Introduction to Alpaca and MCUT

Alpaca: A lovely parallel and concurrency analyzer
MCUT: Microsoft Concurrency Unit Testing Framework

Unit Testing

- What is a Unit Test?

Create an Alpaca/MCUT Project

1. Open Visual Studio 2010
2. Create a new C# Class Library project
   – Example name: AlpacaProject
3. Add a project reference to MCUT Framework:
   – Microsoft.Concurrency.UnitTestingFramework
   – This is installed with Alpaca and should be visible in the Add Reference dialog box in VS

Create a test class

- A test class is just any class that contains test methods
- Requirements:
  - Public
  - Non-static
  - Public, empty constructor
    - In C#, if no explicit constructor is specified, an empty ctor is automatically generated for the class.

Create your first Unit Test

1. Import the namespace:
   `using Microsoft.Concurrency.TestTools.UnitTesting;`
2. Create a test method
   – Requirements: public, non-static
   – Return value is ignored
   – For this example, don’t use any arguments
3. Mark the method as a unit test:
   `[UnitTestMethod]`
4. Add test code body

Hello World Test Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Concurrency.TestTools.UnitTesting;

namespace AlpacaProject1
{
    public class Class1
    {
        [UnitTestMethod]
        public void HelloWorldTest()
        {
            Console.WriteLine("Hello world!");
            Console.Error.WriteLine("Oops, this is an error.");
        }
    }
}
```

Build

Prepare to use Alpaca

- Why?
  - Alpaca creates separate task folders for each test run it performs
  - Can clutter up the folder alpaca is running in

To Prepare:

1. Create a temporary folder.
   E.g. alpaca.tmp
2. Run alpaca from this folder

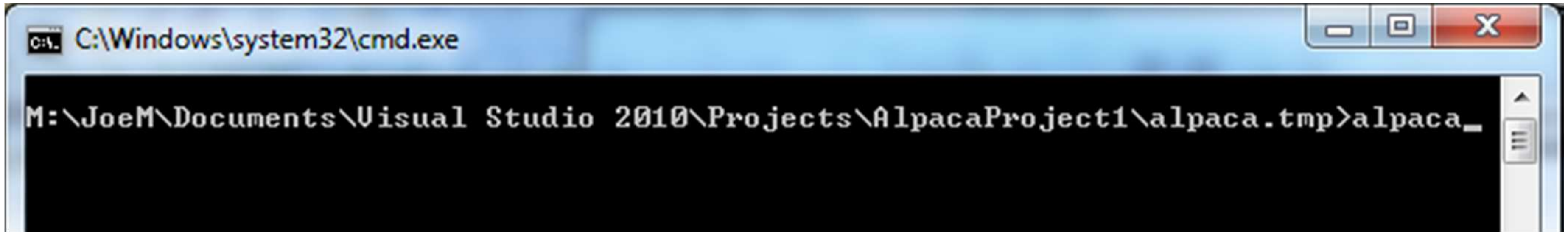
Open Alpaca

1. Open a command prompt and CD into your temporary folder

