Java OOP: Panels, Labels, Text Fields, and Buttons*

Richard Baldwin

This work is produced by The Connexions Project and licensed under the
Creative Commons Attribution License †

Abstract

Learn how to create and service a graphical user interface containing panels, labels, text fields, and buttons.

1 Table of Contents

- Preface (p. 1)
  - Viewing tip (p. 1)
    - Figures (p. 2)
    - Listings (p. 2)
- Preview (p. 2)
- Discussion and sample code (p. 6)
- Run the program (p. 10)
- Summary (p. 10)
- What's next? (p. 10)
- Miscellaneous (p. 10)
- Complete program listing (p. 11)

2 Preface

This module is one of a series of modules designed to teach you about Object-Oriented Programming (OOP) using Java.

The program described in this module requires the use of the Guzdial-Ericson multimedia class library. You will find download, installation, and usage instructions for the library at Java OOP: The Guzdial-Ericson Multimedia Class Library [1] .

2.1 Viewing tip

I recommend that you open another copy of this document in a separate browser window and use the following links to easily find and view the figures and listings while you are reading about them.

*Version 1.2: Oct 1, 2012 12:35 pm -0500

†http://creativecommons.org/licenses/by/3.0/

[1] http://cnx.org/content/m44148/latest/

2.1.1 Figures

- Figure 1 (p. 3) . Program output at startup.
- Figure 2 (p. 4) . Program output after one click.
- Figure 3 (p. 5) . Program output after several clicks.

2.1.2 Listings

- Listing 1 (p. 6) . Beginning of the class named Prob05Runner.
- Listing 2 (p. 6) . Instantiate GUI components.
- Listing 3 (p. 7) . Beginning of the constructor.
- Listing 4 (p. 7) . Add the six GUI components to the panel.
- Listing 5 (p. 8) . Get the frame and add the panel to the frame.
- Listing 6 (p. 8) . Set the background to blue and add a turtle.
- Listing 7 (p. 9) . Define, instantiate, and register a listener on the Move button.
- Listing 8 (p. 9) . Action listener to terminate the program.
- Listing 9 (p. 11) . Source code for Prob05.

3 Preview

In this module, you will learn how to create and service a graphical user interface containing panels, labels, text fields, and buttons.

Just as you did in earlier modules, you will modify the **World** class to make it possible to get access to the **JFrame** object that is encapsulated in a **World** object. However, because I explained those modifications to the **World** class in earlier modules, I won't repeat the explanation here. You can find a modified version of the World class in Java OOP: Background Color, Text Color, Mouse Clicks, etc. [2]

Program specifications

Write a program named **Prob05** that uses the class definition named **Prob05** shown in Listing 9 (p. 11) along with Ericson's media library to produce the graphic output images shown in Figure 1 (p. 3) , Figure 2 (p. 4) , and Figure 3 (p. 5) .

Program output at startup

Figure 1 (p. 3) shows the image that appears on the screen when the program first starts running.

[2] http://cnx.org/content/m44351/latest/#Listing_10

Program output at startup.

