

```
/**
 *
 *
 * BEST PRACTICES FOR BUILDING LIBRARIES
 *
 * @author Kelly Norton & Joel Webber
 *
 */
// voices that matter: google web toolkit
```

```
/**  
 *  
 *  
 * There is a difference in designing libraries  
 * vs. designing applications.  
 *  
 *  
 *  
 */
```

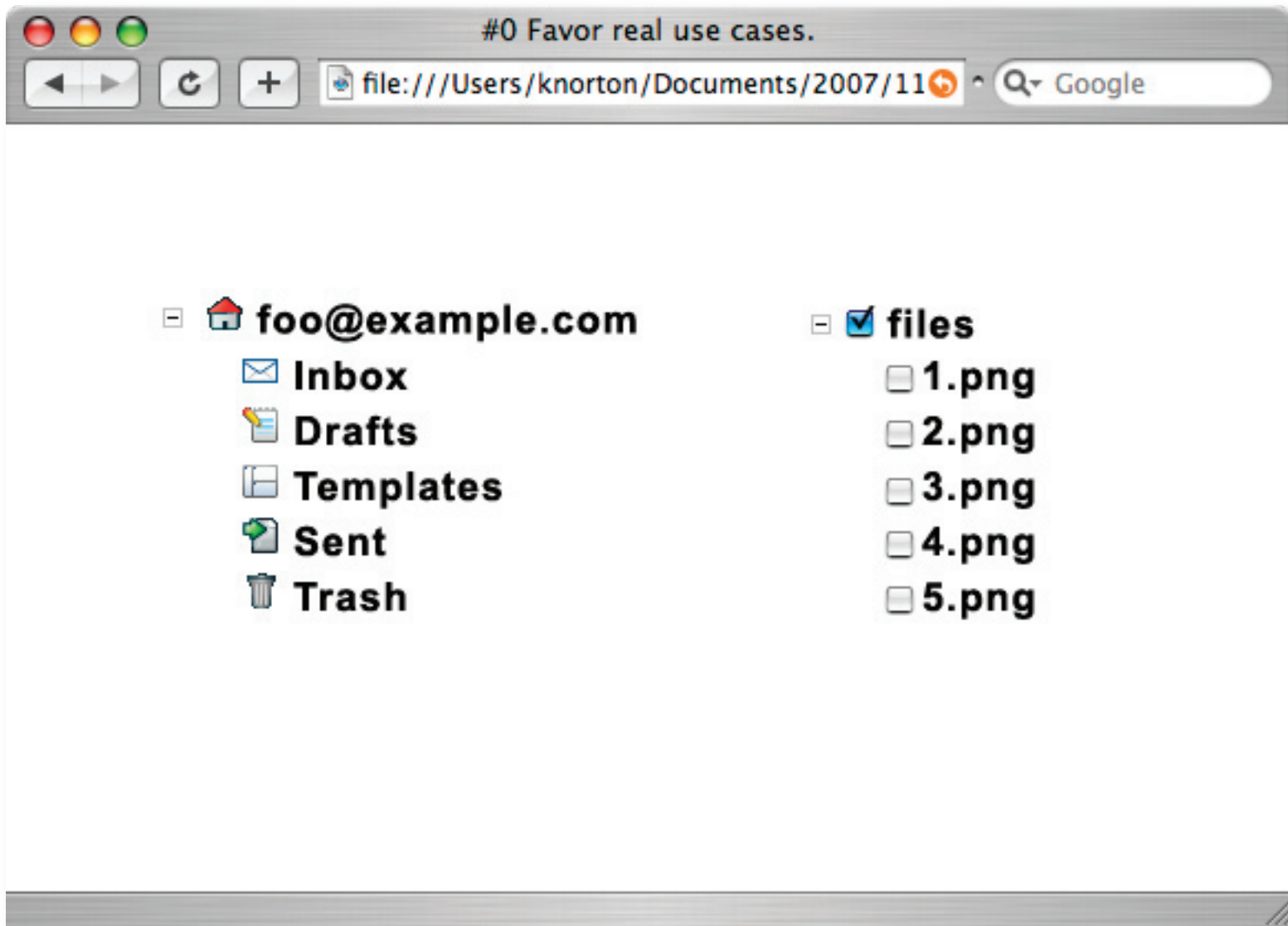
```
/**  
 *  
 *  
 * #0  
 * Design to real use cases not someone's dreamscape.  
 *  
 *  
 *  
 */
```

```
// Example #1:
```

```
Tree tree1, tree2, tree3;  
tree2.add(tree3)  
tree1.add(tree2);
```

```
// Example #2:
```

```
AbstractImagePrototype openIcon, closeIcon;  
Image image = openIcon.createImage();  
// Later  
closeIcon.applyTo(image);
```



```
/**  
 *  
 *  
 * #1  
 * Create independently useful pieces.  
 *  
 * Example: JSON, XML independent of RequestBuilder  
 *  
 */
```

```
/**  
 *  
 *  
 * #2  
 * Don't use a Widget when all you need is HTML.  
 *  
 *  
 */
```

```
class MyWidget extends Widget {  
  MyWidget() {  
    Element elem = DOM.createImage();  
    DOM.setElementAttribute(elem, "src",  
      "dogs.gif");  
    DOM.appendChild(getElement(), elem);  
  }  
}
```

```
/**  
 *  
 *  
 * #3  
 * Design with compiler optimizations in mind.  
 *  
 *  
 */
```

```
// Old way of doing things.  
Tree tree;  
tree.setImageBase("images");
```

```
// New way compiles with no trace of the old way.  
TreeImages images;  
Tree tree = new Tree(images);
```

```
// Example: DOM has no static initializers.  
// So this Java  
Element elem = DOM.getElementById("root");  
DOM.setStyleAttribute(elem, "width", "600px");
```

```
// can become this JavaScript:  
var a;a=$doc.getElementById(l)||null;a.style[m]=n;
```

```
/**  
 *  
 *  
 * #4  
 * Identify your invariants; make them immutable.  
 *  
 *  
 */
```

```
// A relic of the past.  
LinearPanel p = new LinearPanel(LinearPanel.VERTICAL);  
p.setDirection(LinearPanel.HORIZONTAL);  
  
// But nobody changes the direction, so  
HorizontalPanel h = new HorizontalPanel();  
VerticalPanel v = new VerticalPanel();
```

```
/**  
 *  
 *  
 * #5  
 * Beware of implementation inheritance.  
 *  
 *  
 */
```

