1.00/1.001/1.002
Introduction to Computers and Engineering Problem Solving

Recitation 7
Swing

Frame Customization

Events

April 2nd, 3rd 2012

Creating a Frame: Exercise

In a **`main()`** method create and display a 200x100 frame, that resembles the following frame.


```
public static void main(String[] args)
{
    JFrame fr = new JFrame("Empty Frame");
    fr.setSize(200,100);
    fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    fr.setVisible(true);
}
```

There are often multiple ways of doing the same thing with Swing. For example:

```
JFrame fr = new JFrame("Empty Frame");
```

equivalent

```
JFrame fr = new JFrame();
fr.setTitle("Empty Frame");
```

Frame Customization

`CustomFrame` is a custom `JFrame` class

- `CustomFrame` **`extends`** `JFrame`
- Sets the title to “Custom Frame”
- Sets the size to 200 X 200
- Sets the background to blue

```
public class CustomFrame extends JFrame
{
    public CustomFrame(){
        super(”Custom Frame");
        // custom size and background color
        setSize(200,200);
        setBackground(Color.BLUE)
        // other customization statements
    }
}
```

Frame Customization (cont'd)

```
public class CustomFrame extends JFrame
{
    public CustomFrame(){
        super("Custom Frame");
        // custom size and background color
        setSize(200,200);
        setBackground(Color.BLUE)
        // other customization statements
    }

    public static void main(String[] args)
    {
        //example using custom Frame
        JFrame fr = new CustomFrame();
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        fr.setVisible(true);
    }
}
```

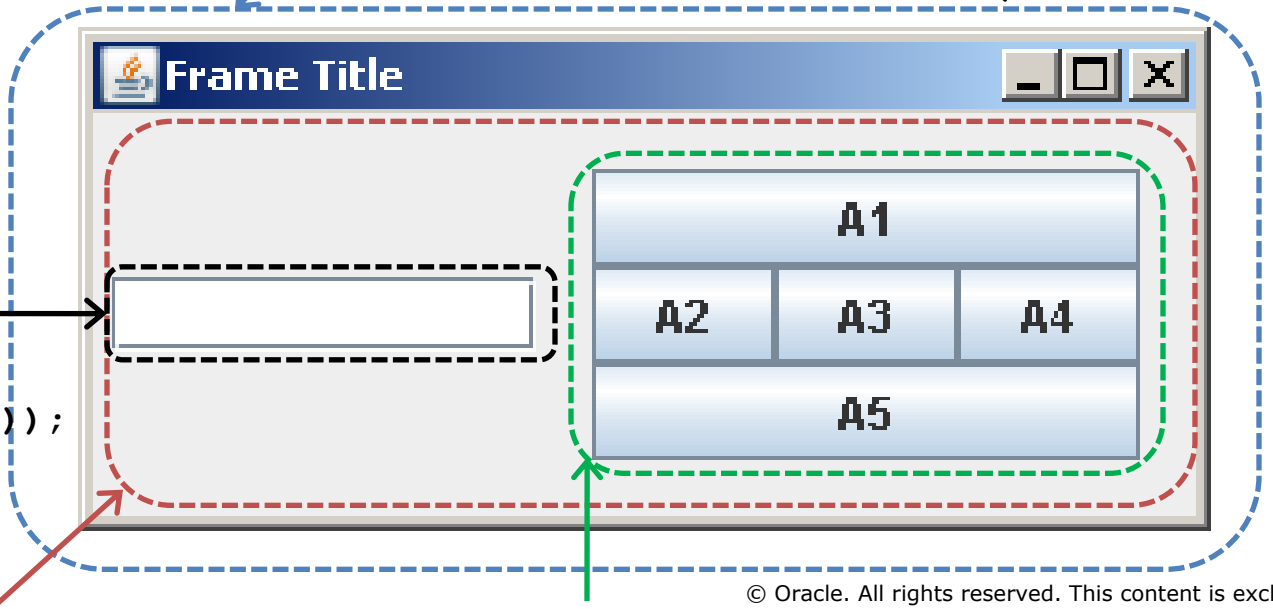
We won't customize JFrame objects in the main() method. Customization statements are placed in the constructor of a class that extends JFrame. The main() method only creates an instance of the custom frame class, makes it visible and sets the default close operation.

