JButton("Car");
11    JButton vansBtn = new JButton("Van");
12    JButton reset = new JButton("Reset");
13    int cars = 0, vans = 0; // counters
14
15    public static void main(String[] args)
16    {
17    new Vehicles();
18    }
19
20    public Vehicles()
21    {
22       setLayout(new BorderLayout());
23       setSize(400, 140);
24       setTitle("Car and van counter");
25       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
26       JPanel top = new JPanel();
27       top.setLayout(new FlowLayout());
28       top.add(new Label("Cars:"));
29       top.add(carsTxt); carsTxt.setEditable(false);
30       top.add(new Label("        Vans:"));
31       top.add(vansTxt); vansTxt.setEditable(false);
32       carsTxt.setText("0");
33       vansTxt.setText("0");
34       add("North", top);
35       JPanel bottom = new JPanel();
36       bottom.setLayout(new FlowLayout());
37       bottom.add(carsBtn);
38       bottom.add(vansBtn);
39       bottom.add(reset);
40       carsBtn.addActionListener(this);
41       vansBtn.addActionListener(this);
42       reset.addActionListener(this);
43       add("South", bottom);
44       setVisible(true);
45    }
46
47    public void actionPerformed(ActionEvent e)
48    {
49       if (e.getSource() == carsBtn)     cars++;
50       else if (e.getSource() == vansBtn) vans++;
51       else if (e.getSource() == reset)   cars = vans = 0;
52       carsTxt.setText("" + cars);
53       vansTxt.setText("" + vans);
54    }
55 }
```

Breaking up the code sections

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

public class Vehicles extends JFrame
        implements ActionListener
{
   JTextField carsTxt = new JTextField(3);
   JTextField vansTxt = new JTextField(3);
   JButton carsBtn = new JButton("Car");
   JButton vansBtn = new JButton("Van");
   JButton reset = new JButton("Reset");
   int cars = 0, vans = 0; // counters

   public static void main(String[] args)
   {
   new Vehicles();
   }
```

the usual GUI details of buttons and text fields

two variables to hold the values for the number of car and van 'clicks'

creating the class

```
public Vehicles()
{
   setLayout(new BorderLayout());
   setSize(400, 140);
   setTitle("Car and van counter");
   setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   JPanel top = new JPanel();
   top.setLayout(new FlowLayout());
   top.add(new Label("Cars:"));
   top.add(carsTxt); carsTxt.setEditable(false);
   top.add(new Label("        Vans:"));
   top.add(vansTxt); vansTxt.setEditable(false);
   carsTxt.setText("0");
   vansTxt.setText("0");
   add("North", top);
   JPanel bottom = new JPanel();
   bottom.setLayout(new FlowLayout());
   bottom.add(carsBtn);
   bottom.add(vansBtn);
   bottom.add(reset);
   carsBtn.addActionListener(this);
   vansBtn.addActionListener(this);
   reset.addActionListener(this);
   add("South", bottom);
   setVisible(true);
}
```

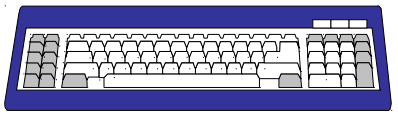
class constructor for Vehicles setting up the content pane adding the GUI features and the listeners to the three buttons

Note the quite complex arrangements of layout to ensure the buttons, labels and text fields end up looking good

Using non-text input and output

- Quite often we have to change text into number before we do the calculation part
- Often we have to change number back to text to display answers
- Here in the Vehicles example we can see one way of getting round the last problem

Vehicles.java



If the event was the car button the car variable is incremented by 1 (cars = cars+1)

```
public void actionPerformed(Act      Event e)
{
   if (e.getSource() == carsBtn)     cars++;
   else if (e.getSource() == vansBtn) vans++;
   else if (e.getSource() == reset)   cars = vans = 0;
   carsTxt.setText("" + cars);
   vansTxt.setText("" + vans);
}
```

If the event was reset car button the car and van variables are both set to 0

this construction allows you to tack a number onto an empty String and everything will then be treated as Strings using toString() methods implicitly

Alternatively here in these 2 options the conversion of the integer to string is stated explicitly