// Example: TextBoxBase is not really meaningful.

```
class TextBox extends TextBoxBase {  
}
```

```
class RichTextArea /* does not */ extends  
    TextBoxBase {  
}
```

```
/**  
 *  
 *  
 * #6  
 * Make it work on Mozilla, IE6/7, Safari and Opera.  
 *  
 *  
 */
```

```
// Example: SVG is the wrong abstraction.
```

```
// Example: Why is it hard to send HEAD with
```

```
// RequestBuilder?
```

```
new RequestBuilder(  
    /* RequestBuilder.HEAD :-( */,  
    "url");
```

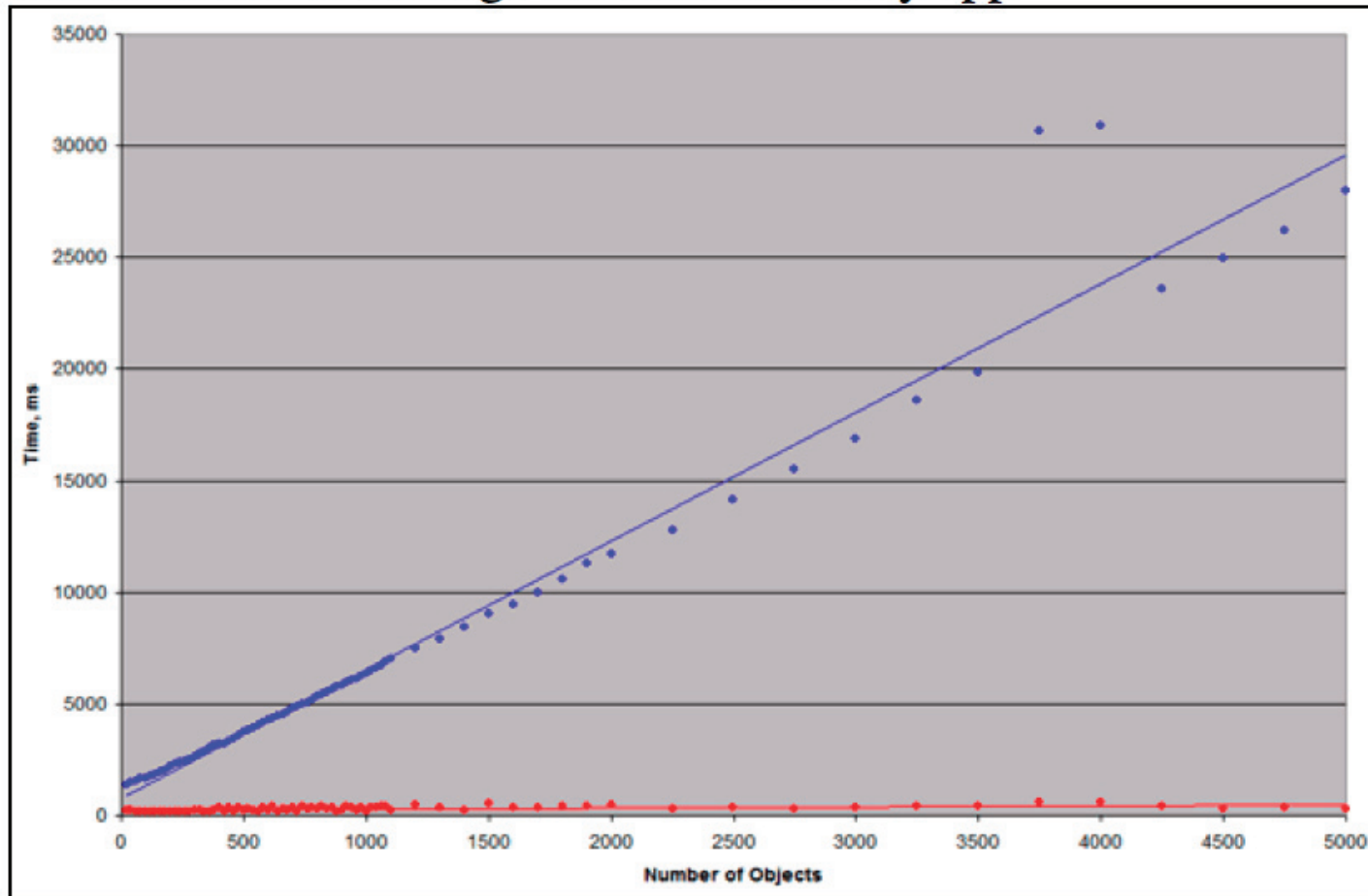
```
/**  
 *  
 *  
 * #7  
 * Use the builder pattern.  
 *  
 *  
 */
```

```
RequestBuilder rb = new RequestBuilder(  
    RequestBuilder.GET, "/my_url");  
rb.setUser("gpburdell");  
rb.setPassword("*****");  
rb.setHeader("X-Foo", "Bar");  
