ORACLE

JTREG TUTORIAL

March 01, 2013

- Balchandra Vaidya

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Table of Contents

Section I: Getting started

1. Introduction

1. Introduction

- The primary purpose of jtreg tool is for developing unit tests
    - It can also be used for writing functional tests.
- The jtreg tool can be used for developing and running tests for java APIs, compilers, VM and GUI components.
- The jtreg tool use JavaTest harness which is a set of tools designed to execute test programs. http://jtharness.java.net/.
- Information about jtreg is available at
    - http://openjdk.java.net/jtreg/
    - http://openjdk.java.net/projects/code-tools/

Section I: Getting started

1. Introduction
2. Nuts and Bolts

2. Nuts and Bolts (1/4)

- A test case is a
  - java program with usual static main method (main test)
  - an applet (applet test)
  - a shell script (shell test)
- A test case should define Pass-Fail criteria.
  - if the test fails, it should throw an exception.
  - if it succeeds, it should return normally.
  - Not recommended to catch general exceptions such as Throwable, Exception, or Error.

2. Nuts and Bolts (2/4)

- A test case is distinguished and controlled by tags
  - **@test** tag identifies a source file that defines a test. This is a mandatory tag. This must be the first tag in a comment block
  - **@run** tag tells the harness how to perform the test.
  - **@summary** tag is used to provide brief description of the test case.
  - **@library** tag is used to reference library source files.
  - **@build** tag is used to compile dependent classes or libraries before running the tests
  - **@compile** tag is used to compile a class. It is useful for writing compiler tests
- More tags and options are available in
  - doc/jtreg/tag-spec.txt in jtreg bundle
  - jtreg -onlineHelp

2. Nuts and Bolts(3/4)

- @run tag is optional for automated tests
  - This tag can be used to run tests with non-default options.
- Manual tests use /manual option with @run tag
  - @run applet/manual
  - @run main/manual
  - @run shell/manual
- Negative test is marked with /fail option
  - @run main/fail
  - @compile/fail

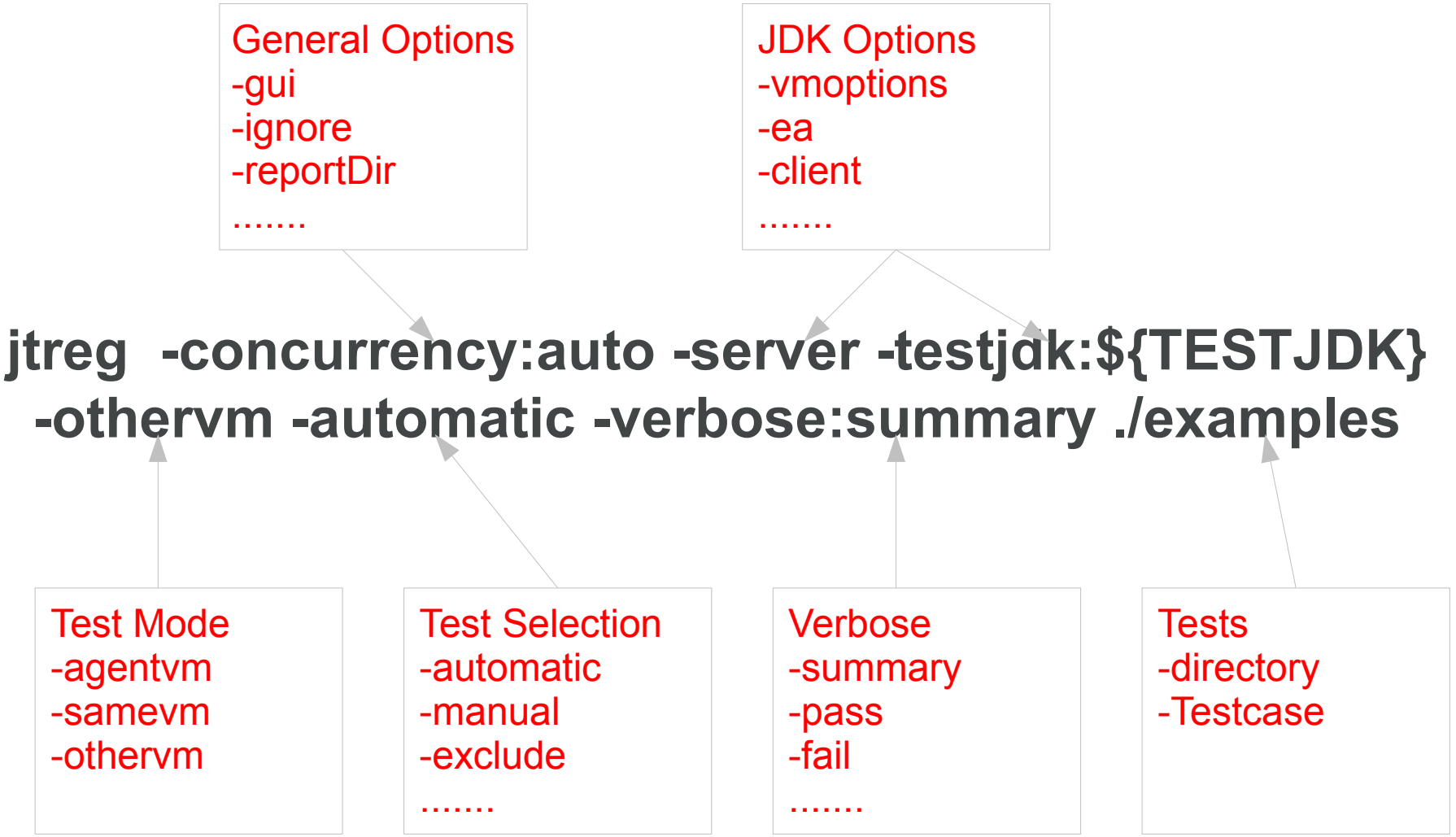
2. Nuts and Bolts (4/4)