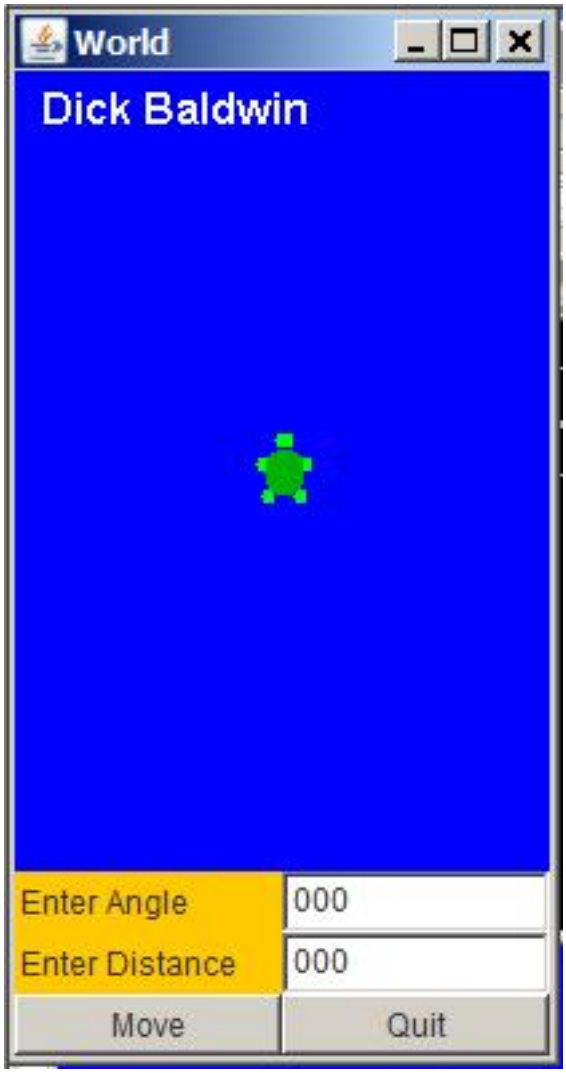

Figure 1: Program output at startup.

Program output after one click

Figure 2 (p. 4) shows the output image after the user enters the values shown into the two text fields and clicks the **Move** button once.

Program output after one click.

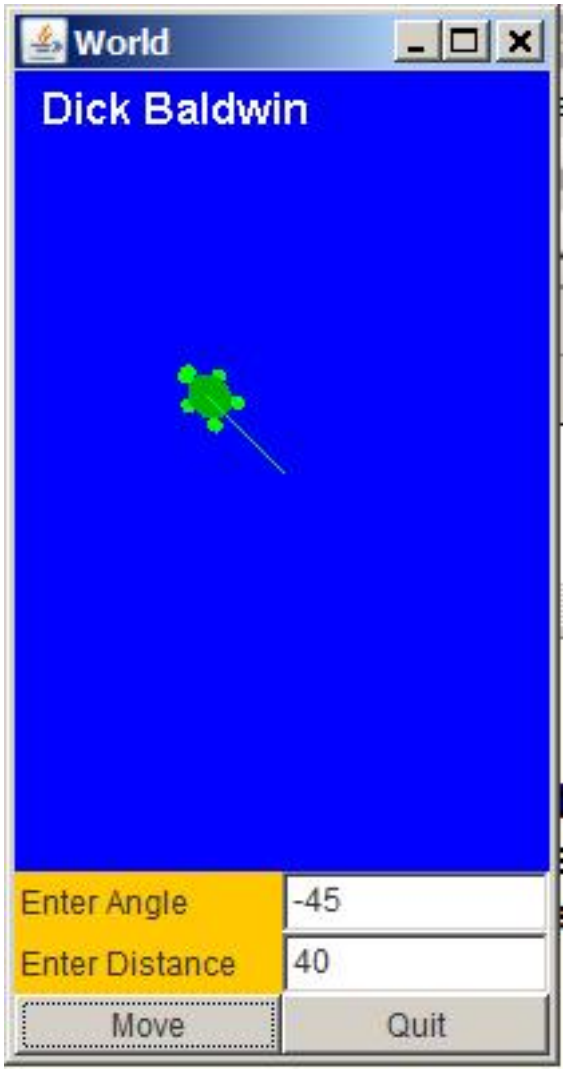

Figure 2: Program output after one click.

Program output after several clicks

Figure 3 (p. 5) shows the output image after the user enters several different sets of values into the text fields and clicks the **Move** button several times.

Program output after several clicks.