rb.sendRequest(null, new RequestCallback() {  
    ...  
});
```

```
/**  
 *  
 *  
 * #8  
 * 00 is a tool, not a goal.  
 *  
 *  
 */
```

```
// Objects for the sake of objects have additional  
// downsides in browsers.  
public class Dimension {  
    public int x, int y;  
}  
widget.setSize(new Dimension(420, 300));
```

## OMG! The IE6 Garbage Collector kills my app



Dan Papius: <http://papius.co.uk/log/2007-03-07/>

```
/**  
 *  
 *  
 * #9  
 * Read Effective Java.  
 *  
 *  
 */
```

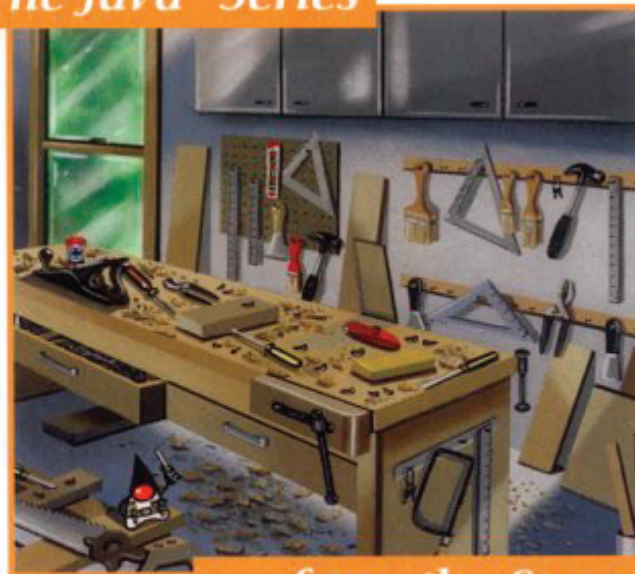
Joshua Bloch

# Effective Java™

## Programming Language Guide

Foreword by Guy Steele

*The Java™ Series*



*... from the Source™*



```
/**  
*  
*  
* #a  
* Ask us questions.  
*  
* @email knorton@google.com  
* @email jgw@google.com  
*/
```