- Applet test requires an HTML file which should contain an applet tag with any necessary parameters.
  - Extension should be .html
- Shell tests should
  - have extension .sh
  - use Bourne shell syntax
- Test fails if tests take longer than 120 seconds
  - It can be controlled by /timeout option e.g. @run main/timeout=100

Section I: Getting started

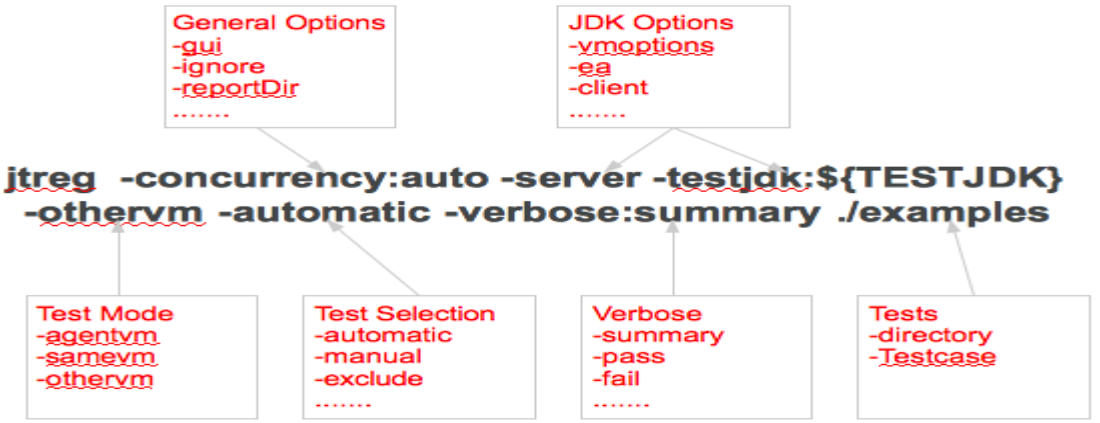
1. Introduction
2. Nuts and Bolts
3. Test Execution

