



the
POWER
of
JAVA™



JavaOne
Part of the Network for Enterprise Software

Swing Advanced Testing and Debugging

Alexander Potochkin

Swing team engineer
Sun Microsystems
www.sun.com

BOF-0204

Goal

Learn numerous techniques that make Swing testing and debugging easier

Agenda

Testing

- Non-visual testing

- Visual testing

Debugging

- AWTEventListener

- Focus issues

- MouseEvent

- Uncaught exceptions

- SwingHelper

Agenda

Testing

Non-visual testing

Visual testing

Debugging

AWTEventListener

Focus issues

MouseEvent

Uncaught exceptions

SwingHelper

Do As Much Model Testing As Possible

- Use listeners for testing
 - Check for notifications to be sent
 - Detect unexpected reentrancy
 - Is it invoked on a proper thread ?
- Implement testing model
 - Validate component's usage of model
 - Keep track of model's state
- You can use `java.lang.reflect.Proxy` to make it easier

Using Listeners

```

JButton button = new JButton();

button.addPropertyChangeListener("text",
    new PropertyChangeListener() {
        public void propertyChange(PropertyChangeEvent e) {

            if (!"Hello".equals(e.getNewValue())) {
                throw new RuntimeException();
            }
        }
    });

button.setText("Hello");
  
```

Using java.lang.reflect.Proxy

```
class TestInvocationHandler
    implements InvocationHandler {
    private final Object myObject;

    public Object invoke(Object proxy,
        Method method, Object[] args) throws Throwable {

        // Add common logic for all methods here

        return method.invoke(myObject, args);
    }
}
```

Using java.lang.reflect.Proxy

```
public Object invoke(Object proxy,  
    Method method, Object[] args) throws Throwable {  
  
    if(!SwingUtilities.isEventDispatchThread()) {  
        throw new RuntimeException("Wrong thread");  
    }  
  
    return method.invoke(myObject, args);  
}  
  
TableModel m = (TableModel) Proxy.newProxyInstance(...);
```

Painting To Offscreen Buffer

Paint component to an image

- Easy way to create a set of images for a component in a different states
- Stored images can be useful for automated regression testing (golden image approach)
- You can compare images from the current build with images from the last stable build (golden build approach)

Painting To Offscreen Buffer

Taking screenshots

```
 JButton b = new JButton("Hello");  
 button.setSize(100, 50); // set initial size  
  
 BufferedImage image =  
     new BufferedImage(b.getWidth(), b.getHeight(),  
                       BufferedImage.TYPE_INT_RGB);  
  
 Graphics g = image.getGraphics();  
  
 button.paint(g);  
  
 ImageIO.write(image, "png", new File("button.png"));
```

Painting To Offscreen Buffer

`javax.swing.DebugGraphics`

- Useful for painting routine testing
- You can log all the information about painting operations
- Gives more information than golden image
- Note that It doesn't extend Graphics2D
 - If you need it to be Graphics2D instance, make your own wrapper

Painting To Offscreen Buffer

Creating tests

```
BufferedImage image = new BufferedImage(  
    width,height,BufferedImage.TYPE_INT_RGB);  
Graphics g = image.getGraphics();
```

```
button.paint(new DebugGraphics(g) {  
    public void setFont(Font aFont) {  
        if (aFont.getSize2D() != 12) {  
            throw new RuntimeException(  
                "Wrong font size !");  
        }  
        super.setFont(aFont);  
    }  
});
```

Agenda

Testing

- Non-visual testing

- Visual testing**

Debugging

- AWTEventListener

- Focus issues

- MouseEvent

- Uncaught exceptions

- SwingHelper

Visual Testing

Common pattern

- Set up your components and show frame on Event Dispatch Thread (EDT)
- Wait for events to be processed
- Use Robot if you need to generate input events
- Check component's state on EDT

Visual Testing

Common pattern

```
SwingUtilities.invokeLater(new Runnable() {  
    public void run() {  
        //Setting up, showing the frame  
    }  
});
```

```
Thread.sleep(500); //Wait for events to be processed
```

```
SwingUtilities.invokeLater(new Runnable() {  
    public void run() {  
        //Check components' state  
        //Throw exception if something is wrong  
    }  
});
```

Robot Makes Testing Easier

`java.awt.Robot`

- Test automation
 - Generating native system input events
 - Keyboard events
 - Mouse events
- Screen capture
 - Screenshots of your application
 - Screenshots of particular components