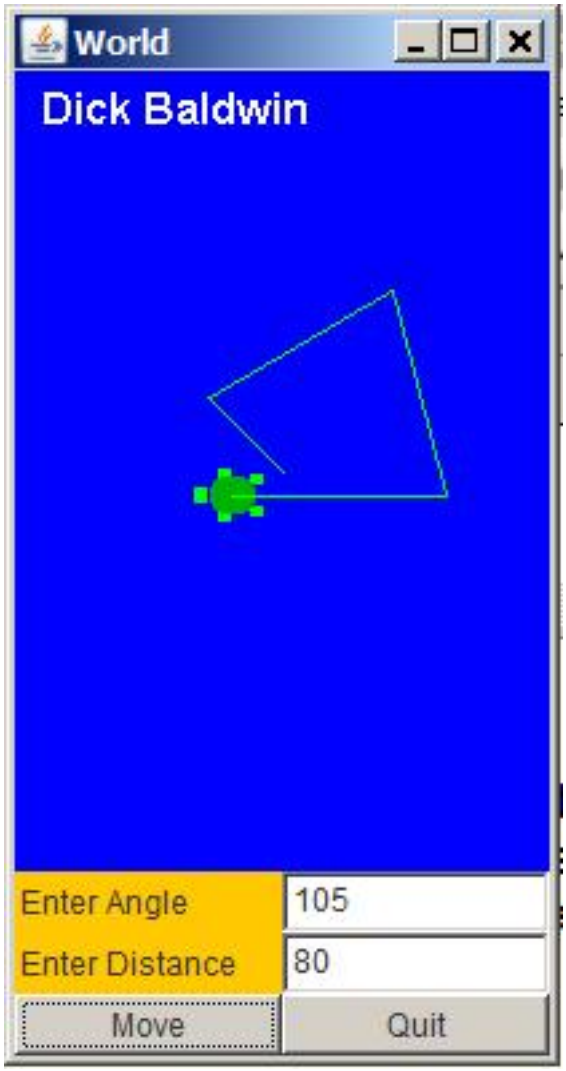

Figure 3: Program output after several clicks.

Labels, text fields, and buttons

As shown in Figure 1 (p. 3) , this program adds two buttons, two labels, and two text fields to form a GUI at the bottom of the **World** object. Although it isn't obvious, those six components are contained in a **Panel** object.

Each time you enter numeric values into the angle and distance fields and then click the **Move** button, the turtle will turn by that angle in degrees and move by that distance in pixels.

Program termination

The program terminates and returns control to the operating system when the user clicks the **Quit**

button.

Use AWT components instead of Swing components

The GUI at the bottom of the **World** object is comprised of ***AWT*** components instead of ***Swing*** components.

Required text output

In addition to the output image described above, your program must produce some rather inconsequential text output on the command- line screen.

4 Discussion and sample code

The class named Prob05

You can view the driver class named **Prob05** at the beginning of the source code in Listing 9 (p. 11) . You are already familiar with the code in the **main** method of that class from earlier modules so I won't waste time explaining it.

Briefly, the **main** method instantiates a new object of the class named **Prob05Runner** . Once that object is instantiated, the program goes idle waiting for an event to happen. Events happen when the user enters text into the text fields or presses one of the buttons shown in Figure 1 (p. 3) .

The class named Prob05Runner

I will explain this program in fragments. A complete listing of the program is provided in Listing 9 (p. 11) near the end of the module

The class named **Prob05Runner** begins in Listing 1 (p. 6) .

Listing 1: Beginning of the class named Prob05Runner.

```
 class Prob05Runner{
Turtle turtle = null;
Picture picture = null;
World world = new World(200,300);
```

There's nothing new in Listing 1 (p. 6) . You have seen code like this in several earlier modules.

Instantiate GUI components

Listing 2 (p. 6) instantiates the GUI components that are used to construct the GUI at the bottom of Figure 1 (p. 3) .

Listing 2: Instantiate GUI components.

```
   Panel mainPanel = new Panel();
Label angleLabel = new Label("Enter Angle");
TextField angleField = new TextField("000");
Label distanceLabel = new Label("Enter Distance");
TextField distanceField = new TextField("000");
Button moveButton = new Button("Move");
Button quitButton = new Button("Quit");

int angle = 0;
int distance = 0;
```

References to the GUI components are saved in instance variables with descriptive names.