3. Test Execution



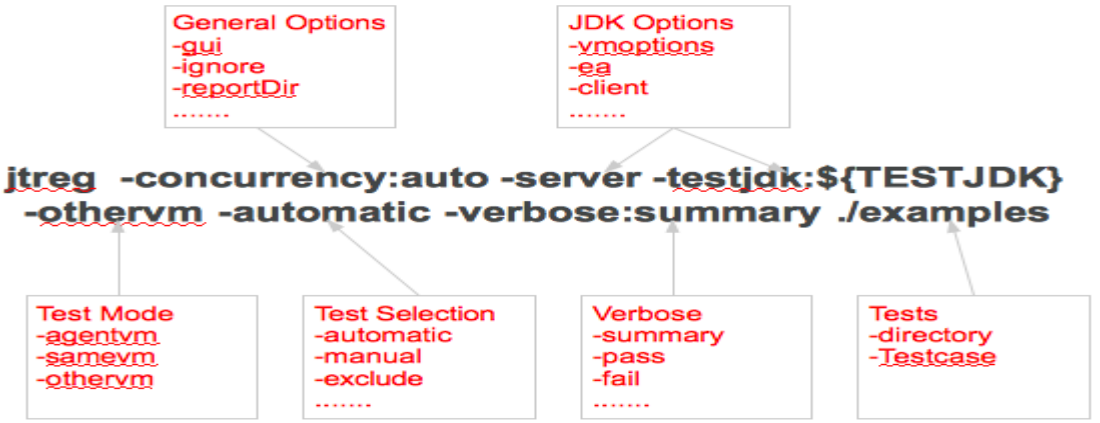
- Online help: jtreg -onlineHelp

3. Test Execution (1/4)



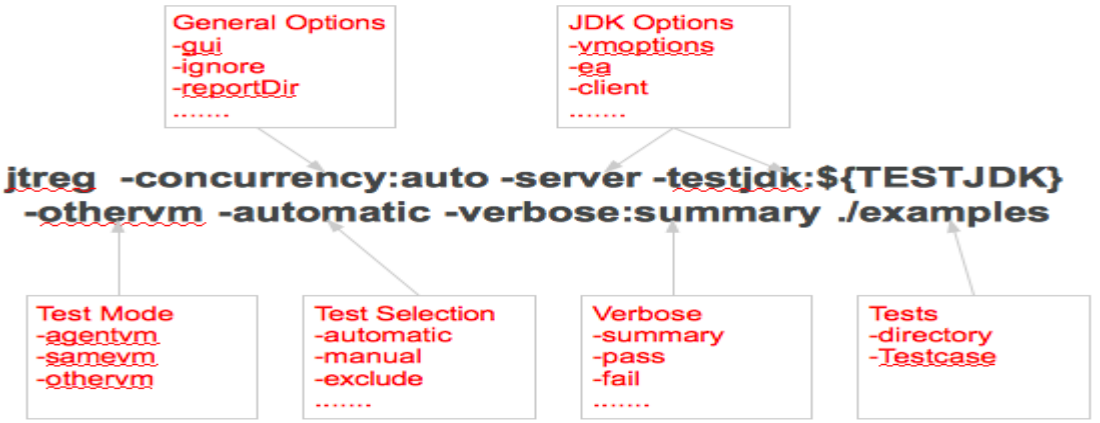
- Verbose options
  - amount of output written to the console while running tests.
  - -verbose:summary, -verbose:fail,error, ..
- Test selection options
  - -a | -automatic can be used to run automatic tests
  - -m | -manual can be used to run manual tests
  - -k:<keywordExpr> can be used to run only tests with <keywordExpr> which is a simple boolean expression containing keywords
  - -bug:<bugid> can be used to run only tests with <bugid>
  - All tests run by default

3.Test Execution (2/4)



- Test Mode
  - -othervm : Run every test in its own JVM; maximum isolation between tests; maximum overhead
  - -samevm : Run each test in the same JVM as the JavaTest harness; minimum overhead; minimum isolation between tests; one bad test can cause problems for all subsequent tests
  - -agentvm : Run tests using a pool of reusable JVMs; reuse vm if it can be reset to a clean state after a test; cannot be enforced in a test
  - Test mode can also be specified in a test source file e.g. @run main/agentvm, @run main/othervm/timeout=300
- Tests can be run concurrently using -conc:*N* | -concurrency:*N* option
  - This option can not be used with -samevm option
  - Not suitable for GUI tests e.g. a test require mouse interaction

3.Test Execution (3/4)



- JDK options
  - -vmoption can be used to pass all applicable JVM options
  - Most common JVM options, e.g. -server, -d64, -client etc. are supported directly
- Test JDK
  - -testjdk can be used to pass JDK version for testing
  - JDK version used for running 'jtreg' harness is tested by default
- Excluding testcases
  - ProblemList.txt can contain known failing tests.
  - -exclude:ProblemList.txt excludes tests present in the files from executing.

3. Test Execution (4/4)

- Test log is written to a text file with .jtr extension.
  - Contains the command used to execute tests
  - Contains diagnostic message included in exception
  - Contains output of System.out and System.err
- Report and log files are written to JTreport and JTwork directory, by default
  - JTwork directory contains .jtr files and .class files.
  - JTreport directory contains html summary report and summary.txt
  - -r and -w options can be used to specify non-default work and report directory
- -xml option can be used to produce test log files in xml format
  - Log files with .jtr.xml extension are found in JTwork directory

Section I: Getting started

1. Introduction
2. Nuts and Bolts
3. Test Execution
4. An Example

4. An Example (1/5)

```
/* @test
   @summary Example1 - a simple testcase
   @run main Example1
*/
import javax.swing.*;
import javax.accessibility.*;

public class Example1 {
   public static void main(String[] args) throws Exception {
        JLayer l = new JLayer();
        AccessibleContext acc = l.getAccessibleContext();
        if (acc == null) {
            throw new RuntimeException("JLayer's AccessibleContext is null");
        }
    }
}
```

Indicates source is a test

4. An Example (2/5)

```
 /* @test
   @summary Example1 - a simple testcase
   @run main Example1
*/
import javax.swing.*;
import javax.accessibility.*;

public class Example1 {
   public static void main(String[] args) throws Exception {
        JLayer l = new JLayer();
        AccessibleContext acc = l.getAccessibleContext();
        if (acc == null) {
            throw new RuntimeException("JLayer's AccessibleContext is null");
        }
    }
}
```

- **Brief description of the test.**
- **Appear in output .jtr file**

4. An Example (3/5)

```
 /* @test
   @summary Example1 - a simple testcase
   @run main Example1
*/
import javax.swing.*;
import javax.accessibility.*;

public class Example1 {
   public static void main(String[] args) throws Exception {
        JLayer l = new JLayer();
        AccessibleContext acc = l.getAccessibleContext();
        if (acc == null) {
            throw new RuntimeException("JLayer's AccessibleContext is null");
        }
    }
}
```

- **It means invoke main method of Example1 class**
- **The line is optional for default execution.**

4. An Example (4/5)

```
 /* @test
   @summary Example1 - a simple testcase
   @run main Example1
*/
import javax.swing.*;
import javax.accessibility.*;

public class Example1 {
   public static void main(String[] args) throws Exception {
       JLayer l = new JLayer();
       AccessibleContext acc = l.getAccessibleContext();
       if (acc == null) {
          throw new RuntimeException("JLayer's AccessibleContext is null");
       }
    }
}
```

- **Throw an exception to indicate test failure.**
- **Test should exit normally to indicate PASS.**

