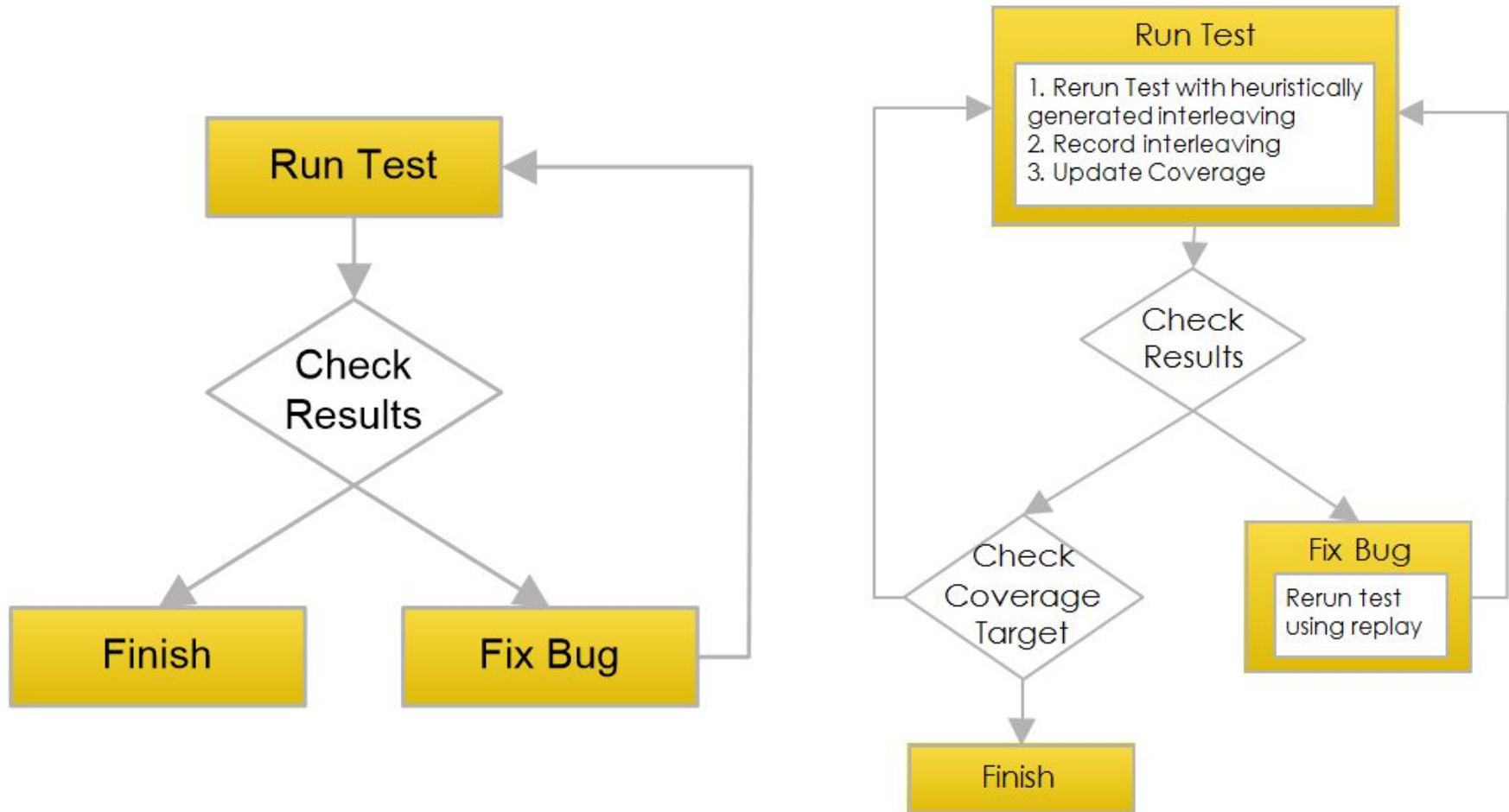

SyncTest

Alexander Marshall, Honors Thesis Student
Supervisor: Dr. Jeremy Bradbury
April 8th, 2016

Motivation

- Unit testing is a common practice for testing computer programs
 - The most common tool for unit testing Java code is JUnit, which runs a set of tests in a random or specified order
- For multithreaded code, there are several tools for finding concurrency related bugs
 - SyncDebugger: automatically debugs multithreaded code by cutting the target area in half each iteration
 - Eclipticon: an Eclipse plugin designed to help optimize concurrent software by creating thread delays
 - ConTest: developed by IBM, it forces race conditions and deadlocks to occur by altering the execution order of threads.
- Both are great, but they aren't designed to work together

Traditional Testing vs. IBM ContTest

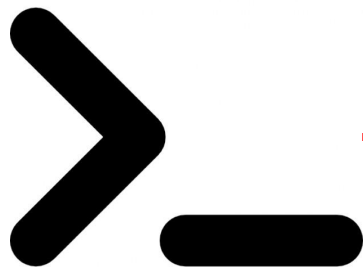


Proposal - SyncTest

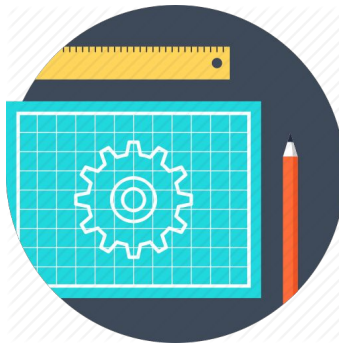
- An Eclipse plugin for unit testing multithreaded code
- The plugin should be able to:
 - Run tests in repetition
 - Track test passes, failures, errors, and deadlocks
 - Check for deadlocked threads and kill them
 - Visually represent results
- To instrument code, we can use SyncDebugger



