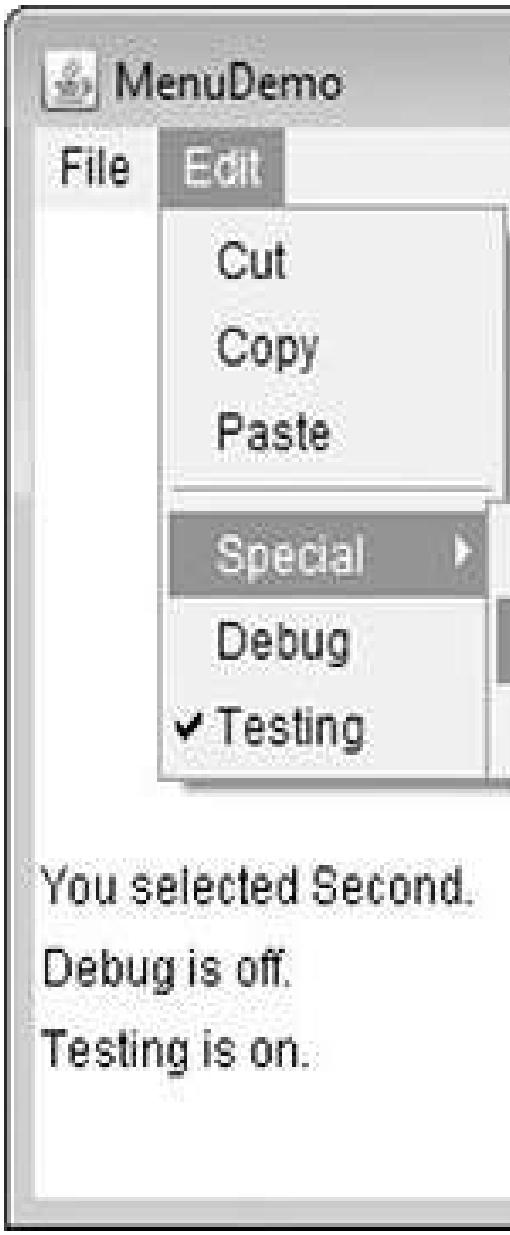
MENU

```
// Illustrate menus.
import java.awt.*;
import java.awt.event.*;

class MenuDemo extends Frame {
  String msg = "";
  CheckboxMenuItem debug, test;

  public MenuDemo() {

    // Create menu bar and add it to frame.
    MenuBar mbar = new MenuBar();
    setMenuBar(mbar);

    // Create the menu items.
    Menu file = new Menu("File");
    MenuItem item1, item2, item3, item4, item5;
    file.add(item1 = new MenuItem("New..."));
    file.add(item2 = new MenuItem("Open..."));
    file.add(item3 = new MenuItem("Close"));
    file.add(item4 = new MenuItem("-"));
    file.add(item5 = new MenuItem("Quit..."));
    mbar.add(file);

    Menu edit = new Menu("Edit");
    MenuItem item6, item7, item8, item9;
    edit.add(item6 = new MenuItem("Cut"));
    edit.add(item7 = new MenuItem("Copy"));
    edit.add(item8 = new MenuItem("Paste"));
    edit.add(item9 = new MenuItem("-"));

    Menu sub = new Menu("Special");
    MenuItem item10, item11, item12;
    sub.add(item10 = new MenuItem("First"));
    sub.add(item11 = new MenuItem("Second"));
    sub.add(item12 = new MenuItem("Third"));
    edit.add(sub);
```



```
    // These are checkable menu items.
    debug = new CheckboxMenuItem("Debug");
    edit.add(debug);
    test = new CheckboxMenuItem("Testing");
    edit.add(test);

    mbar.add(edit);

    // Create an object to handle action and item events.
    MyMenuHandler handler = new MyMenuHandler();

    // Register to receive those events.
    item1.addActionListener(handler);
    item2.addActionListener(handler);
    item3.addActionListener(handler);
    item4.addActionListener(handler);
    item6.addActionListener(handler);
    item7.addActionListener(handler);
    item8.addActionListener(handler);
    item9.addActionListener(handler);
    item10.addActionListener(handler);
    item11.addActionListener(handler);
    item12.addActionListener(handler);
    debug.addItemListener(handler);
    test.addItemListener(handler);

    // Use a lambda expression to handle the Quit selection.
    item5.addActionListener((ae) -> System.exit(0));

    addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent we) {
        System.exit(0);
      }
    });
  }
```

```
public void paint(Graphics g) {
  g.drawString(msg, 10, 220);

 if(debug.getState())
    g.drawString("Debug is on.", 10, 240);
  else
    g.drawString("Debug is off.", 10, 240);

  if(test.getState())
    g.drawString("Testing is on.", 10, 260);
  else
    g.drawString("Testing is off.", 10, 260);
}

public static void main(String[] args) {
  MenuDemo appwin = new MenuDemo();

  appwin.setSize(new Dimension(250, 300));
  appwin.setTitle("MenuDemo");
  appwin.setVisible(true);
}

// An inner class for handling action and item events
// for the menu.
class MyMenuHandler implements ActionListener, ItemListener
```

```
    // Handle action events.
    public void actionPerformed(ActionEvent ae) {
        msg = "You selected ";
        String arg = ae.getActionCommand();

        if(arg.equals("New..."))
          msg += "New.";
        else if(arg.equals("Open..."))
          msg += "Open.";
        else if(arg.equals("Close"))
          msg += "Close.";
        else if(arg.equals("Edit"))
          msg += "Edit.";
        else if(arg.equals("Cut"))
          msg += "Cut.";
        else if(arg.equals("Copy"))
          msg += "Copy.";
        else if(arg.equals("Paste"))
          msg += "Paste.";
        else if(arg.equals("First"))
          msg += "First.";
        else if(arg.equals("Second"))
          msg += "Second.";
        else if(arg.equals("Third"))
          msg += "Third.";
        else if(arg.equals("Debug"))
          msg += "Debug.";
        else if(arg.equals("Testing"))
          msg += "Testing.";

        repaint();
    }

    // Handle item events.
    public void itemStateChanged(ItemEvent ie) {
      repaint();
    }
  }
}
```