4. Running an Example (5/5)

- jtreg -verbose:summary -testjdk:{TESTJDK} Example1.java
  - An empty file TEST.ROOT should exist at the root directory of the test.
  - Creates JTwork and JTreport directory, if the directory is not present.
- JTreport/html/report.html contains the overall results summary
- JTreport/text/summary.txt contains the results summary in text format
- JTwork/Example1.jtr contains test log file

Section I: Getting started

1. Introduction
2. Nuts and Bolts
3. Test Execution
4. An Example
5. Log file

5. Log file (1/3)

```
#Test Results (version 2)
#Tue Nov 20 16:53:27 GMT 2012
#checksum:45cf21a8ef8efa
#-----testdescription-----
$file=/home/jtest/jtregtests/examples/Example1.java
$root=/home/jtest/jtregtests
run=USER_SPECIFIED main Example1\n
source=Example1.java
title=Example1 - a simple testcase

#-----environment-----

#-----testresult-----
description=file\:/home/jtest/jtregtests/examples/Example1.java
elapsed=928 0\:00\:00.928
end=Tue Nov 20 16\:53\:27 GMT 2012
environment=regtest
execStatus=Passed. Execution successful
hostname=testmachine
javatestOS=Linux 2.6.32-100.28.5.el6.x86_64 (amd64)
javatestVersion=4.4
jtregVersion=jtreg 4.1 fcs b04
script=com.sun.javatest.regtest.RegressionScript
sections=script_messages build compile main
start=Tue Nov 20 16\:53\:26 GMT 2012
test=examples/Example1.java
user.name=jtest
work=/home/jtest/jtregtests/JTwork/examples
```

Test information

Log file has an extension .jtr and can be found in sub-directories of JTwork directory

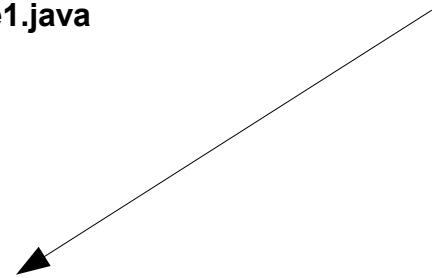
5. Log file (2/3)

```
#Test Results (version 2)
#Tue Nov 20 16:53:27 GMT 2012
#checksum:45cf21a8ef8efa
#-----testdescription-----
$file=/home/jtest/jtregtests/examples/Example1.java
$root=/home/jtest/jtregtests
run=USER_SPECIFIED main Example1\n
source=Example1.java
title=Example1 - a simple testcase

#-----environment-----

#-----testresult-----
description=file\:/home/jtest/jtregtests/examples/Example1.java
elapsed=928 0\:00\:00.928
end=Tue Nov 20 16\:53\:27 GMT 2012
environment=regtest
execStatus=Passed. Execution successful
hostname=testmachine
javatestOS=Linux 2.6.32-100.28.5.el6.x86_64 (amd64)
javatestVersion=4.4
jtregVersion=jtreg 4.1 fcs b04
script=com.sun.javatest.regtest.RegressionScript
sections=script_messages build compile main
start=Tue Nov 20 16\:53\:26 GMT 2012
test=examples/Example1.java
user.name=jtest
work=/home/jtest/jtregtests/JTwork/examples
```

Test result summary



5. Log file (3/3)

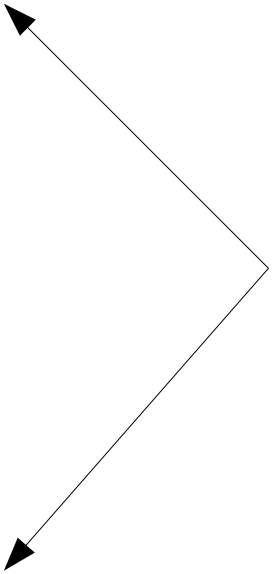
```
#section:script_messages
----------messages:(4/184)----------
JDK under test: (/home/jtest/jdks/jdk1.8.0)
java version "1.8.0-ea"
Java(TM) SE Runtime Environment (build 1.8.0-ea-b59)
Java HotSpot(TM) 64-Bit Server VM (build 25.0-b02, mixed mode)

#section:build
----------messages:(3/93)----------
command: build Example1
reason: Named class compiled on demand
elapsed time (seconds): 0.635
result: Passed. Build successful

#section:compile
----------messages:(3/143)----------
command: compile /home/jtest/jtregtests/examples/Example1.java
reason: .class file out of date or does not exist
elapsed time (seconds): 0.631
----------System.out:(0/0)----------
----------System.err:(0/0)----------
result: Passed. Compilation successful

#section:main
----------messages:(3/102)----------
command: main Example1
reason: User specified action: run main Example1
elapsed time (seconds): 0.22
----------System.out:(0/0)----------
----------System.err:(1/15)----------
STATUS:Passed.
result: Passed. Execution successful


test result: Passed. Execution successful
```



Detail test results broken down into different phases of test execution

Section I: Getting started

1. Introduction
2. Nuts and Bolts
3. Test Execution
4. An Example
5. Log file

Section II: Examples

1. Examples

1. Examples

- Specifying VM Options
- Applet testcase
- Manual testcase
- Shell test
- Defining timeout
- Reusing the testcase to run with different VM/GC options
- Compiler test
- Negative test
- OS/Platform checks