Evolution of SyncTest



Bash Script

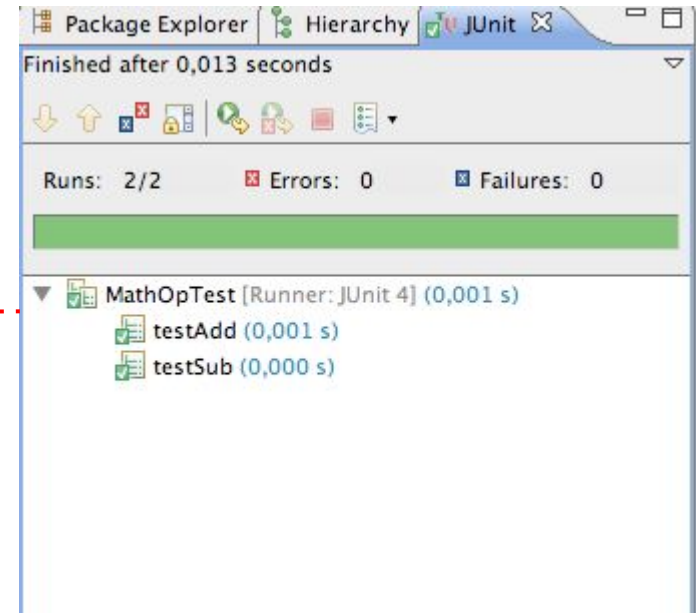
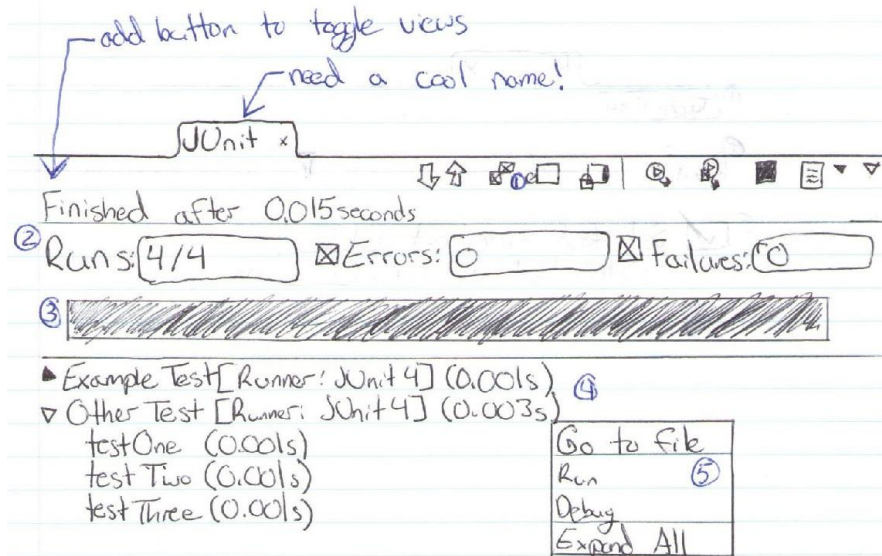


Prototype

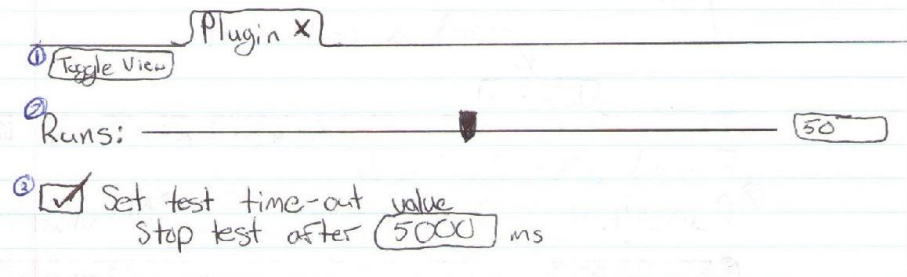


Eclipse Plugin

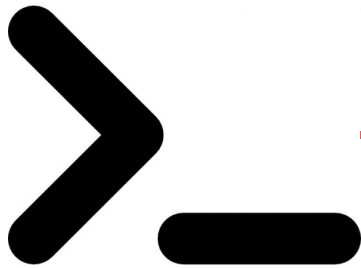
Early Sketches



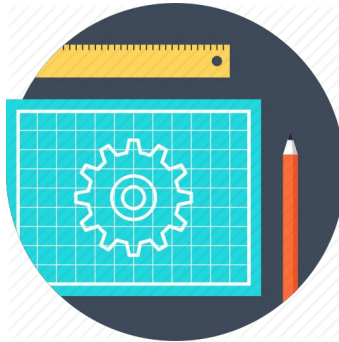
- ① Show failed or skipped tests only
 - ↳ Add a button to show tests that timed-out
 - ② Shows number of tests run, errors, and failures
 - ↳ Should also show number of runs per test as well as the number of tests that timed out
 - ③ Shows a fully green or red bar (pass/fail)
 - ↳ Change to show as pass, error, fail, time-out in proportion to each other
 - ↳ Add a second bar to show similar stats for the multiple runs of a second selected test
 - ④ Shows JUnit version and elapsed time
 - ↳ add fraction or percentage of passed tests
 - ⑤ Allows you to re-run a test
 - ↳ add sub-menu to re-run 1, 10, 100, or n times
- Other: Allow user to set time-out value and a total percentage to pass threshold



Evolution of SyncTest

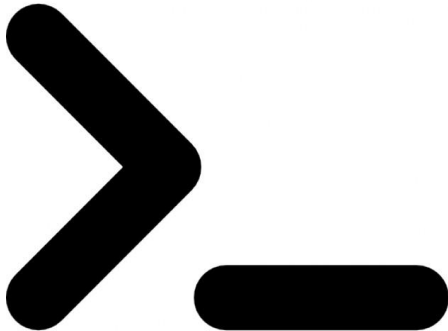


Bash Script



Bash Script

- Started out as a script to run a JUnit test in repetition
- Now it does much more:
 - Takes paths to directories and run count as inputs
 - Runs all tests the specified amount of times
 - Uses a second script to find and kill deadlocks
 - Saves all output from tests
 - A separate Java class parses the output and prints results
- For testing purposes, instrumentation was done separately through SyncDebugger



