Quiz: Layout Managers

Back

Objectives

- Gain familiarity with layout managers.

Instructions

All of these exercises for the lab are based on the Layout.java class. The lab exercise is broken into a series of different portions or "cases". Each case consists as a set of method calls for a Layout instance; they are given as an application method. You can either add an application method to the Layout class, or make these method calls interactively within your chosen IDE. In each case, you need to answer the questions posed for that case. Note: the cases are independent. You will submit the answers for the questions in cases 1 through 9 in the Moodle quiz. We will discuss the "Challenge Questions" in class next week.

Case 1

Consider the following application method for Layout.

```
public static void main(String[] args)
{
  Layout theApp = new Layout();
}
```

Answer the following questions:

a. How many JButton instances exist in the application?
b. How many JButton instances are shown?
c. What happens as the user resizes the window?

Case 2

Consider the following application method for Layout.

```
public static void main(String[] args)
{
  Layout theApp = new Layout();
  theApp.useFlowLayout();
}
```

Answer the following questions:

a. What does the window look like? (Refer to images page.)
b. Are the buttons all the same size?
c. What determines the size of the buttons?
d. What happens as the user resizes the window, making it considerably narrower? Wider?

Case 3

Consider the following application method for Layout.

```
public static void main(String[] args)
{
  Layout theApp = new Layout();
  theApp.useFlowLayout();
  theApp.pack();
}
```

Answer the following questions:

a. What does the window look like? (Refer to images page.)
b. What does the **pack** method do?

Case 4

Consider the following application method for Layout.

```
public static void main(String[] args)
{
  Layout theApp = new Layout();
  theApp.useGridLayout();
}
```

Answer the following questions:

a. What does the window look like? (Refer to images page.)
b. What determines the size of the buttons?
c. What happens as the user resizes the window?

Case 5

Consider the following application method for Layout.

```
public static void main(String[] args)
{
  Layout theApp = new Layout();
  theApp.useBorderLayout();
}
```

Answer the following questions:

a. What does the window look like? (Refer to images page.)

b. What happens as the user resizes the window?

c. What determines the size of the buttons?

Hint: What happens if you add the call theApp.pack() to the end of main?

Case 6

Consider the following application method for Layout.

```
public static void main(String[] args)
{
  Layout theApp = new Layout();
  theApp.useToolbar();
}
```

Answer the following questions:

a. What does the window look like? (Refer to images page.)
b. When the window first appears, how many buttons are visible? Explain.

Hint: What happens if the user resizes the window?

Case 7

Consider the following application method for Layout.

```
public static void main(String[] args)
{
  Layout theApp = new Layout();
  theApp.useToolbarII();
}
```

Answer the following questions:

a. What does the window look like? (Refer to images page.)
b. Why is "Ernie ..." now visible?

Hint: Read the documentation for setPreferredSize. This method is inherited from JComponent.

Case 8

Consider the following application method for Layout.

```
public static void main(String[] args)
{
  Layout theApp = new Layout();
  theApp.useCustomLayout();
}
```

Answer the following questions:

a. What does the window look like? (Refer to images page.)
b. What happens as the user resizes the window?
c. What happens when c.validate() is called at the end of the useCustomLayout method?

Case 9

Consider the following application method for Layout.

```
public static void main(String[] args)
{
  Layout theApp = new Layout();
```

```
    theApp.useBoxLayout();
  }
```

Answer the following questions:

a. What does the window look like? (Refer to images page.)
b. What happens as the user resizes the window?
c. Briefly describe the difference between BoxLayout (case 9) and FlowLayout (case 6) for the toolbar.

Hint: The javax.swing.Box class provides an easy way to use javax.swing.BoxLayout. For more information, see the API documentation for these two classes.

"Can you figure it out?" Questions

These cases will be discussed in class, but are not part of the quiz *per se*.

Case 10

Consider the following alternate application methods for Layout.

```
public static void main(String[] args)
{
  Layout theApp = new Layout();
  theApp.useToolbarII();
  theApp.useGridLayout();
}

public static void main(String[] args)
{
  Layout theApp = new Layout();
  theApp.useToolbarII();
  theApp.useFlowLayout();
}
```

Answer the following questions:

a. What do the windows look like?
b. The arrangement of buttons is different due to the call to useToolbarII. Explain.

Hint: What happens if a call to resetToolbar is added to either of these application methods?

Case 11

Consider the following application method for Layout.

```
public static void main(String[] args)
{
  Layout theApp = new Layout();
  theApp.useGridLayout();
  theApp.useBorderLayout();
}
```

Answer the following questions:

a. What does the window look like?
b. What happens as the user resizes the window?
c. Briefly explain the behavior of "Charlie".

Case 12

Consider the following application method for Layout.

```
public static void main(String[] args)
{
  Layout theApp = new Layout();
  theApp.useBorderLayout();
  theApp.showAButton();
}
```

Answer the following questions:

a. What buttons appear?
b. What happens as the user moves the mouse over the window?
c. What happens if the user resizes the window?
d. Briefly explain this behavior.

Hint: See Case 11.

Back