Swing Components

JFrame

Has a ContentPane

```
JFrame frame = new JFrame("Frame Title");
```



JTextField

```
cp.add(new JTextField(5));
```


Container (ContentPane)

Contains a JTextField and a JPanel, organized in FlowLayout.

```
Container cp = frame.getContentPane();
cp.setLayout(new FlowLayout());
```

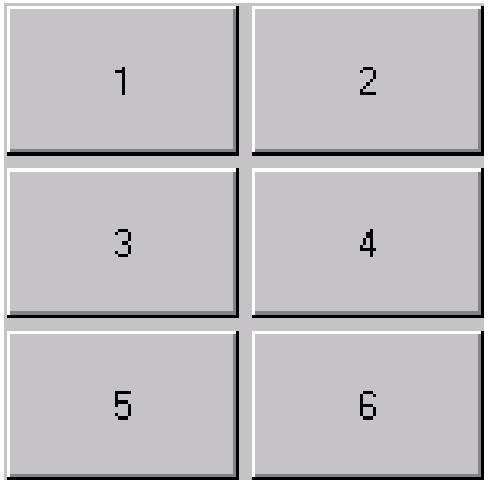
JPanel

Contains 5 JButtons organized in BorderLayout

```
JPanel panel = new JPanel();
panel.setLayout(new BorderLayout());
// add buttons to the panel
cp.add(panel);
```

`GridLayout` Example

The following code creates a custom frame with a content pane resembling the following image.



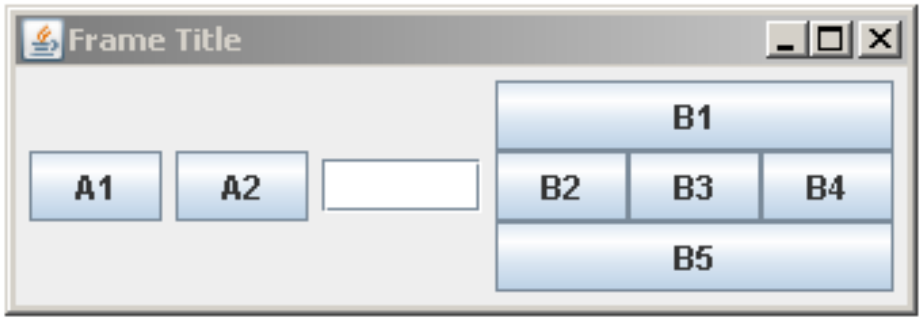

```
// constructor only for custom JFrame
public CustomFrame(){
    super("Frame Title");

    Container cp = getContentPane();
    cp.setLayout(new GridLayout(3, 2)); //override default BorderLayout

    cp.add(new JButton("1"));
    cp.add(new JButton("2"));
    cp.add(new JButton("3"));
    cp.add(new JButton("4"));
    cp.add(new JButton("5"));
    cp.add(new JButton("6"));

    pack();
}
```

Also, call the **`revalidate()`** method if you get strange bugs in your layout. It causes the Container to resize the components and may resolve the bug

The **`pack()`** method sizes the frame so that all its contents are at or above their preferred sizes. This method must be called AFTER all components are added to the frame or panel.

Exercise

Create a frame resembling the following:

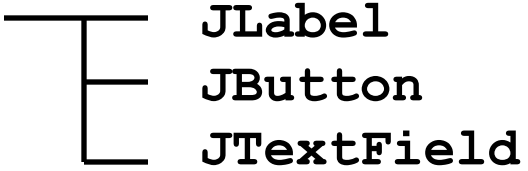


Swing Event Model

Top-level containers **`JFrame`**, **`JDialog`**, **`JApplet`**

Containers **`Container`** ——— **`JPanel`**

Components **`JComponent`** ——— **`JLabel`**, **`JButton`**, **`JTextField`**



Event sources

Events are triggered by **`JComponent`**s.

