

Dependable Software Systems

Topics in Unit Testing Tools

Material drawn from [junit.org, jcoverage.com]
Courtesy Spiros Mancoridis



Software Engineering (Unit Testing Tools)

junit and jcoverage

- We will use a combination of two tools to test Java programs:
 - *junit* is a unit testing tool
 - *jcoverage* is a statement/branch coverage tool
- Both tools are available for free from the WWW.
- *jcoverage* has its own testing tool (*jtestrun*) but we will use *junit* as a testing harness.
 - *junit* is included in the *jcoverage* distribution.



Downloading the Software

- Download *junit* from:
 - www.junit.org
- Download *jcoverage* from:
 - www.jcoverage.com
- Set your Java CLASSPATH to include:
 - `jcoverage.jar, junit.jar, log4j.jar, bcel.jar, jakarta-oro.jar, java-getopt.jar`
 - `junit.jar, log4j.jar, bcel.jar, jakarta-oro.jar, java-getopt.jar` are in the *jcoverage* distribution under the `lib` directory.



Implementation of the SimpleStack class

```
import java.util.Vector;

class SimpleStack {

    private Vector s = null;

    public SimpleStack (int initSize) {
        s = new Vector(initSize);
    }

    public void push (Object o) {
        s.addElement(o);
    }
}
```

```
public Object pop () {
    Object top = null;

    if (s != null && s.size() != 0) {
        top = s.elementAt(s.size()-1);
        s.removeElementAt(s.size()-1);
    }

    return top;
}

public Object top () {
    Object o = null;

    if (s != null && s.size() != 0)
        o = s.elementAt(s.size()-1);

    return o;
}
}
```



Testing the SimpleStack class using a junit StackTest class

```
import junit.framework.*;
import java.util.Vector;

public class StackTest extends TestCase {
    private SimpleStack s;

    public static void main (String args[]) {
        junit.textui.TestRunner.run (suite());
    }

    public static Test suite() {
        return new TestSuite(StackTest.class);
    }

    protected void setUp() {
        s = new SimpleStack(100);
    }
}
```

```
public void testPushOne() {
    s = new SimpleStack(100);
    String hiString = new String("Hi");

    s.push(hiString);
    assertTrue(s.top() == hiString);
    System.out.println(hiString);
}

public void testPopEmpty () {
    s = new SimpleStack(0);
    Object o = s.pop();
    assertTrue(o == null);
}

public void testTopEmpty () {
    s = new SimpleStack(0);
    assertTrue(s.top() == null);
}
}
```



Run the StackTest class test cases in batch mode

- Compile SimpleStack and StackTest
 - Don't forget to set your CLASSPATH.
- Then execute the StackTest test cases.
 - Use batch or GUI mode

```
java junit.textui.TestRunner StackTest
```

```
Output:
```

```
-----
```

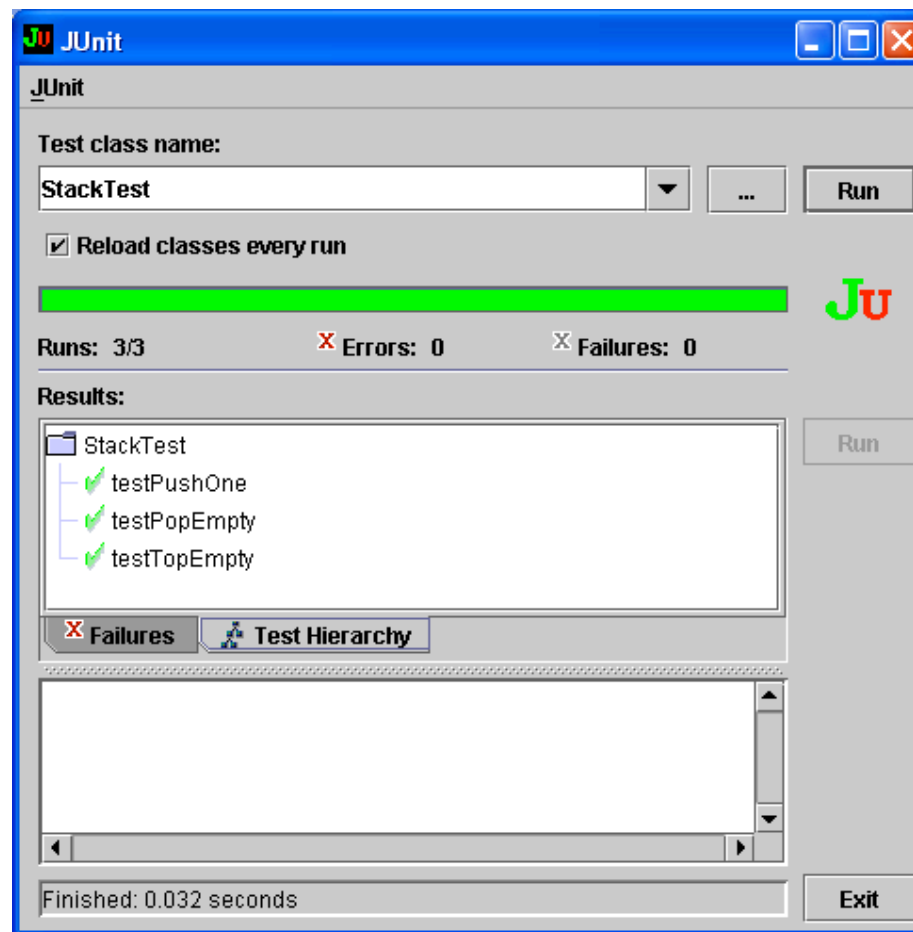
```
Time: 0
```

```
OK (3 tests)
```



Run the StackTest class test cases in GUI mode

```
java junit.swingui.TestRunner StackTest
```