GUI component initialization

Except for the **Panel** object, each GUI component is initialized with information that is appropriate to the type of component. Compare the initialization values in Listing 2 (p. 6) with the image in Figure 1 (p. 3) for a better understanding of what I mean by this.

Brief description of the GUI components

You are encouraged to visit the Sun Java documentation and read about the following AWT components:

- **Panel** : A panel provides space in which an application can attach any other components, including other panels.
- **Label** : A Label object is a component for placing text in a container. A label displays a single line of read-only text. The text can be changed by the application, but a user cannot edit it directly.
- **TextField** : A text component that allows for the editing of a single line of text.
- **Button** : This class creates a labeled button. The application can cause some action to happen when the button is pushed.

Figure 1 (p. 3) contains a **Panel** object at the bottom. The panel contains two **Label** objects *(shown as orange)* , two **TextField** objects *(white)* , and two **Button** objects *(gray)* . Those seven objects are instantiated in Listing 2 (p. 6) .

The angle and distance variables

Listing 2 (p. 6) also declares and initializes two instance variables named **angle** and **distance** . *(Note that these two variables would be automatically initialized to zero if I didn't initialize them, but I prefer to initialize them explicitly in order to make the code more self-documenting.)*

Beginning of the constructor

The constructor for the class named **Prob05Runner** begins in Listing 3 (p. 7) .

Listing 3: Beginning of the constructor.

```
  public Prob05Runner(){
System.out.println("Dick Baldwin");

mainPanel.setBackground(Color.ORANGE);

mainPanel.setLayout(new GridLayout(0,2));
```

The constructor begins by displaying my name on the command line screen. This is inconsequential insofar as the overall operation of the program is concerned.

Set the panel background color to orange

Then Listing 3 (p. 7) sets the background color of the panel to orange. This is what causes the two labels in Figure 1 (p. 3) to appear to be orange. They are actually transparent except for the text. The orange color shows through causing the labels to appear to be orange.

Set the layout to GridLayout

Setting the layout manager controls how the components that are placed in the panel will be arranged. A **GridLayout** causes all components to be the same size arranged in rows and columns.

Overloaded constructors

There are several overloaded versions of the **GridLayout** constructor. For the constructor used in Listing 3 (p. 7) , the parameters specify the number of rows and the number of columns in that order. Specifying the number of rows as 0 and the number of columns as 2 means that the layout manager will accept any number of rows but only two columns.

The order of component placement

The intersections of the rows and columns create ***cells*** . Components are placed in the cells in left to right, top to bottom order as they are added to the panel.

Add the six GUI components to the panel

Listing 4 (p. 7) adds the six GUI components to the panel in the left to right, top to bottom order described above.

Listing 4: Add the six GUI components to the panel.

```
    mainPanel.add(angleLabel);
mainPanel.add(angleField);
mainPanel.add(distanceLabel);
mainPanel.add(distanceField);
mainPanel.add(moveButton);
mainPanel.add(quitButton);
```

At this point, the panel has been populated with GUI components, but the panel itself has not been added to the **JFrame** object that forms the **World** object. That is accomplished in Listing 5 (p. 8) .

Get the frame and add the panel to the frame

Listing 5 (p. 8) gets a reference to the **World** object's frame and adds the panel to the SOUTH location in that frame.

Listing 5: Get the frame and add the panel to the frame.

```
    JFrame frame = world.getFrame();
frame.getContentPane().add(
                        mainPanel,BorderLayout.SOUTH);

frame.pack();
```

You can surmise from the word SOUTH that the panel is added to the bottom of the frame. To learn more about this, visit the **BorderLayout** class in Sun's Java documentation.

Pack the frame

As you learned in an earlier module, it is very important that you pack the frame at this point. Packing the frame causes the frame to adjust its dimensions in order to accommodate all of the components that have been added to it. In this case, the frame contains a **Picture** object *(placed there when the **World** was constructed)* and a **Panel** object placed there in Listing 5 (p. 8) .

What about the size of the panel?