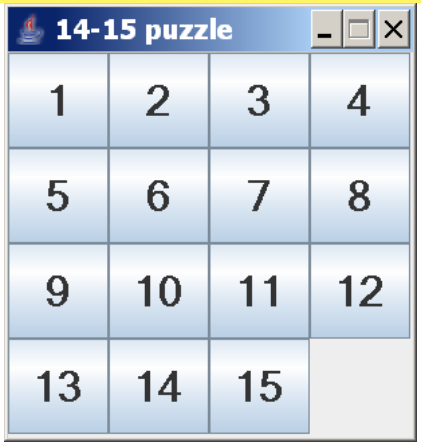
```
public void actionPerformed(ActionEvent e)
{
   if (e.getSource() == carsBtn)     cars++;
   else if (e.getSource() == vansBtn) vans++;
   else if (e.getSource() == reset)   cars = vans = 0;
   carsTxt.setText(String.valueOf(cars));
   vansTxt.setText(String.valueOf(vans));
}
```

```
public void actionPerformed(ActionEvent e)
{
   if (e.getSource() == carsBtn)     cars++;
   else if (e.getSource() == vansBtn) vans++;
   else if (e.getSource() == reset)   cars = vans = 0;
   carsTxt.setText(Integer.toString(cars));
   vansTxt.setText(Integer.toString(vans));
}
```

- Remember if any calculation is to be done you will have to be making sure the Strings you read in are changed to number using some method of the number class like
  - Double.parseDouble(String)
  - Integer.parseInt(String)
- If you are to display numbers in text they must be converted to Strings using an implicit conversion
  - "" + value
- Or one of the methods of the String class
  - String.valueOf(int)
- Or you can use a method of the number class such as
  - Integer.toString(int)

Examples from the notes

- Other examples show the variety of GUIs you can design. Learn by studying these.

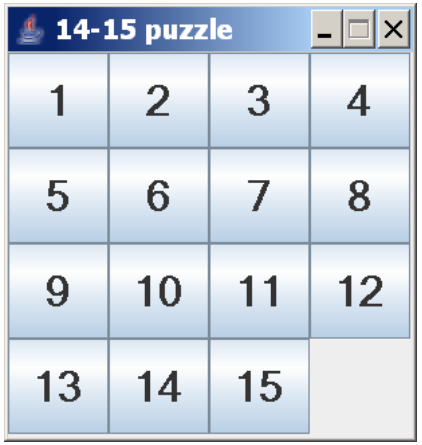


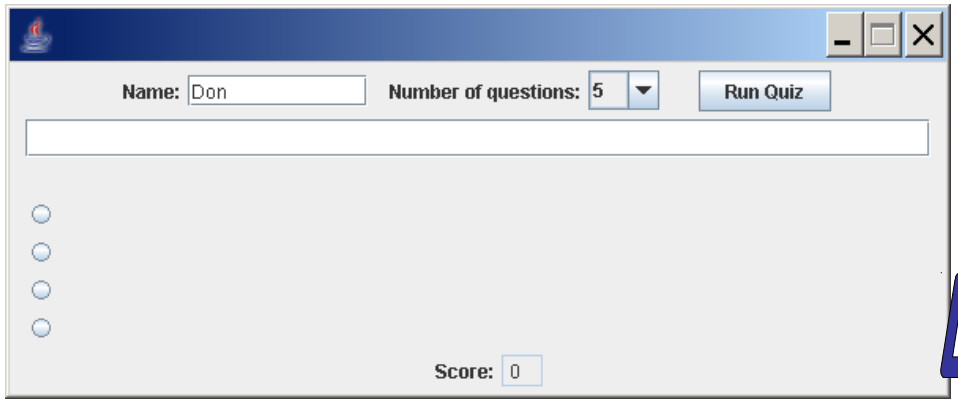

```
Puzzle() {
   setLayout(new GridLayout(4, 4));
   setSize(308, 327); // makes each tile 75 X 75 pixels
   setTitle("14-15 puzzle");
   setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   setResizable(false);
   setVisible(true);
```

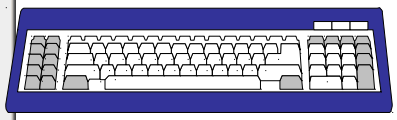
- Puzzle uses 2 classes and is quite complex but is a good example of a grid layout