Case Study:

Bank Account System

```
alex@NormandySR-2 syncTest$ sh syncTest.sh account/src account/tests account/out 3
```

```
-----VARS-----
```

```
PATH_TO_SRC: account/src  
PATH_TO_TST: account/tests  
PATH_TO_OUT: account/out  
LOOP_COUNT : 3  
-----
```

```
Running Test100...
```

```
Execution 1: Passed
```

```
Execution 2: Passed
```

```
Execution 3: Passed
```

```
Running Test10...
```

```
Execution 1: Passed
```

```
Execution 2: Passed
```

```
Execution 3: Failed
```

```
Running Test10K...
```

```
Execution 1: Passed
```

```
Execution 2: Passed
```

```
Execution 3: Passed
```

```
Running Test15K...
```

```
Execution 1: Passed
```

```
Execution 2: Passed
```

```
Execution 3: Passed
```

```
Running Test1K...
```

```
Execution 1: Passed
```

```
Execution 2: Passed
```

```
Execution 3: Passed
```

```
Running Test2...
```

```
Execution 1: Passed
```

```
Execution 2: Passed
```

```
Execution 3: Passed
```

```
Running Test50K...
```

```
Execution 1: Passed
```

```
Execution 2: Passed
```

```
Execution 3: Failed
```

```
Running Test5K...
```

```
Execution 1: Passed
```

```
Execution 2: Passed
```

```
Execution 3: Passed
```

```
=====Test100=====
```

```
Tests Run: 3
```

```
Passed: 3
```

```
Failed: 0
```

```
Deadlocked: 0
```

```
=====Test10=====
```

```
Tests Run: 3
```

```
Passed: 2
```

```
Failed: 1
```

```
Deadlocked: 0
```

```
=====Test10K=====
```

```
Tests Run: 3
```

```
Passed: 3
```

```
Failed: 0
```

```
Deadlocked: 0
```

```
=====Test15K=====
```

```
Tests Run: 3
```

```
Passed: 3
```

```
Failed: 0
```

```
Deadlocked: 0
```

```
=====Test1K=====
```

```
Tests Run: 3
```

```
Passed: 3
```

```
Failed: 0
```

```
Deadlocked: 0
```

```
=====Test2=====
```

```
Tests Run: 3
```

```
Passed: 3
```

```
Failed: 0
```

```
Deadlocked: 0
```

```
=====Test50K=====
```

```
Tests Run: 3
```

```
Passed: 2
```

```
Failed: 1
```

```
Deadlocked: 0
```

```
=====Test5K=====
```

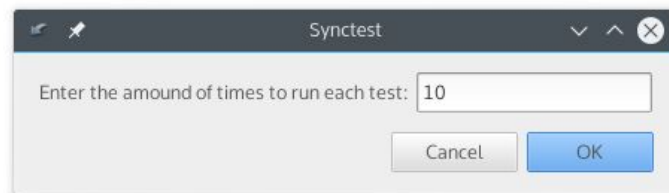
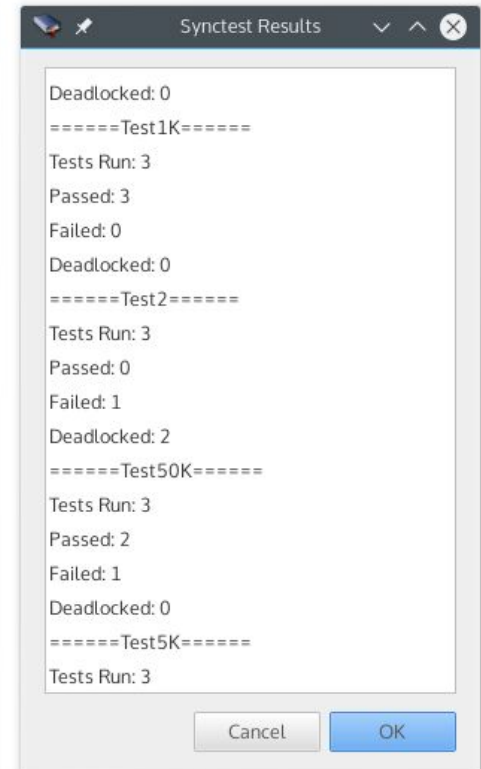
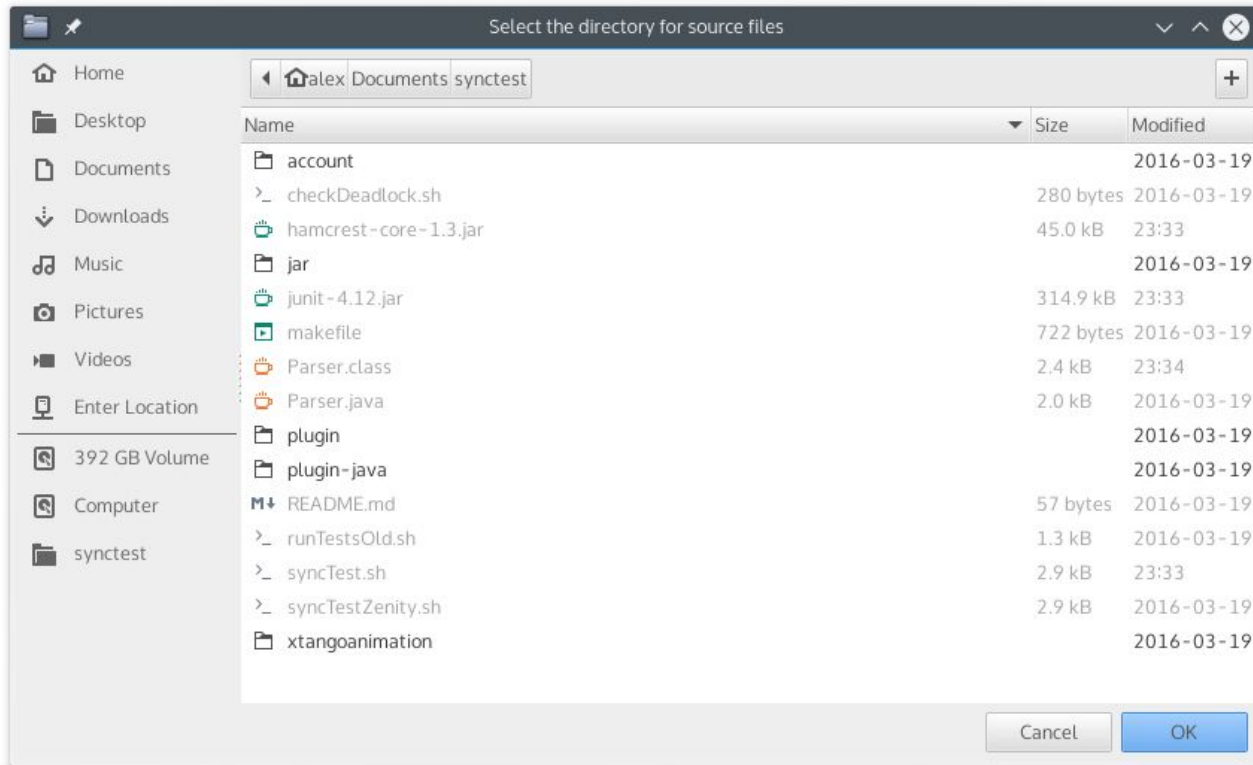
```
Tests Run: 3
```

```
Passed: 3
```