Exactly how the panel decides what size it needs to be to accommodate the six GUI components is a fairly complicated topic, so I won't go into it here. However, if you do much work developing GUIs, you definitely need to understand the process. I have explained the automatic sizing process in several tutorials on my website.

Set the background to blue and add a turtle

Listing 6 (p. 8) sets the background color of the world to blue and adds a turtle to the world.

Listing 6: Set the background to blue and add a turtle.

```
    //Initialize the picture.
picture = world.getPicture();
//Set picture background to BLUE
picture.setAllPixelsToAColor(Color.BLUE);
//Display the student's name on the picture.
picture.addMessage("Dick Baldwin",10,20);
//Add a turtle to the world. This causes the
// world to be repainted.
turtle = new Turtle(world);
```

There is nothing new in Listing 6 (p. 8) . I have explained code very similar to this code in earlier modules.

Could stop at this point

If we were to stop programming at this point, the program would be executable, and would produce the output shown in Figure 1 (p. 3) when it is run. However it would be completely passive. By that, I mean

that entering values into the text fields and clicking the buttons at the bottom of Figure 1 (p. 3) would have no effect. However, the buttons would appear to be active from a visual viewpoint because the animation behavior is built into objects of the **Button** class.

Register listener objects

In order to cause the buttons to impact the behavior of the program, we must instantiate and register listener object on the buttons. I will do that using anonymous classes.

Define, instantiate, and register a listener on the Move button

The code in Listing 7 (p. 9) :

- Defines an anonymous **ActionListener** class.
- Instantiates an anonymous object of that class.
- Registers that object on the **Move** button

Listing 7: Define, instantiate, and register a listener on the Move button.

```
    moveButton.addActionListener(
  new ActionListener(){

    public void actionPerformed(ActionEvent e){
      angle = Integer.parseInt(angleField.getText());
      distance = Integer.parseInt(
                            distanceField.getText());
      turtle.turn(angle);
      turtle.forward(distance);
    }//end action performed

  }//end newActionListener
);//end addActionListener
```

The actionPerformed method

As you learned in an earlier module, the code in the **actionPerformed** method is executed each time the user presses the **Move** button.

The **actionPerformed** method in Listing 7 (p. 9) begins by getting the text from each of the text fields, converting the text to type **int** , and saving the **int** values in the variables named **angle** and **distance** .

Then Listing 7 (p. 9) calls the **turn** and **forward** methods of the **Turtle** class to cause the turtle to turn by the specified angle and then move forward by the specified distance.

Action listener to terminate the program

Listing 8 (p. 9) registers an action listener on the **Quit** button, which will cause the program to terminate when the user clicks the button.

Listing 8: Action listener to terminate the program.

```
    quitButton.addActionListener(
  new ActionListener(){
    public void actionPerformed(ActionEvent e){

      System.exit(0);

    }//end action performed
  }//end newActionListener
```

```
    );//end addActionListener

  }//end constructor
  //-----------------------------------------------------//

}//end class Prob05Runner
```

As you can see in Listing 8 (p. 9) , this code causes the **exit** method of the **System** class to be called when the user clicks the **Quit** button. According to the Sun documentation, a call to the **exit** method *"Terminates the currently running Java Virtual Machine."*

The end of the constructor and the end of the class

Listing 8 (p. 9) also shows the end of the constructor for the **Prob05Runner** class and the end of the class.

5 Run the program

I encourage you to copy the code from Listing 9 (p. 11) . Compile the code and execute it. Experiment with the code, making changes and observing the results of your changes. Make certain that you can explain why your changes behave as they do.

6 Summary

In this module, you learned how to create and service a graphical user interface containing panels, labels, text fields, and buttons.

7 What's next?

In the next module, you will learn about:

- Alpha transparency
- A buffered image of type TYPE_INT_ARGB
- The ability to use the **getBasicPixel** and **setBasicPixel** methods,
- The use of the bitwise AND and OR operators,
- The use of the drawImage method of the **Graphics** class.

