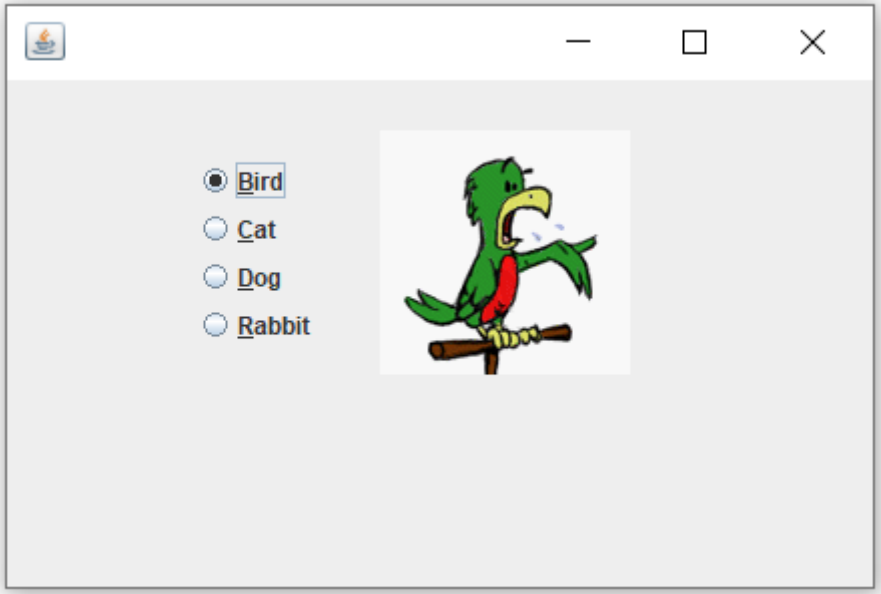
Event Handling

Placing Radio Buttons

Placing a Radio buttons

- A **JRadioButton** object is a GUI component that represents a Radio button.
- Here's an example of radio buttons.



Steps to place radio buttons

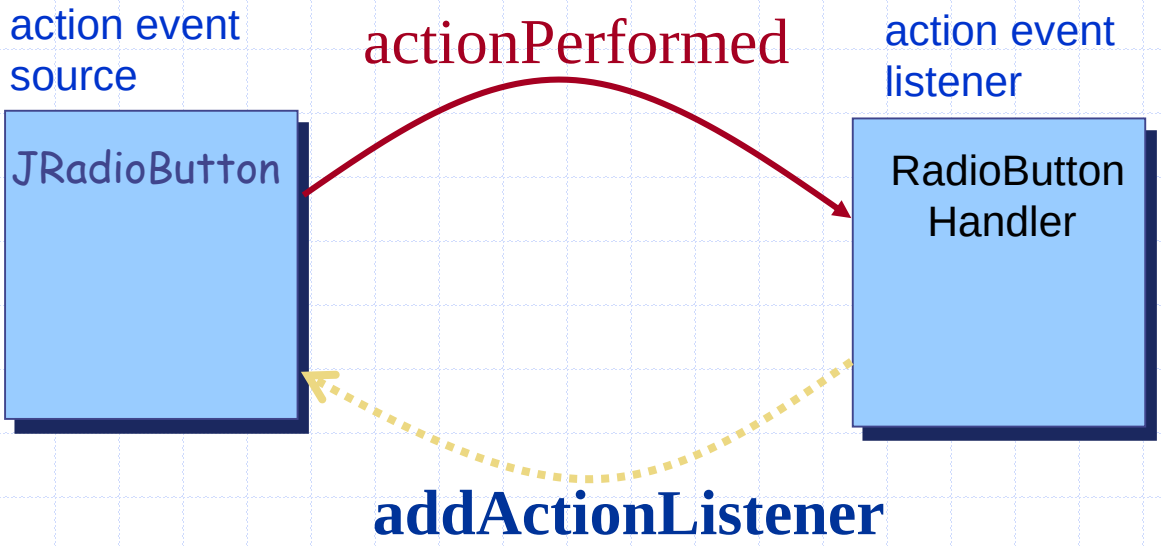
- Declare radio-buttons
  - **private** JRadioButton birdButton, catButton;
- Create the radio-buttons.
  - birdButton = **new** JRadioButton(*birdString*);
  - catButton = **new** JRadioButton(*catString*);
- Group the radio buttons

Steps to place radio buttons

- Group the radio buttons
  - ButtonGroup group = **new** ButtonGroup();
  - group.add(birdButton);
  - group.add(catButton);

- Register a listener for the radio buttons.
  - birdButton.addActionListener(**this**);
  - catButton.addActionListener(**this**);

Handling Action Events




```
JRadioButton button = new JRadioButton("OK");
ButtonHandler handler = new ButtonHandler( );

button.addActionListener(handler);
```

ActionListener Interface

- When we call the **addActionListener** method of an event source, we must pass an instance of a class that implements the **ActionListener** interface.
- The ActionListener interface includes one method named **actionPerformed**.
- A class that implements the ActionListener interface must therefore provide the method body of **actionPerformed**.
- Since actionPerformed is the method that will be called when an action event is generated, this is the place where we put a code we want to be executed in response to the generated events.

The ButtonHandler Class

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

class ButtonHandler implements ActionListener {
    . . .
    public void actionPerformed(ActionEvent event) {
                // do the action                    }
}
```