Example: a **`JButton`** triggers an **`ActionEvent`** when the user clicks it

Event listeners

An object implementing a listener interface can listen to events.

Each listener interface has (a) method(s) that react to events.

Example: an object implementing the **`ActionListener`** interface has an **`ActionPerformed`** method that reacts to **`ActionEvents`** triggered by **`JButtons`**.

Source-listener relationships

Event listeners are registered at event sources

Example: **`aJButton.addActionListener(aListenerObject)`**

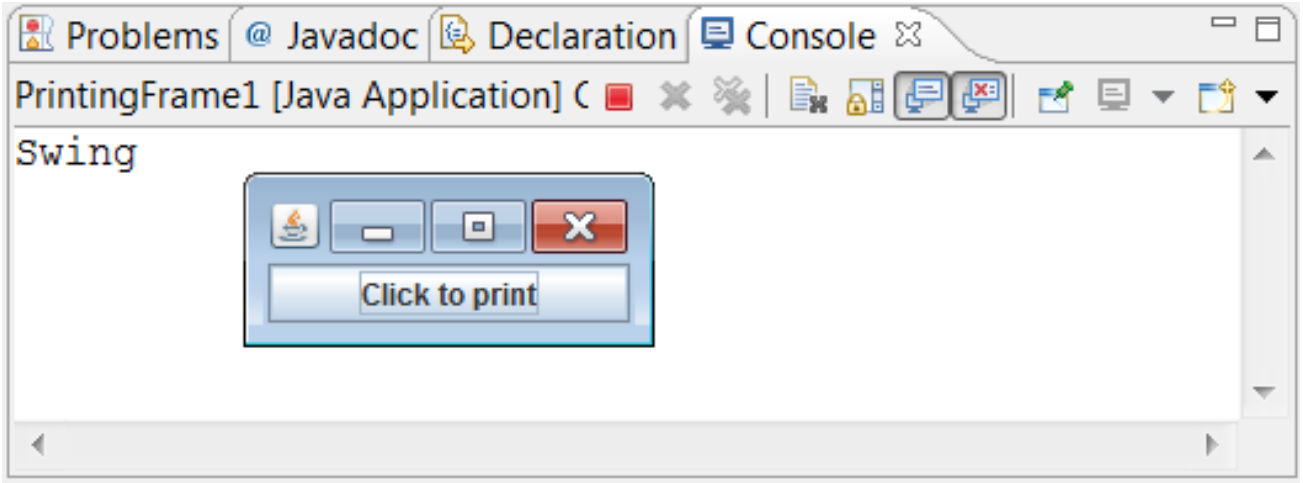
Swing Event Model

3 Types of Source-Listener relationships:

- The listener is the **container**.
- The listener is an **object of an inner class** of the class containing the source.
- The listener is an **anonymous inner class** of the class containing the source.

Example:

A frame holding a JPanel with a button that prints "Swing" to the console when clicked.



Source? The JButton "Click to print"

Listener? An object that has an ActionPerformed method printing "Swing".

Option 1: Container Listens

Complete the PrinterPanel class by implementing a listener for the JButton and print "Swing" when the JButton is clicked

```
public class PrinterPanel extends JPanel {

    JButton b;

    public PrinterPanel(){
        b = new JButton("Click to Print");
        add(b);

    }



}
```

Option 2: Inner Listener Class

Complete the inner class Printer so that it can listen to JButtons and print "Swing" when a JButton is clicked. Then add, to the PrinterPanel class, a Printer object that listens to the existing JButton.

```
public class PrinterPanel extends JPanel{

   JButton b;


   public PrinterPanel(){
      b = new JButton("Click to Print")
      add(b);


   }

   public class Printer {




   }
}
```

Option 3: Anonymous Inner Class

Add an anonymous inner class to the JButton to print "Swing" when the JButton is clicked.

```
public class PrinterPanel extends JPanel{

    JButton b;

    public PrinterPanel(){
        b = new JButton("Click to Print")
        add(b);









    }
}
```

Exercise

We will model a combination lock using Swing.