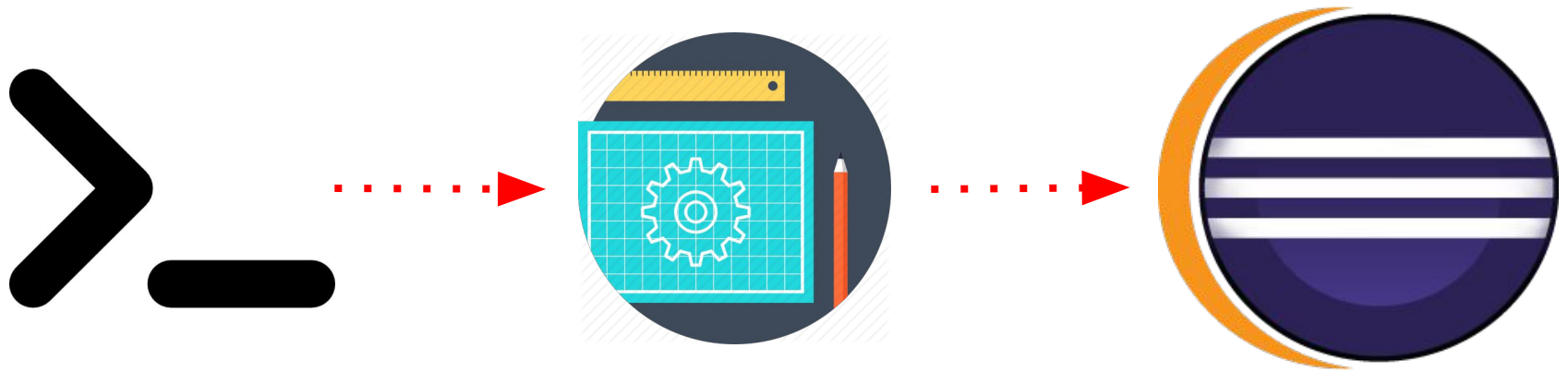
```
Failed: 0
```

```
Deadlocked: 0
```