1. Examples (1/9): Specifying VM options

```
 /* @test
 * @summary Example 2 - passing VM parameters
 * @run main/othervm -Xms100m -Xmx100m  -XX:+PrintGC -XX:
+UseParallelGC Example2
 */
public class Example2 {
    public static void main(String[] args) throws InterruptedException {
        ……..
         for (GarbageCollectorMXBean collector : collectors) {
            String x = collector.getName();
            if (x.contains("MarkSweep")) {
               long count = collector.getCollectionCount();
                if (count > 0) {
                    throw new RuntimeException("Failed.”);
                }
            }
        }
    }
}
```

- **VM options can be specified with main/othervm**
- **Equivalent to java <vm options> Example2**

1. Examples (2/9) : Applet testcase

```
<html>
<!--
  @test
  @summary - Example3 - running applet
  @run applet Example3.html
  -->
<head>
<title> Example3 </title>
</head>
<body>

<h1>Example3</h1>

<APPLET CODE="Example3.class" WIDTH=200
HEIGHT=200></APPLET>
</body>
</html>
```

- Run the applet specified at <APPLET> tag in Example3.html
- Equivalent to “appletviewer Example3.html”

1. Examples (3/9): Manual testcase

```
<html>
<!--
  @test
  @summary - Example4 - a manual test
  @run applet/manual=yesno Example4.html
  -->
<head>
<title> Example4 - a manual test</title>
</head>
<body>

<h1> Example4 - a manual test</h1>


<APPLET CODE="Example4.class" WIDTH=200
HEIGHT=200></APPLET>
</body>
</html>
```

- **Run the applet specified at <APPLET> tag in Example4.html, but tester need to interact and choose whether test is pass or fail.**
- **NOTE: OpenJDK test workspace provides convenient helper classes to display test instruction.**

1. Examples (4/9): Shell test

```
# @test
# @summary Example5 - shell tests
# @run shell Example5.sh

$TESTJAVA/bin/jar -tf "$TESTJAVA/jre/lib/rt.jar" >
/dev/null 2>&1

if [ $? -ne 0 ]; then
   echo FAILED: rt.jar file corrupt
   exit 1
fi

exit 0
```

- **Invokes the Bourne shell to run Example5.sh**

1. Examples (5/9) : Defining timeout

```
/*
 * @test
 * @summary Example6 - interrupting loop/hang
 * @run main/timeout=1 Example6
 */


public class Example6 {
    public static void main(String[] args) throws
InterruptedException {
        System.out.print("Test Loop .... ");
        for (int i = 0; i < 1000000000; i++) {
            Thread.sleep(10);
        }
        System.out.println("Passed.");
    }
}
```

- **Invoke main method with timeout set in seconds.**
- **Test exit with Error if the execution takes longer than timeout period.**
- **Useful option to avoid test hangs.**

1. Examples (6/9) : Reusing the testcase to run with different VM/GC options

```
/* @test
 * @summary Example7 - run tests with multiple java/vm flags
 * @run main/othervm -client  Example7
 * @run main/othervm -Xint Example7
 * @run main/othervm -server Example7
 */
public class Example7 {
    public static void main(String argv[]) {
        Integer x = new Integer(100);
        Integer y = new Integer(100);

        if ( ! x.equals(y) ) {
            throw new RuntimeException("Failed. x is not equal y");
        }

        if (x.hashCode() != y.hashCode()) {
            throw new RuntimeException("Failed. Hashcode comparison failed.");
        }
    }
}
```

- **Invoke main method multiple times with different VM options.**
- **Generates one output file (.jtr)**
- **Test fails if any one invocation of main method fails.**

1. Examples (7/9) : Compiler test

```
/*
 * @test
 * @summary Example8 - compile test
 * @compile Example8.java
 */

public class Example8 {

    static class Base<E> {}

    static void test(Base<?> je) {
        Object o = (Base<Integer>)je;
    }
}
```

- **Invoke compiler and compiles Example8.java**
- **Test pass if compilation succeeds.**
- **Equivalent to “javac Example8.java”**

1. Examples (8/9) : Negative test

```
/*
 * @test
 * @summary Example9 - test pass if compilation fails
 * @compile/fail Example9.java
 */

public class Example9 {
    void test(Float f, Integer i) {
        boolean b = f == i;
    }
}
```

- **Test pass if compilation fails**
- **The option /fail can be used with main, applet and shell tests as well, but such use is discouraged**
  - **@run main/fail <test>**
  - **@run applet/fail <test>**
  - **@run shell/fail <test>**

1. Examples (9/9) : OS/Platform checks

```
# @test
# @summary Testing OS specific feature
# @run shell Example12.sh

OS=`uname -s`
if [ "$OS" != "Windows_NT" ]; then
    echo "This is a Windows only test"
    exit 0
fi

echo “run a test case”
```

- Write a wrapper shell test
- Check OS name
- Skip the test (pass) if the test is not applicable
- May use 'case' statement if test is applicable to more than one OS.
- NOTE: Such testcase filtering is not allowed if the feature is applicable to all OS/platforms.