Examples from the notes

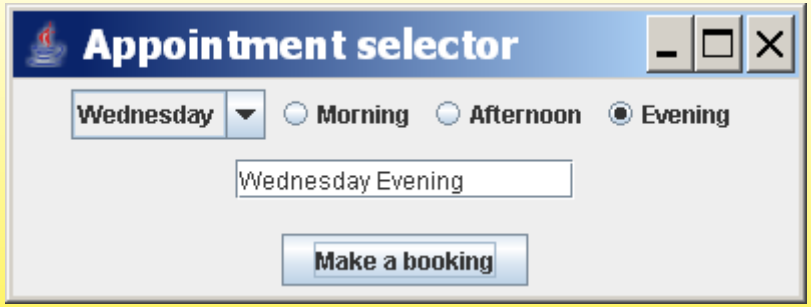
- HelloAge combines many of the GUI aspects we have seen and some and if statements with logic
- Study it, run it and see if you can explain the code
- Other examples show the variety of GUIs you can design. Learn by studying these



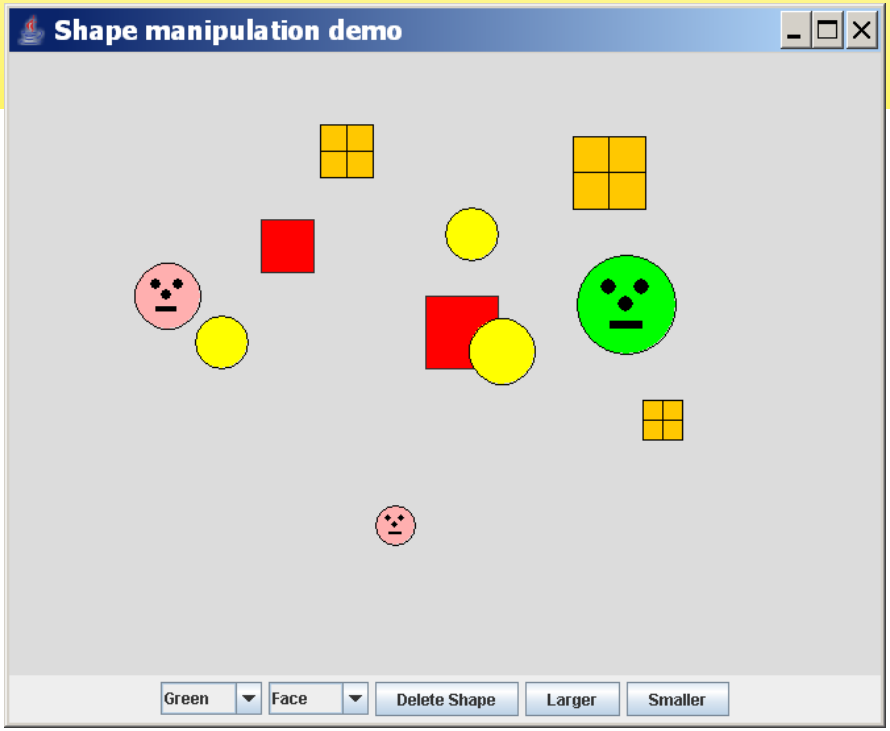




HelloAge.java
Appointment.java


Appointment selector
Wednesday
Morning
Afternoon
Evening
Wednesday Evening
Make a booking


Car and van counter
Cars:
45
Vans:
21
Car
Van
Reset


Shape manipulation demo
Green
Face
Delete Shape
Larger
Smaller

Summary

- This section has been a very brief introduction to GUI programming using Swing. We investigated the following Swing controls:
- JFrames – with a useful EditPlus template file
- JButtons and the ActionListener technique for event handling
- JLabels, JTextFields and JTextAreas for text handling
- JCheckBoxes, JRadioButtons + ButtonGroups and JComboBoxes for making choices with getSource and isSelected methods

```
JButton myButton = new JButton("Press");
```

```
JTextField myText = new JTextField(10);
```

```
JTextArea myArea = new JTextArea(3,20);
```

```
JLabel myLbl = new JLabel("Name:");
```

labels are often just created when needed as
add(new JLabel("Name:");

```
JCheckBox myChBx = new JCheckBox("Try checking it");
```

```
JRadioButton myBtn = new JRadioButton("Male", false);
```

```
ButtonGroup gender = new ButtonGroup();
```

Summary

- We saw that all button click events are handled by a single ActionPerformed method. This has an ActionEvent parameter that can be tested to discover which button (or other object) caused the event to happen
- We also saw that there are layout managers to control the appearance of the various controls on a form – flow layout, border layout, grid layout.
- There are many other types of control, layout managers and event listeners in Java you can investigate but these are the fundamental ones