On The Side... Zenity

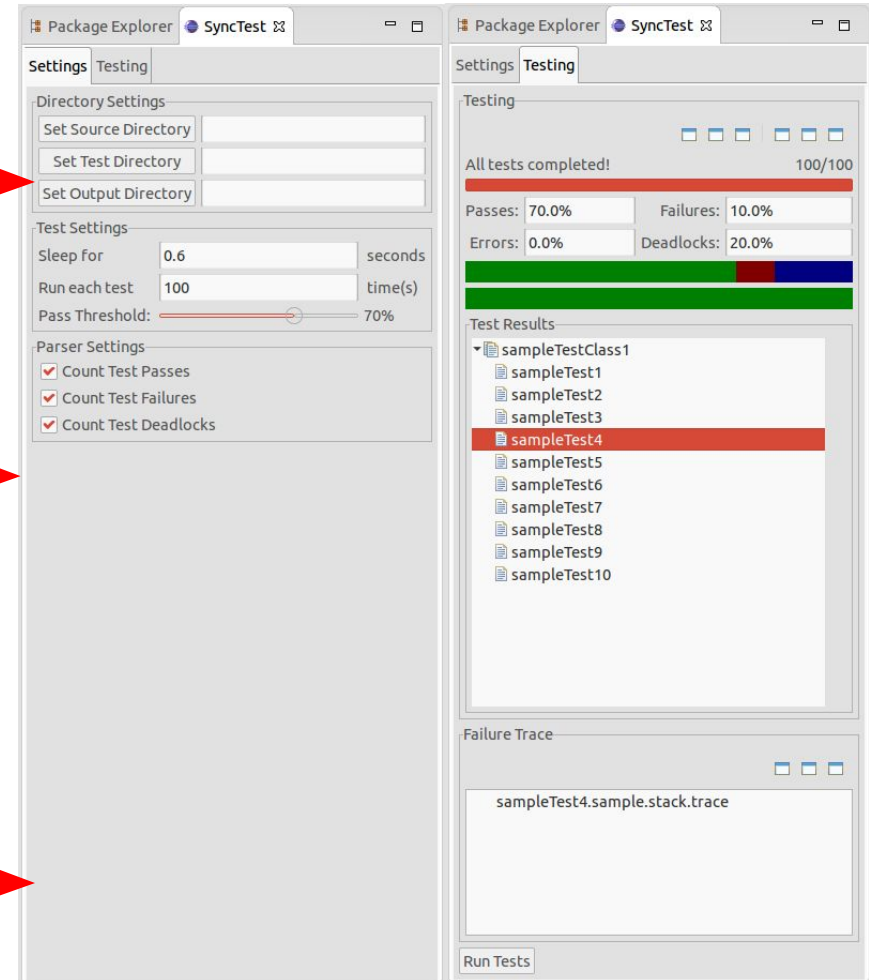
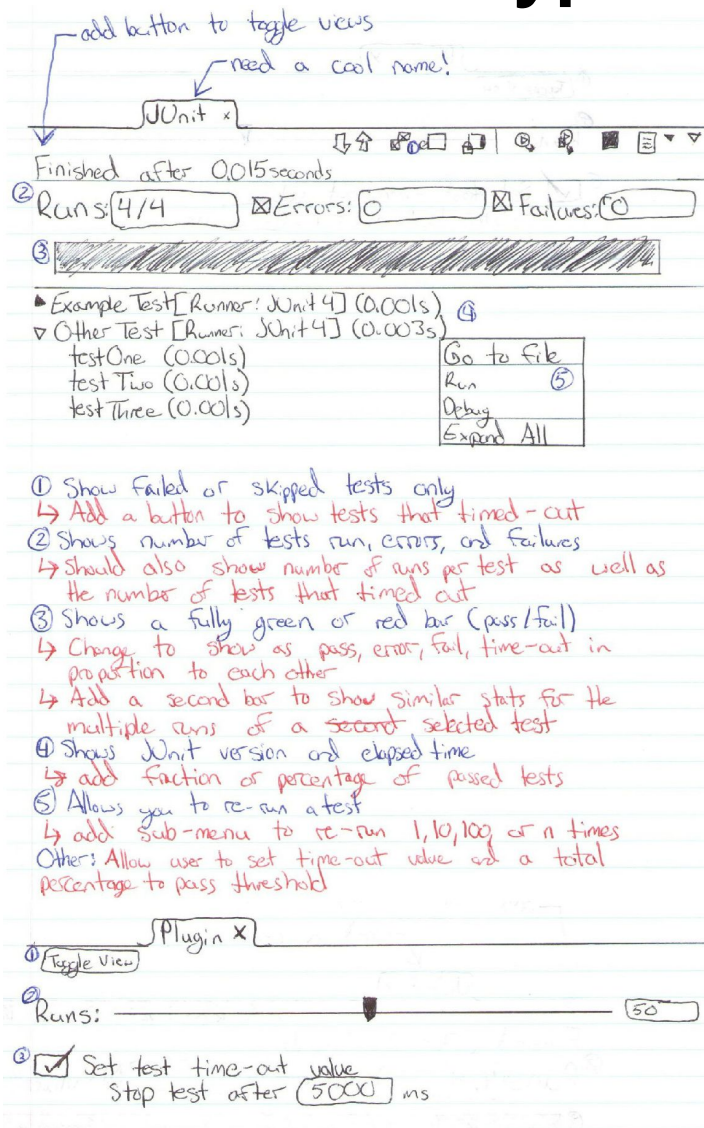


Evolution of SyncTest



Prototype

Visual Prototype



Package Explorer

SyncTest

Settings

Testing

Directory Settings

Set Source Directory

Set Test Directory

Set Output Directory

Test Settings

Sleep for

0.6

seconds

Run each test

100

time(s)

Pass Threshold:

70%

Parser Settings

☒ Count Test Passes

☒ Count Test Failures

☒ Count Test Deadlocks

Package Explorer SyncTest

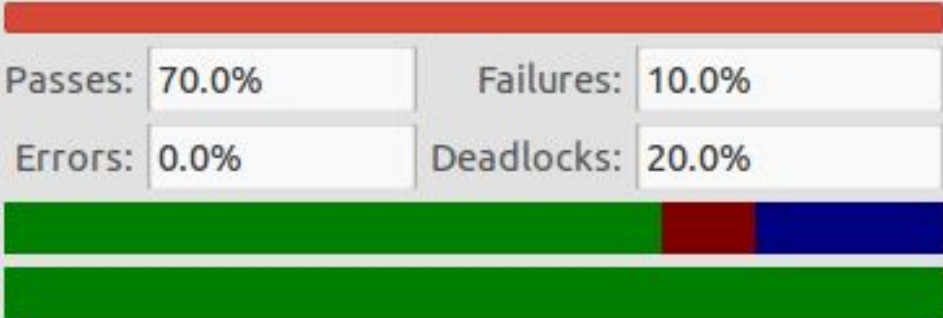
Settings Testing

Testing

All tests completed! 100/100

Passes: 70.0% Failures: 10.0%

Errors: 0.0% Deadlocks: 20.0%



The image shows a testing interface with two progress bars. The top bar is red and represents the overall completion status, which is 100%. The bottom bar is green and represents the pass rate, which is 70.0%. The bar is divided into three segments: green (70%), red (10%), and blue (20%).

Test Results

- sampleTestClass1
 - sampleTest1
 - sampleTest2
 - sampleTest3
 - sampleTest4**
 - sampleTest5
 - sampleTest6
 - sampleTest7
 - sampleTest8
 - sampleTest9
 - sampleTest10