Coverage Tools

- Tools like *jcoverage* can be used to determine the degree of comprehensiveness of a test suite.
- This is important because a test suite may have many test cases, but may only cover a small percentage of the source code.
- Covering all source statements doesn't guarantee much, but NOT covering them is a sign of trouble.



Coverage Analyzers

- Coverage analyzers must instrument the code to keep track of which statements are executed.
- Code instrumentation for Java programs can be done in various ways:
 - Modify the source code
 - Modify the byte code (what *jcoverage* does)
 - Modify the Java VM



jcoverage

byte code instrumentation

- `java com.jcoverage.coverage.Instrument`
 `[-ignore ignore-regex] [-d destination-directory]`
 `[@classlist-file...] [classfiles...]`
- Important options:
 - `-d destination-directory`
 Directory where instrumented classes are written. If this isn't specified, the classes are instrumented in place (overwrite mode).
 - `classfiles`
 Specifies the list of classes to be instrumented.



jcoverage

byte code instrumentation

- The instrumentation program generates a new version of each class.
 - It is good practice to save the new versions of the bytecode in a separate directory.
 - Don't instrument instrumented bytecode.
- The instrumentation program generates a *jcoverage.ser* file.
 - Don't delete this file before you execute the instrumented code.
 - The file contains a integer entry for each line of code that is incremented when the code gets executed (tested).



Example invoking the jcoverage Instrumentation program

```
set ICLASSES=c:\Documents and Settings\smancori\Calculator\classes
```

```
java com.jcoverage.coverage.Instrument -d "%ICLASSES%" *.class
```

- Execute this code from the directory that contains the bytecode of the program.
- The instrumented bytecode will be placed in the location specified in the ICLASSES environment variable.
- Make sure that the following programs are in your CLASSPATH environment variable:
 - log4j.jar
 - bcel.jar
 - jcoverage.jar
 - jakarta-oro.jar,
 - junit.jar
 - java-getopt.jar



jcoverage

HTML Report Generation

- `java com.jcoverage.coverage.reporting.Main`
 `[-i instrumentation-file][-o reports-directory]`
 `[-s java-source-directory]`
- options:
 - `-i instrumentation-file`
 The instrumentation file to generate reports (i.e.,
 jcoverage.ser)
 - `-o reports-directory`
 Directory where HTML report is created
 - `-s java-source-directory`
 Directory containing the Java source code.