Section I: Getting started

1. Introduction
2. Nuts and Bolts
3. Test Execution
4. An Example
5. Log file

Section II: Examples

1. Examples
2. Test Execution

2. Test Execution (1/3)

- VM option from command line
  - jtreg -verbose:summary -testjdk:${TESTJDK} -ea -vmoption:"-XX:+UseParallelGC" Example1.java
- Run all tests present in a directory
  - jtreg -verbose:summary -testjdk:${TESTJDK} ./examples
- Run all manual tests in a directory
  - jtreg -verbose:summary -testjdk:${TESTJDK} -m ./examples
- Run all tests with specific keyword
  - jtreg -verbose:summary -testjdk:${TESTJDK} -a -k:abc ./examples
  - TEST.ROOT file should contain a line “keys abc”

2. Test Execution (2/3)

- Run all automated tests present in a directory
  - jtreg -verbose:summary -testjdk:${TESTJDK} -a ./examples
- Running tests in concurrent mode
  - jtreg -verbose:summary -testjdk:${TESTJDK} -a -concurrency:auto ./examples
  - jtreg -verbose:summary -testjdk:${TESTJDK} -a -concurrency:2 ./examples
  - Tests with /othervm can be run in concurrent mode
- Increase time requirements to run a test
  - jtreg -verbose:summary -testjdk:${TESTJDK} -timeoutFactor:2 Example13.java

2. Test Execution (3/3)

- Excluding tests with known issues
  - jtreg -verbose:summary -testjdk:${TESTJDK} -a -exclude:./ProblemList.txt ./examples
- Re-running only the tests with error status in previous tun
  - jtreg -verbose:summary -testjdk:${TESTJDK} -a -status:error,fail ./examples
- There are many more options. Refer online help.
  - jtreg -h
  - jtreg -onlineHelp