Failure Trace

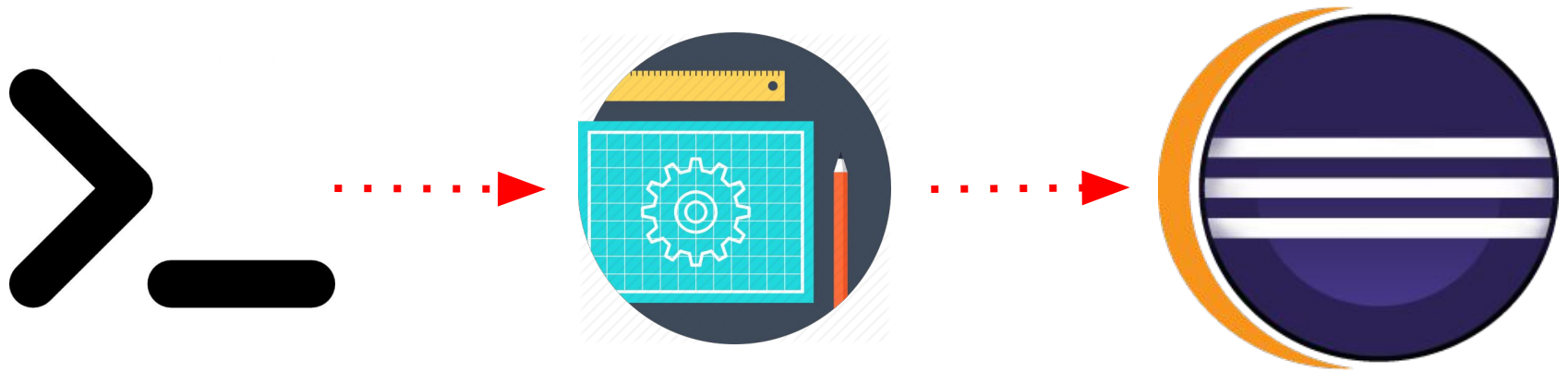
sampleTest4.sample.stack.trace

Run Tests

Functional Prototype

- Runs a modified version of the bash script
- Eclipse plugin provides inputs
- Outputs are used to update plugin UI
- Bash script limits OS

Evolution of SyncTest



Eclipse Plugin

Case Study:

Bank Account System Redux

Settings Testing

Directory Settings

Select the project base directory



☒ Automatically find source and test directories

Select the directory containing source code



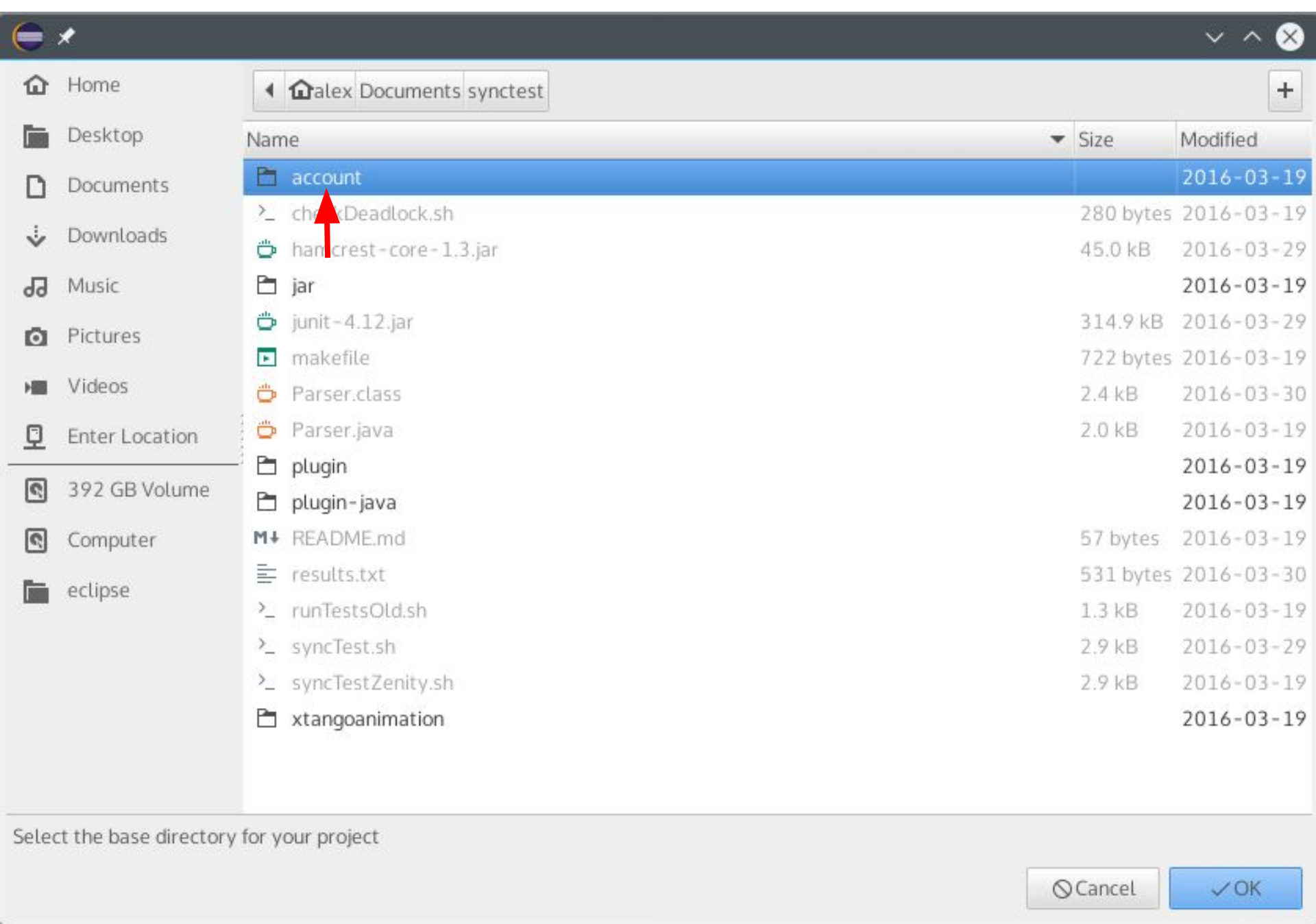
Select the directory containing junit tests



Test Settings

Check for deadlocks every seconds

Run each test time(s)