Robot Makes Testing Easier

`java.awt.Robot`

- `Robot.waitForIdle()`
 - Doesn't wait for some native events
 - Don't rely on it
- Set reasonable delays
 - Allow at least 10ms between Key/MouseEvents
 - Don't exceed key repeat rate unexpectedly
 - Implement mouse dragging with several `mouseMove()`

Using Robot

```
SwingUtilities.invokeLater(new Runnable() {  
    public void run() {  
        frame.add(textField);  
        frame.setVisible(true); // make it visible  
    }  
});
```

```
robot.delay(500); // wait for the frame to be shown  
robot.setAutoDelay(10); // set 10 ms delay  
robot.keyPress(KeyEvent.VK_J); // press "j"  
robot.keyRelease(KeyEvent.VK_J);
```

```
SwingUtilities.invokeLater(new Runnable() {  
    public void run() {  
        if (!textField.getText().equals("j"))  
            throw new RuntimeException("Where is our j ?");  
        frame.dispose(); //to exit normally  
    }  
});
```

Swing Testing Projects:

- Jemmy
 - <http://jemmy.netbeans.org/>
 - Last update Apr 20, 2006
- Abbot
 - <http://abbot.sourceforge.net/>
 - Last update Dec 22, 2005
- JFCUnit
 - <http://jfcunit.sourceforge.net/>
 - Last update Dec 21, 2004
- Marathon
 - <http://marathonman.sourceforge.net/>
 - Last update Nov 7, 2004

Agenda

Testing

- Non-visual testing

- Visual testing

Debugging

- AWTEventListener

- Focus issues

- MouseEvent

- Uncaught exceptions

- SwingHelper

Agenda

Testing

- Non-visual testing

- Visual testing

Debugging

- AWTEventListener**

- Focus issues

- MouseEvent

- Uncaught exceptions

- SwingHelper

java.awt.AWTEventListener

Receives all AWTEvents

- Debugging
 - Focus problems
 - Key events
 - Components hierarchy changes
 - Many more...
- Exploring
 - What events were generated ?
 - What component receives Mouse/Keyboard events ?
 - Many more...

Agenda

Testing

- Non-visual testing

- Visual testing

Debugging

- AWTEventListener

Focus issues

- MouseEvent

- Uncaught exceptions

- SwingHelper

Tracking Focus Events

Using AWTEventListener

```
Toolkit.getDefaultToolkit().addAWTEventListener(  
    new AWTEventListener() {  
        public void eventDispatched(AWTEvent event) {  
            System.out.println("event = " + event);  
        }  
    }, AWTEvent.FOCUS_EVENT_MASK |  
    AWTEvent.WINDOW_FOCUS_EVENT_MASK );
```

Focus Issues

Listening to KeyboardFocusManager

```
KeyboardFocusManager.getCurrentKeyboardFocusManager().  
    addPropertyChangeListener("focusOwner",  
  
        new PropertyChangeListener() {  
            public void propertyChange(PropertyChangeEvent e) {  
  
                System.out.println("New focusOwner is " +  
                    e.getNewValue());  
  
            }  
        }  
    );
```

Focus Issues

Overriding requestFocus*() methods

```

frame.add(new JButton("Hello")) {

    public boolean requestFocusInWindow() {
        Thread.dumpStack(); // who requested focus ?
        return super.requestFocusInWindow();
    }

    public void requestFocus() {
        Thread.dumpStack();
        super.requestFocus();
    } // and so on

}
  
```

Agenda

Testing

- Non-visual testing

- Visual testing

Debugging

- AWTEventListener

- Focus issues

MouseEvent

- Uncaught exceptions

- SwingHelper

MouseEvent

Transparency example

```
MouseListener ma = new MouseAdapter() {  
    public void mouseClicked(MouseEvent e) {  
        System.out.println("MouseClicked");  
    }  
};
```

```
frame.addMouseListener(ma);
```

```
JLabel label = new JLabel("Label");
```

```
// MouseEvent go through the label  
frame.add(label);
```

MouseEvent

Broken transparency

```
MouseListener ma = new MouseAdapter() {
    public void mouseClicked(MouseEvent e) {
        System.out.println("MouseClicked");
    }
};

frame.addMouseListener(ma);
JLabel label = new JLabel("Label");

// it breaks transparency
label.setToolTipText("Hello");

frame.add(label);
```

MouseEvent

ToolTip example

```
// No mouse related listeners initially
// label is "transparent" for mouse events
System.out.println(
    label.getMouseListeners().length); // 0

label.setToolTipText("Hello");

// MouseListener attached,
// label doesn't pass mouseEvents to the frame
System.out.println(
    label.getMouseListeners().length); // 1
```