Section I: Getting started

1. Introduction
2. Nuts and Bolts
3. Test Execution
4. An Example
5. Log file

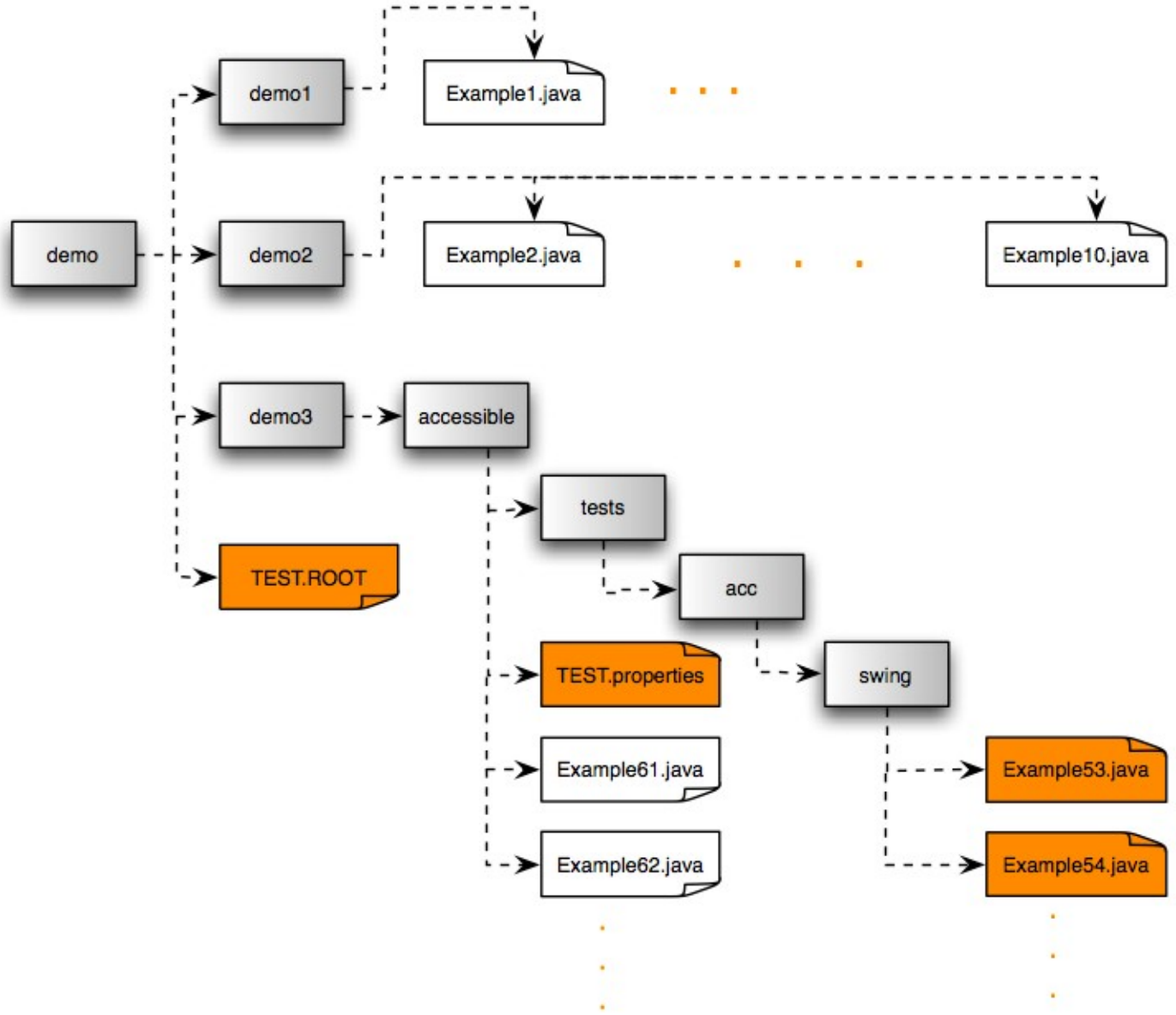
Section II: Examples

1. Examples
2. Test Execution
3. TestNG tests

3. TestNG tests

- JTreg supports TestNG tests in two ways – Type A and Type B
- Type A:
    - Write a testcase using TestNG
    - Add “TestNG.dirs = dir1 dir2” in TEST.properties or TEST.ROOT file, where, dir1 and dir2 represents a package root directory relative test root (TEST.ROOT) directory
    - TEST.properties file, if present, should be present at package root directory
    - While an individual testcase may include jtreg information tags such as @test or @summary, it must not include jtreg action tags such as @run or @compile
- Type B:
    - Write a testcase using TestNG
    - Add jtreg tags (e.g. @test) to the test source
    - Use “@run testng ..” tag instead of “@run main ..” tag

3. TestNG tests – Type A (1/2)



TEST.properties
TestNG.dirs = .

OR

TEST.ROOT
TestNG.dirs = accessible

package tests.acc.swing

3. TestNG tests – Type A (2/2)

tests/acc/swing/Example53.java

```
package tests.acc.swing;
import javax.swing.*;
import javax.accessibility.*;
import org.testng.annotations.Test;

@Test
public class Example53 {
    public void testAccessibleJMenu() throws Exception {
        JMenu l = new JMenu();
        AccessibleContext acc = l.getAccessibleContext();
        if (acc == null) {
            throw new RuntimeException("JMenu's AccessibleContext is null");
        }
    }
    public void testAccessibleJMenuBar() throws Exception {
        JMenuBar l = new JMenuBar();
        AccessibleContext acc = l.getAccessibleContext();

            if (acc == null) {

            throw new RuntimeException("JMenuBar's  AccessibleContext is null");
        }
    }
}
```

tests/acc/swing/Example54.java

```
package tests.acc.swing;
import javax.swing.*;
import javax.accessibility.*;
import org.testng.annotations.Test;

public class Example54 {
    @Test
    public void testAccessibleJMenuItem() throws Exception {
        JMenuItem l = new JMenuItem();
        AccessibleContext acc = l.getAccessibleContext();
        if (acc == null) {
            throw new RuntimeException("JMenuItem's AccessibleContext is null");
        }
    }
    @Test
    public void testAccessibleJFileChooser() throws Exception {
        JFileChooser l = new JFileChooser();
        AccessibleContext acc = l.getAccessibleContext();
        if (acc == null) {
            throw new RuntimeException("JFileChooser's AccessibleContext is
null");
        }
    }
}
```

No jtreg specific tags in the test source files

3. TestNG tests – Type B

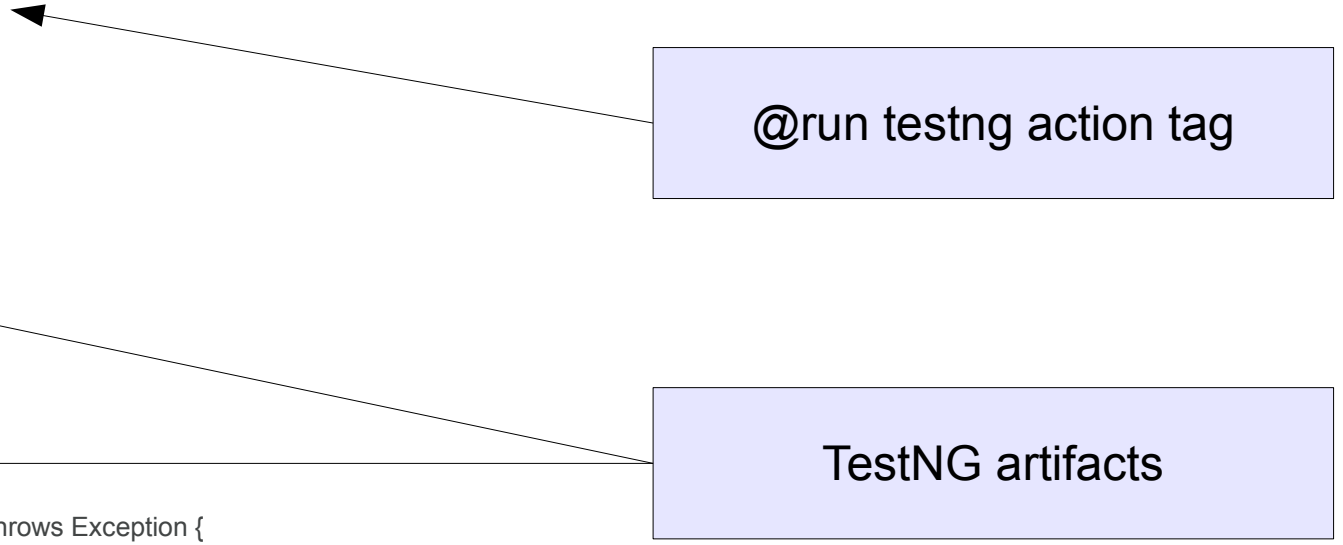
```
/* @test
   @summary Example61 - a simple TestNG testcase that can be run under jtreg
   @run  testng Example61
*/

import javax.swing.*;
import javax.accessibility.*;

import org.testng.annotations.Test;

public class Example61 {

    @Test
    public void testAccessibleContext() throws Exception {
        JLayer l = new JLayer();
        AccessibleContext acc = l.getAccessibleContext();
        if (acc == null) {
            throw new RuntimeException("JLayer's AccessibleContext is null");
        }
    }
}
```