Settings Testing

Directory Settings

/home/alex/Documents/synctest/account



☒ Automatically find source and test directories

/home/alex/Documents/synctest/account/src



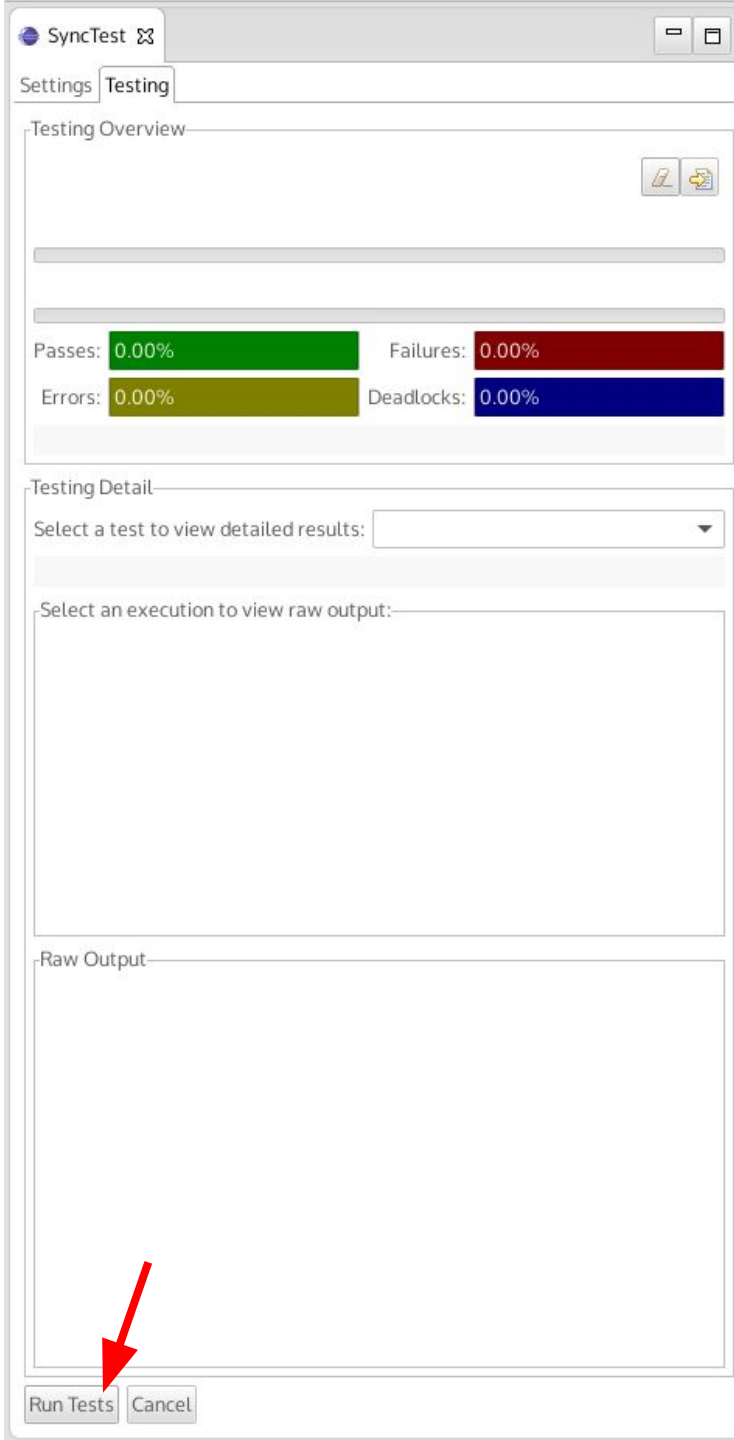
/home/alex/Documents/synctest/account/tests



Test Settings

Check for deadlocks every seconds

Run each test time(s)



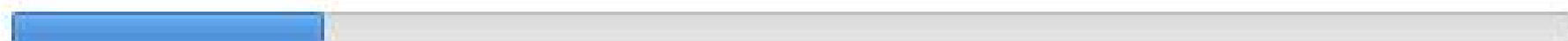
Testing Overview



Running: Test10K



Execution 2/10:



Passes: 90.91%

Failures: 9.09%

Errors: 0.00%

Deadlocks: 0.00%



Testing Overview

Running: Test2

Execution 3/10:

Passes: 88.68%

Failures: 11.32%

Errors: 0.00%

Deadlocks: 0.00%

Testing Detail

Select a test to view detailed results: Test10

Select an execution to view raw output:

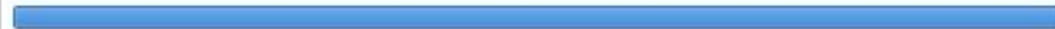
Execution 1: pass
Execution 2: pass
Execution 3: pass
Execution 4: pass
Execution 5: pass
Execution 6: pass
Execution 7: pass
Execution 8: pass

Settings Testing

Testing Overview



All Tests Completed



All Executions Completed



Passes: 90.00%

Failures: 8.75%

Errors: 0.00%

Deadlocks: 1.25%



Testing Detail

Select a test to view detailed results: Test10



Select

Pass: 9, Fail: 1, Error: 0, Deadlock: 0

Execution 1: pass
Execution 2: pass
Execution 3: pass
Execution 4: pass
Execution 5: pass
Execution 6: pass
Execution 7: pass
Execution 8: pass

Testing Detail

Select a test to view detailed results: Test10 ▼



Select an execution to view raw output:

Execution 3: pass
Execution 4: pass
Execution 5: pass
Execution 6: pass
Execution 7: pass
Execution 8: pass
Execution 9: fail
Execution 10: pass



Raw Output

```
JUnit version 4.12
.Running program with 10 accounts.
<There is amount with less than 300, No Lock>
E
Time: 0.01
There was 1 failure:
1) test10(account.tests.Test10)
java.lang.AssertionError
    at org.junit.Assert.fail(Assert.java:86)
    at org.junit.Assert.assertTrue(Assert.java:41)
    at org.junit.Assert.assertTrue(Assert.java:52)
    at account.tests.Test10.test10(Test10.java:12)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
```