MouseEvent

Summary

- If a component doesn't have any Mouse/MouseMotion/MouseWheelListeners it is “transparent“ for MouseEvent
- Adding any mouse related listener to such a component changes its behavior
- **Add mouse related listeners from parent directly to a child component, not relying on “transparency”**

MouseEvent

Correct example

```
MouseListener ma = new MouseAdapter() {  
    public void mouseClicked(MouseEvent e) {  
        System.out.println("MouseClicked");  
    }  
};
```

```
frame.addMouseListener(ma);  
JLabel label = new JLabel("Label");
```

```
label.addMouseListener(ma);  
frame.add(label);
```

MouseEvent

Unsafe recursion

```
static void addMyMouseListener(  
    Component c, MouseListener ma) {  
    c.addMouseListener(ma); // add mouse listener  
  
    if (c instanceof Container) {  
        Component[] cs =  
            ((Container) c).getComponents();  
  
        for (Component c1 : cs) {  
            addMyMouseListener(c1, ma);  
        }  
    }  
}
```

MouseEvent

Better version

```
static void addMouseListener(
    Component c, MouseListener ma) {

    if (c.getMouseListeners().length > 0 ||
        c.getMouseMotionListeners().length > 0 ||
        c.getMouseWheelListeners().length > 0) {
        c.addMouseListener(ma);
    }

    if (c instanceof Container) {
        Component [] cs =
            ((Container) c).getComponents();
        for (Component c1 : cs) {
            addMouseListener(c1, ma);
        }
    }
}
```

Agenda

Testing

- Non-visual testing

- Visual testing

Debugging

- AWTEventListener

- Focus issues

- MouseEvent

Uncaught exceptions

- SwingHelper

Uncaught Exceptions

```
 JButton button = new JButton("Hello");

 button.addActionListener(new ActionListener() {

     public void actionPerformed(ActionEvent e) {
         throw new RuntimeException();
     }

 });

 frame.add(button);
```

Uncaught Exceptions

for Event Dispatch Thread

```
public class CatchException {  
  
    public void handle(Throwable e) {  
        System.out.println("Uncaught exception on EDT !");  
    }  
  
    public static void main(String[] args) {  
  
        System.setProperty("sun.awt.exception.handler",  
                            "mypackage.CatchException");  
    }  
}
```

Uncaught Exceptions

for all threads

```

Thread.setDefaultUncaughtExceptionHandler (
    new Thread.UncaughtExceptionHandler() {
public void uncaughtException(Thread t, Throwable e) {

    System.out.println("UncaughtException !");

    //Any exception thrown by this method
    //will be ignored
    //System.exit(1);
    }
});
// It doesn't work for EDT
// if a modal dialog is shown !
  
```

Agenda

Testing

- Non-visual testing

- Visual testing

Debugging

- AWTEventListener

- Focus issues

- MouseEvent

- Uncaught exceptions

- SwingHelper**

SwingHelper

<https://swinghelper.dev.java.net/>

- Collecting and discussing all kinds of debugging techniques for Swing
- Invaluable examples:
 - CheckThreadViolationRepaintManager
 - EventDispatchThreadHangMonitor
- Sample code with Proxy interfaces, DebugGraphics and more
- Join the mailing lists and share your opinion !

SwingHelper

CheckThreadViolationRepaintManager

- All Swing related work must be done on Event Dispatch Thread (EDT) only
- It is usually difficult to find the place where Swing accessed from the wrong thread
- Custom RepaintManager detects EDT rule violations
- Created by Scott Delap <http://www.clientjava.com/>

SwingHelper

EventDispatchThreadHangMonitor

- Time consuming tasks shouldn't be executed on EDT, because it leads to GUI hangs
- Custom EventQueue monitors the EDT for events that take longer than a certain time to be dispatched
- Detects EDT deadlocks
- Created by Elliott Hughes <http://elliottth.blogspot.com/>

Summary

- Write as many automated tests as you can
- Swing has numerous ways to automate testing
- Swing is easy to debug

For More Information

- Coming sessions:
 - **TS-4855**: Swing Threading 101
Wednesday 11:00 AM - 12:00 PM
 - **BOF-0588**: Meet the Swing, AWT, and I18N Teams
Wednesday 07:30 PM - 08:20 PM
- URL's
 - <http://www.clientjava.com/>
 - <http://elliottth.blogspot.com/>
 - <https://swinghelper.dev.java.net/>

Q&A

Alexander.Potochkin@sun.com

<https://swinghelper.dev.java.net/>