Open/Close button (**`JButton`**)

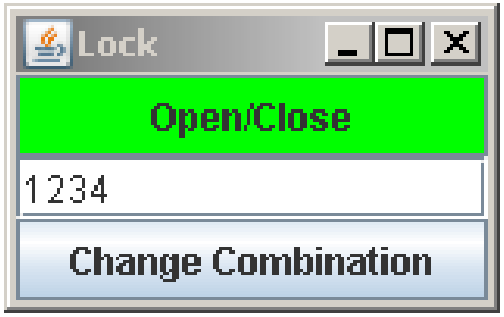
- green when lock is opened, red when it is closed
- when clicked:
  - if the lock is opened, close it.
  - if the lock is closed, open it if digits match combination

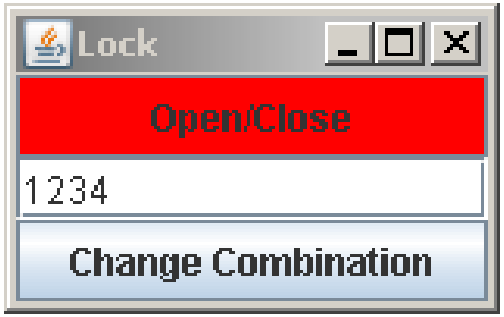
Digit Text field (**`JTextField`**)

- Take input digits for lock combination

Change Combination button (**`JButton`**)

- when clicked:
  - if lock is opened, set the combination to the current digits.





Lock opened

Lock closed

Class Structure

Create a class LockPanel that extends JPanel. Import all necessary packages. Start by adding a main() method to create an instance of the LockPanel, insert it into a JFrame and display it to the user.

- We will implement the LockPanel constructor later.

Data Members

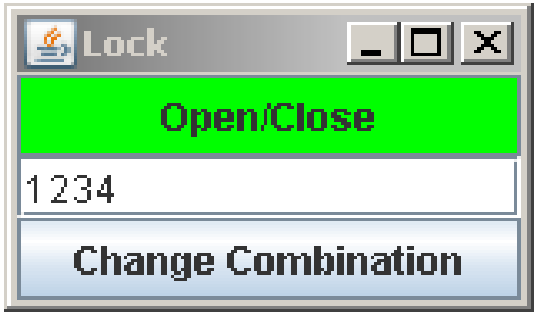
Add the appropriate data members to the LockPanel class.

- Which data members do you need to model the lock and build the GUI?

JComponents

Write the constructor of the LockPanel class. Within the constructor, create and add the appropriate JComponents and input arguments. Initialize the lock to be open.

- **`JButton`** has a **`setBackground(Color)`** method.



Event Listeners

In the constructor, attach an anonymous inner class to the *'change combination'* button. The combination can only be changed when the lock is opened.

- Class JTextField has a **`String getText()`** method
- Convert a string into an integer using **`Integer.parseInt(String)`**

Event Listeners

Attach an anonymous inner class the the *'open/close'* button. The lock can always be closed, but it can be opened only if the digits match the combination.

- **`JButton`** has a **`setBackground(Color)`** method.

Lock opened

Lock closed

Homework 6

Model

- Write the Antenna class(es), which model the antenna using Inheritance and the equations from homework 1

Controller

- Do not need to draw Antennae
- All the textboxes, combo boxes, and buttons should be displayed and functional

1.00 / 1.001 / 1.002 Introduction to Computers and Engineering Problem Solving

Spring 2012