Run Tests Cancel

Testing Overview



All Tests Completed



All Executions Completed



Passes: 90.00%



Failures: 7/80



Errors: 0.00%



Deadlocks: 1/80



Testing Overview



export results to text file

All Tests Completed



All Executions Completed



Passes: 90.00%



Failures: 7/80

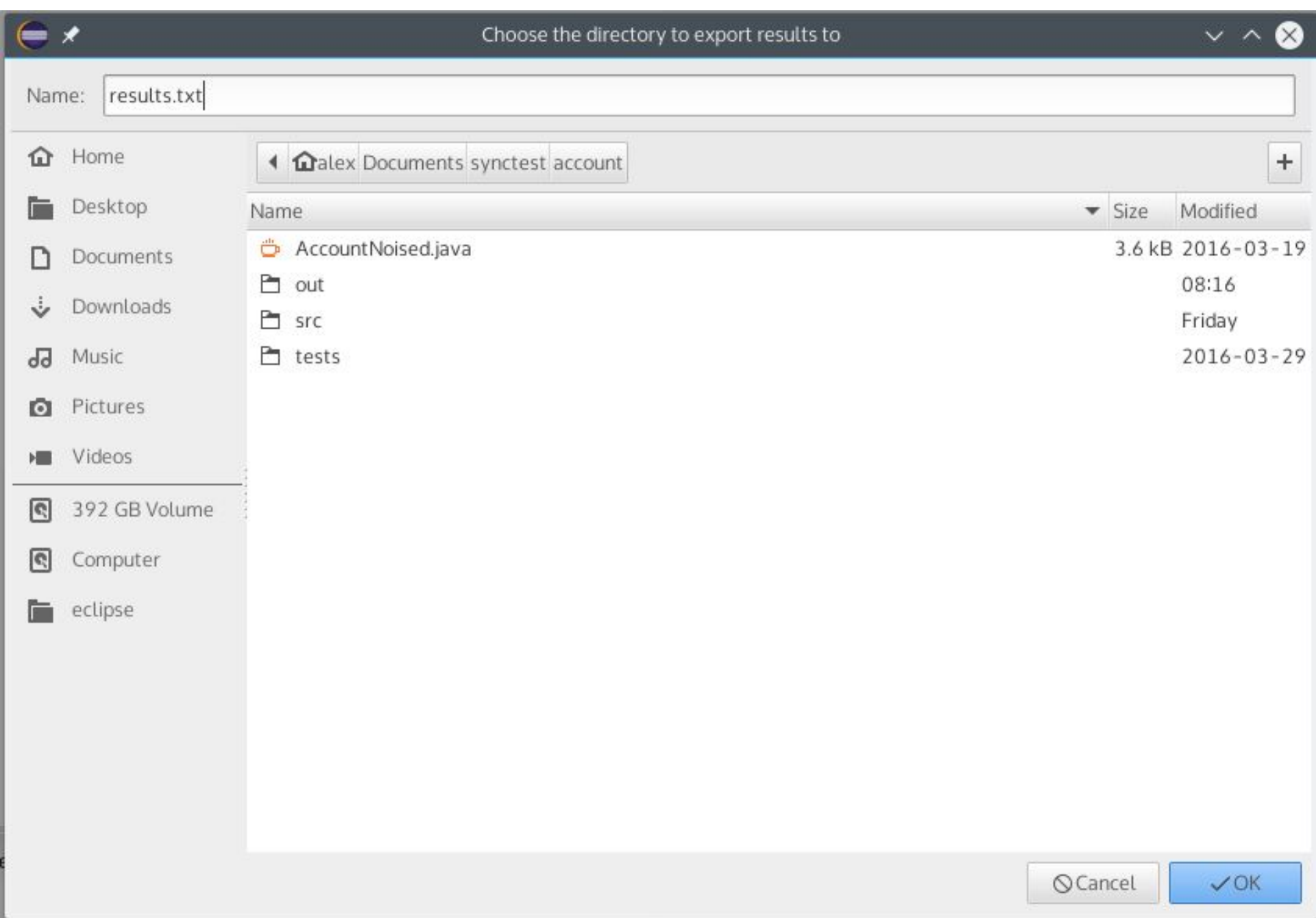


Errors: 0.00%



Deadlocks: 1/80





results.txt — Kate

File Edit View Projects Bookmarks Sessions Tools Settings Help

results.txt

```
Test: Test100
-----
Pass: 9
Fail: 1
Error: 0
Deadlock: 0
Total: 10

Test: Test10
-----
Pass: 9
Fail: 1
Error: 0
Deadlock: 0
Total: 10

Test: Test10K
-----
Pass: 9
Fail: 1
Error: 0
Deadlock: 0
Total: 10

Test: Test15K
-----
Pass: 8
Fail: 2
Error: 0
Deadlock: 0
Total: 10

Test: Test1K
-----
Pass: 9
Fail: 1
Error: 0
```

Line 1, Column 1

INSERT Soft Tabs: 4 (8) UTF-8 Normal

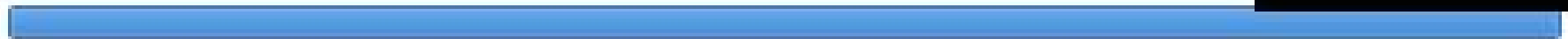
Search and Replace Current Project

Testing Overview



clear all fields

All Tests Completed



All Executions Completed



Passes: 90.00%

Failures: 7/80

Errors: 0.00%

Deadlocks: 1/80



SyncTest

SettingsTesting

Testing Overview

All Tests Completed

All Executions Completed

Passes: 0.00%

Failures: 000/000

Errors: 0.00%

Deadlocks: 000/000

Testing Detail

Select a test to view detailed results:

Select an execution to view raw output:

Raw Output

Run Tests

Cancel

Limitations

- Bash script: Linux and Mac OS
- Project structure
- Scalability

Future Work

- SyncDebugger Integration
- Import functionality from JUnit
 - Re-run tests
 - Sorting options
 - Test history
- Testing