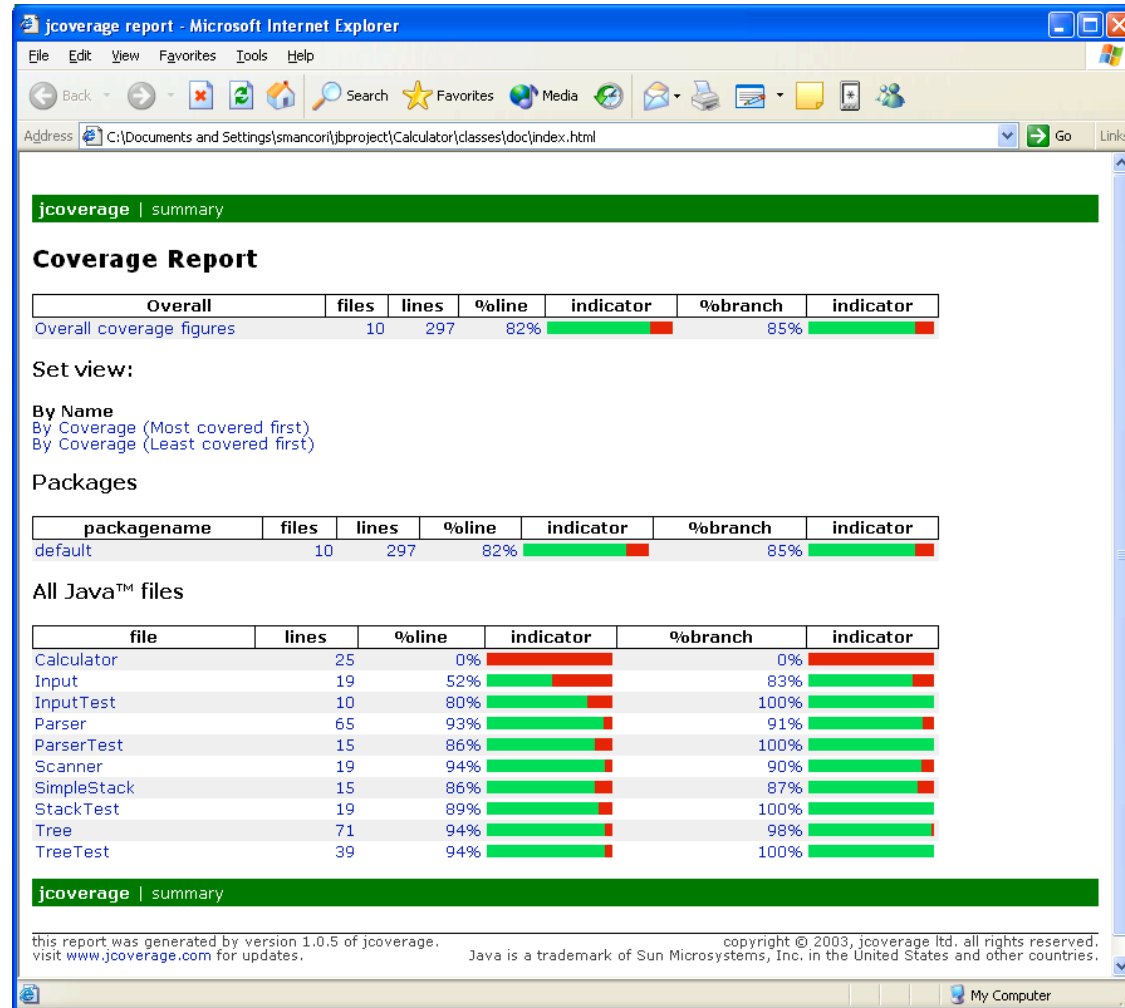
jcoverage

HTML Report Generation

- Generates a multi-page coverage report in HTML from an instrumentation file.
 - Statement coverage percentage
 - Branch coverage percentage
 - Bar-chart visualization used
- The report has links to HTML versions of the source code:
 - Highlighted statements that were not executed.
 - Hit counts for statements that were executed multiple times.



HTML Coverage Report Statement & Branch Coverage



HTML Coverage Report Source Code Annotations

Line	Hits	Source
1		import java.io.*;
2		
3		class Input {
4		
5		private BufferedReader is;
6	1	private String input = "";
7		
8		/* Read from a specified file */
9	1	Input(String fileName) {
10		try {
11	1	is = new BufferedReader(new FileReader(fileName));
12	0	} catch (Exception e) {
13	0	e.printStackTrace();
14	0	System.exit(1);
15		}
16	2	}
17		
18		/* Read from stdin */
19	0	Input() {
20	0	is = new BufferedReader(new InputStreamReader(System.in));
21	0	}
22		
23		/* Read the entire input and store it into a large string */
24		public String input2String () {
25	1	String input = new String("");
26		
27		try {
28	3	while (true) {
29	4	String line = is.readLine();
30		
31	4	if (line == null) break;



Example invoking the jcoverage HTML Report Generation program

```
set SERFILE=c:\Documents and Settings\smancori\Calculator\jcoverage.ser
set SRCFILES=c:\Documents and Settings\smancori\Calculator
set OUTFILES=c:\Documents and Settings\smancori\Calculator\doc
```

```
java com.jcoverage.coverage.reporting.Main -i "%SERFILE%"
-o "%OUTFILE" -s "%SRCFILES%"
```

- Execute this program from any directory.
- The HTML files for the report will be placed in the location specified by OUTFILES.
- SERFILE directs the program to the instrumentation file (i.e., *jcoverage.jar*)



jcoverage

XML Report Generation

- `java com.jcoverage.coverage.reporting.xml.Main`
 `[-i instrumentation-file] [-o reports-directory]`
 `[-s java-source-directory]`
- options:
 - `-i instrumentation-file`
 The instrumentation file to generate reports (i.e.,
 jcoverage.ser)
 - `-o reports-directory`
 Directory where HTML report is created
 - `-s java-source-directory`
 Directory containing the Java source code.



jcoverage

Merge Instrumentation program

- `java com.jcoverage.tool.merge.Main`
 - `[-i instrumentation-file ...]`
 - `[-o destination-directory]`
- options:
 - `-i instrumentation-file ...`

The two or more instrumentation files to merge into a unified instrumentation file (i.e., *jcoverage.ser*)
 - `-o destination-directory`

Directory where the unified instrumentation file is created.



jcoverage

Merge Instrumentation program

- Merges two or more instrumentation files into a single instrumentation file.
- A single integrated report can be created from multiple coverage runs.
- The integrated report can then be fed to the Report Generation program to produce an integrated report.



Example invoking the jcoverage Merge Instrumentation program

```
set MERGESERDIR=c:\Documents and Settings\smancori\Calculator
```

```
java com.jcoverage.tool.merge.Main -i "T1\jcoverage.ser" -i  
"T2\jcoverage.ser" -i "T3\jcoverage.ser" -o "%MERGESERDIR%
```

- Execute this program from the parent directory of the directories containing the *jcoverage.ser* files to be merged.
- The merged *jcoverage.ser* file will be created in the location specified by MERGESERDIR.



References

- jcoverage site: www.jcoverage.com
- junit site: www.junit.org