3. TestNG tests – specifying libraries

- Specifying libraries in “Type 1” tests
  - Add “lib.dirs = dir1 dir2” to the TEST.properties file
  - If path starts with '/', the path is evaluated from the directory containing TEST.ROOT file, else the path is evaluated from the directory containing TEST.properties file
- Specifying libraries in “Type 2” tests
  - Use @library tag <path>, it is same as in other jtreg tests
  - If path starts with '/', the path is evaluated from the directory containing TEST.ROOT file, else the path is evaluated from the directory relative to the testcase

Section I: Getting started

1. Introduction
2. Nuts and Bolts
3. Test Execution
4. An Example
5. Log file

Section II: Examples

1. Examples
2. Test Execution
3. TestNG tests

Section III: Contributing

1. How to contribute?

1. How to Contribute?

- Guidelines
  - http://openjdk.java.net/contribute/
- Subscribe to jtreg-use@openjdk.java.net
  - Ask general questions about jtreg use
- Subscribe to jtreg-dev@openjdk.java.net and
  - Ask tough questions!
  - Send enhancement proposals
- Subscribe to quality-discuss@openjdk.java.net
  - Discuss quality issues
  - Propose test cases, send change sets, send feedback etc.

Section I: Getting started

1. Introduction
2. Nuts and Bolts
3. Test Execution
4. An Example
5. Log file

Section II: Examples

1. Examples
2. Test Execution
3. TestNG tests

Section III: Contributing

1. How to contribute?
2. Recommended Practice

2. Recommended Practice

- Modifying an existing testcase
  - Do not overwrite the test scenarios, unless existing test scenario is obsolete with a new fix
- Test case should run on Windows, Linux, and Solaris.
  - No Exception
  - If testcase is not applicable to a OS/platform, make the testcase PASS.
- Code review
  - Recommend two peer reviews
  - Send email to OpenJDK component mailing list for feedback

Section I: Getting started

1. Introduction
2. Nuts and Bolts
3. Test Execution
4. An Example
5. Log file

Section II: Examples

1. Examples
2. Test Execution
3. TestNG tests

Section III: Contributing

1. How to contribute?
2. Recommended Practice
3. OpenJDK workspace

3. OpenJDK workspace (1/3)

- Cloning OpenJDK workspace
  - $ hg clone http://hg.openjdk.java.net/jdk8/jdk8 localbuild
  - $ cd localbuild
  - $ sh get_source.sh
- Test Root
  - ${localbuild}/jdk/test
  - ${localbuild}/langtools/test
  - ${localbuild}/hotspot/test
- ProblemList.txt present in “Test Root”

3. OpenJDK workspace (2/3)

- A convenient 'Makefile' present in “Test Root”
  - Export environment varaibles
  - make CONCURRENCY=auto <target>
- Environment Variables

```
$ export JT_HOME=<location of jtreg>
$ export JTREG=${JT_HOME}/<linux,solaris,win32>/bin/jtreg
$ export PRODUCT_HOME=<location of test jdk>
$ export ANT_HOME=<location of ant>
$ export PATH=${PATH}:${ANT_HOME}/bin
$ export ALT_OUTPUTDIR=<results home dir>
$ export JPRT_JTREG_HOME=${JT_HOME}     // Required for langtools and hotspot
$ export JPRT_JAVA_HOME=<location of stable jdk>  // Required for langtools
$ export JTREG_TESTDIRS=<location of langtools test> // Required for langtools
$ export TESTDIRS="compiler gc runtime serviceability"     // Required for hotspot
```

3. OpenJDK workspace (3/3)

- Executing langtool and hotspot tests
  - Export environment variables
  - make CONCURRENCY=auto jtreg-tests
- Executing jdk tests
  - Export environment variables
  - make CONCURRENCY=auto jdk_nio1
  - make CONCURRENCY=auto jdk_nio2
  - make CONCURRENCY=auto jdk_lang
  - make CONCURRENCY=auto jdk_net
  - ……..
  - make CONCURRENCY=auto jdk_all // Runs all jdk tests.

Section I: Getting started

1. Introduction
2. Nuts and Bolts
3. Test Execution
4. An Example
5. Log file

Section II: Examples

1. Examples
2. Test Execution
3. TestNG tests

Section III: Contributing

1. How to contribute?
2. Recommended Practice
3. OpenJDK workspace
4. References

References

- http://openjdk.java.net/jtreg/
- http://openjdk.java.net/projects/code-tools/
- http://mail.openjdk.java.net/mailman/listinfo/jtreg-dev
- jtreg-dev@openjdk.java.net
- FAQ included in jtreg bundle
- tag-spec.txt included in jtreg bundle
- jtreg -onlineHelp <word>
- jtreg -help <word>