8 Miscellaneous

This section contains a variety of miscellaneous information.

NOTE: **Housekeeping material**

- Module name: Java OOP: Panels, Labels, Text Fields, and Buttons
- File: Java3110.htm
- Revised: 08/20/12

NOTE: **Disclaimers: Financial** : Although the Connexions site makes it possible for you to download a PDF file for this module at no charge, and also makes it possible for you to purchase a pre-printed version of the PDF file, you should be aware that some of the HTML elements in this module may not translate well into PDF.

I also want you to know that, I receive no financial compensation from the Connexions website even if you purchase the PDF version of the module.

In the past, unknown individuals have copied my modules from cnx.org, converted them to Kindle books, and placed them for sale on Amazon.com showing me as the author. I neither receive compensation for those sales nor do I know who does receive compensation. If you purchase such a book, please be aware that it is a copy of a module that is freely available on cnx.org and that it was made and published without my prior knowledge.

Affiliation : I am a professor of Computer Information Technology at Austin Community College in Austin, TX.

9 Complete program listing

A complete listing of the program discussed in this lesson is shown in Listing 9 (p. 11) below.

Listing 9: Source code for Prob05.

```
    /*File Prob05 Copyright 2008 R.G.Baldwin
 *Revised 12/31/08
*******************************************************/
import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.Panel;
import java.awt.TextField;
import java.awt.Label;
import java.awt.Button;

import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.Color;

import javax.swing.JFrame;

public class Prob05{
  //DO NOT MODIFY THE CODE IN THIS CLASS DEFINITION.
  public static void main(String[] args){
    new Prob05Runner();
  }//end main method
}//end class Prob05
//End program specifications.
/*-----------------------------------------------------//

*******************************************************/

class Prob05Runner{
  Turtle turtle = null;
  Picture picture = null;
  World world = new World(200,300);
  Panel mainPanel = new Panel();
  Label angleLabel = new Label("Enter Angle");
  TextField angleField = new TextField("000");
  Label distanceLabel = new Label("Enter Distance");
  TextField distanceField = new TextField("000");
```

```
  Button moveButton = new Button("Move");
  Button quitButton = new Button("Quit");
  int angle = 0;
  int distance = 0;

  public Prob05Runner(){
    System.out.println("Dick Baldwin");

    //Construct the GUI.
    mainPanel.setBackground(Color.ORANGE);
    mainPanel.setLayout(new GridLayout(0,2));
    mainPanel.add(angleLabel);
    mainPanel.add(angleField);
    mainPanel.add(distanceLabel);
    mainPanel.add(distanceField);
    mainPanel.add(moveButton);
    mainPanel.add(quitButton);

    //Get a reference to the world frame and add the GUI
    // to the frame.
    JFrame frame = world.getFrame();
    frame.getContentPane().add(
                           mainPanel,BorderLayout.SOUTH);


    frame.pack();

    //Initialize the picture.
    picture = world.getPicture();
    //Set picture background to BLUE
    picture.setAllPixelsToAColor(Color.BLUE);
    //Display the student's name on the picture.
    picture.addMessage("Dick Baldwin",10,20);
    //Add a turtle to the world. This causes the
    // world to be repainted.
    turtle = new Turtle(world);

    //-----------------------------------------------------//
    //Register anonymous listeners on the two buttons.
    moveButton.addActionListener(
      new ActionListener(){
        public void actionPerformed(ActionEvent e){
          angle = Integer.parseInt(angleField.getText());
          distance = Integer.parseInt(
                                 distanceField.getText());
          turtle.turn(angle);
          turtle.forward(distance);
        }//end action performed
      }//end newActionListener
    );//end addActionListener
```

```
    quitButton.addActionListener(
      new ActionListener(){
        public void actionPerformed(ActionEvent e){
          System.exit(0);
        }//end action performed
      }//end newActionListener
    );//end addActionListener

  }//end constructor
  //---------------------------------------------------//

}//end class Prob05Runner
```

-end-