# Chapter 17

# GUIs and event-driven programs

## Lesson page 17-1. GUIs and event-driven programming

**Question 1.** GUI stands for “Graphical User Interface”.

**Question 2.** The newer one, the Swing package, provides more flexibility, and the fact that most of its classes are lightweight is an additional advantage. However, it is not fully implemented on all systems, and it is sometimes messy to get it working with applets. If you are on a Windows PC, there is no problem, but the Macintosh may give a few problems. (Mac OS X solves these problems.)

**Question 3.** A lightweight class is one that is written entirely in Java.

**Question 4.** A heavyweight class is one that is written at least partially using native methods —methods that are written in another programming language or in a machine language.

**Question 5.** Anther name for the Swing package is the ”Java Foundation Classes”, or JFC.

**Question 6.** Use a `JFrame`. A `JFrame` has a title and can be resized and dragged around; a `JWindow` doesn’t have these properties.

**Question 7.** Here’s the sequence:

```
JFrame jf= new JFrame("JFrame");
jf.pack();
jf.setVisible(true);
```

## Lesson page 17-2. Components and containers

**Question 1.** Here are the components:

1. `JButton`: a button is a component that can be clicked on (with the mouse) to have some action performed.
2. `JLabel`: a label is a sequence of characters, which can be changed by the program but not by the user.

3. `JTextField`: a text field is an editable line of text; typically, the user can type into it, and the program can monitor the typing or read it all at once, for example, when a button is pressed.

4. `JTextArea`: a text area is like a text field, except that it typically contains more than one line; scroll bars appear automatically when the displayed area is not big enough for the text.

5. `JList`: a list contains a series of items. The user can select one or more of them (depending on how the list is constructed). The list automatically has a scroll bar if all its items can't be displayed at one time.

**Question 2.** The other possible arguments are: `BorderLayout.WEST`, `BorderLayout.NORTH`, `BorderLayout.SOUTH`, and `BorderLayout.CENTER`. See the first footnote on Lesson page 17-3 for a picture that shows what each of these arguments means.

**Question 3.** `Container` is a Java class whose instances are objects that can contain `Components` (e.g. `JButtons`).

**Question 4.** Here are the statements:

```
JPanel jp= new JPanel();
JButton button1= new JButton("first");
JButton button2= new JButton("second");
jp.add(button1);
jp.add(button2);
jf.getContentPane().add(jp, BorderLayout.EAST);
```

## Lesson page 17-3. Layout managers

**Question 1.** A layout manager is an instance of a class that is associated with a container and performs the task of laying out the components in the container.

**Question 2.** We don't draw a picture here; instead, look near the top of the first footnote on Lesson page 17.3 for a picture. A statement that will add a component `jb` to `JFrame jf` is shown below. The <*argument*> can be one of `BorderLayout.EAST`, `BorderLayout.WEST`, `BorderLayout.NORTH`, `BorderLayout.SOUTH`, and `BorderLayout.CENTER`.

```
jf.getContentPane().add(jb,<argument>);
```

**Question 3.** `jp.add(jb);`

**Question 4.** Since the layout manager is a `FlowLayout` manager, the five components appear one after another, in a row. If there is not enough room for them all in one row, then a second row is used, and so on. If the panel is resized and made wider, then more of them will be placed in the first row; if it is resized and made narrower, then fewer of them appear in each row.

**Question 5.** `jf.getContentPane().setLayout(`**new** `FlowLayout());`

## Lesson page 17-4. Listening to a GUI

**Question 1.** The four underlined items are: `ActionListener`; `actionPerformed`; `ActionEvent`; `jb.addActionListener(this);`.

**Question 2.** We show only method `actionPerformed`. The only other thing to do is to add the statement `bw.setText("");` after the statement `bw.setEnabled(`**false**`);`.

```
public void actionPerformed(ActionEvent e) {
    boolean b= (be.isEnabled());
    be.setEnabled(!b);
    bw.setEnabled(b);
    if (be.isEnabled()) {
        be.setText("first");
        bw.setText("");
    }
    else {
        be.setText("");
        bw.setText("second");
    }
}
```