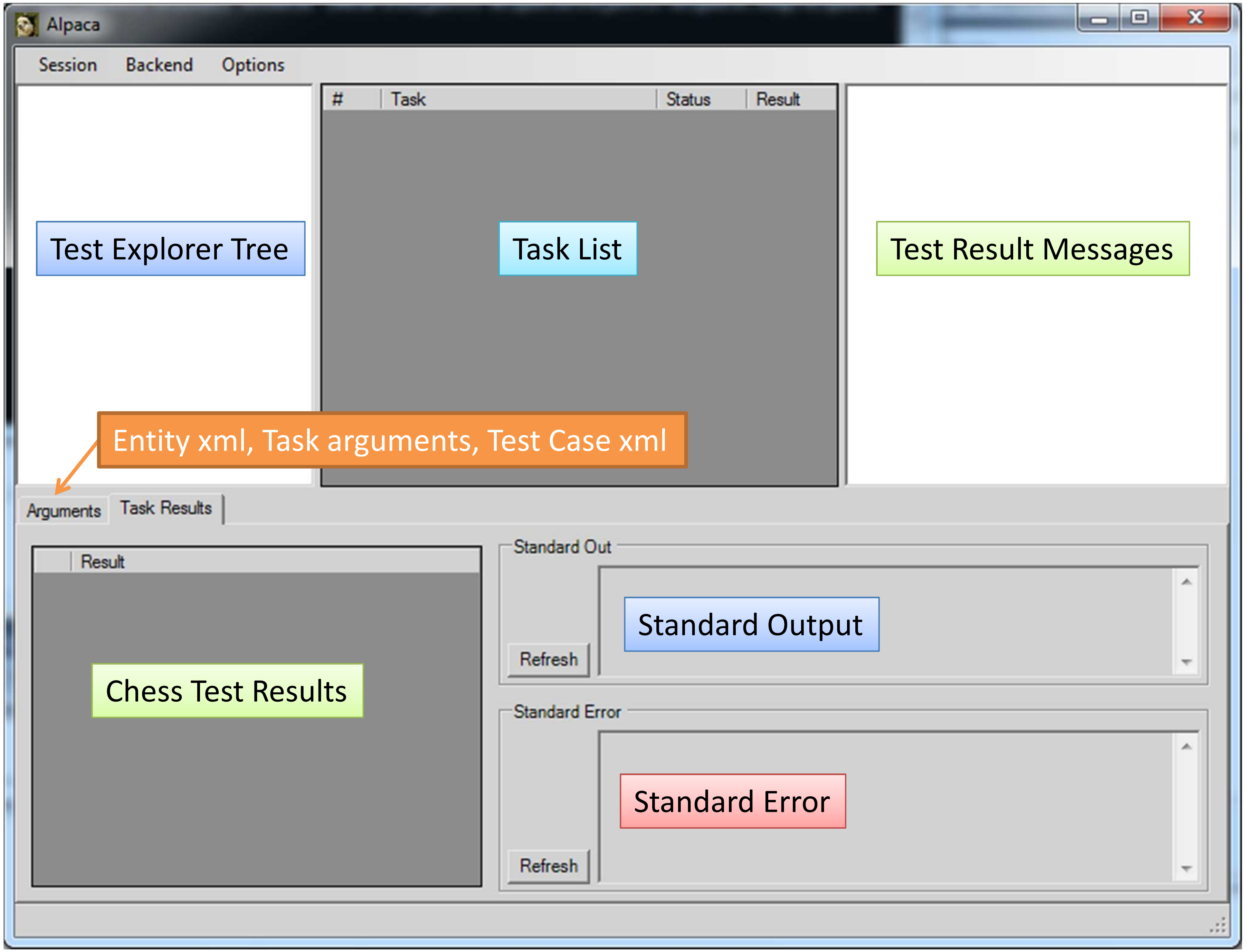
   Alternate method (Win 7)

   a) Shift+right-click on/in your temporary folder
   b) Click “Open command window here”

2. Open alpaca by typing alpaca



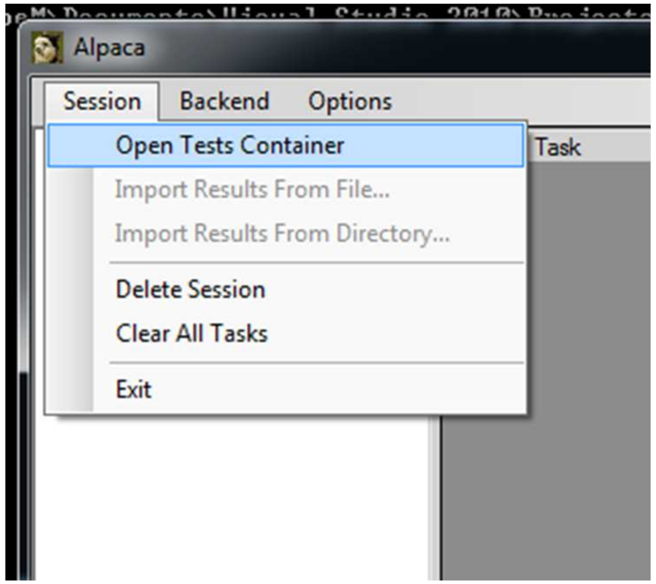
```
M:\JoeM\Documents\Visual Studio 2010\Projects\AlpacaProject1\alpaca.tmp>alpaca_
```

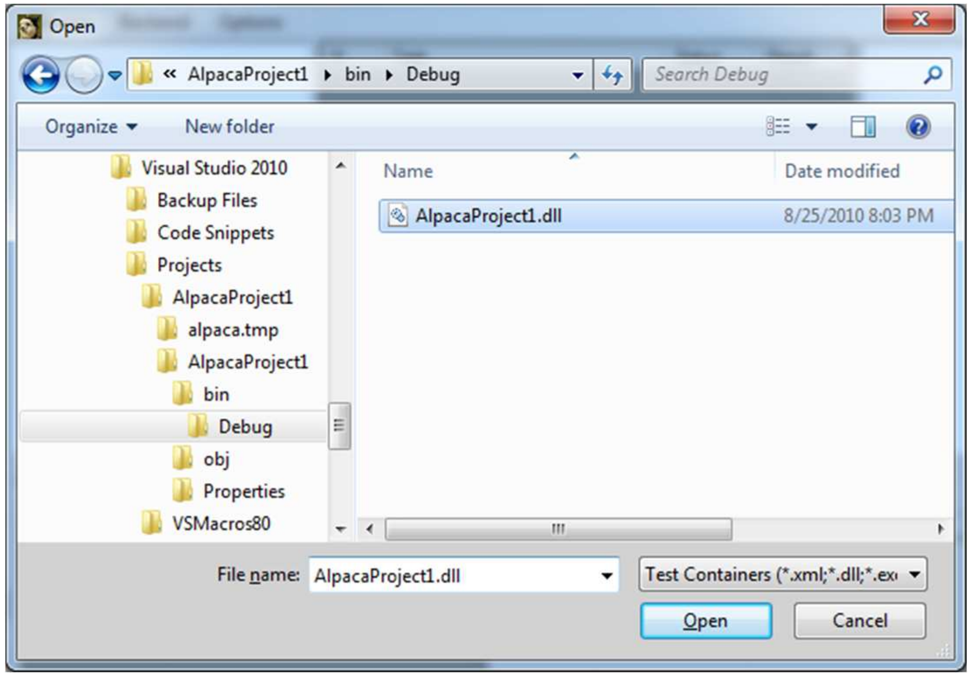

Alpaca
Session
Backend
Options
#
Task
Status
Result
Test Explorer Tree
Task List
Test Result Messages
Entity xml, Task arguments, Test Case xml
Arguments
Task Results
Result
Chess Test Results
Standard Out
Refresh
Standard Output
Standard Error
Refresh
Standard Error

Open Test Assembly

1. Session -> Open Tests Container
2. Navigate to the dll that you built for your alpaca project

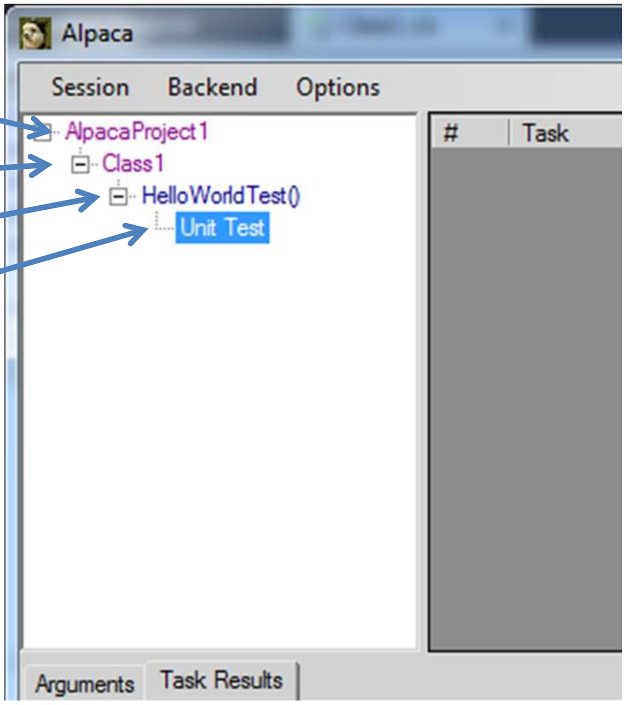
Note: If you don't see your assembly make sure it's been built first.





Test Assembly Tree

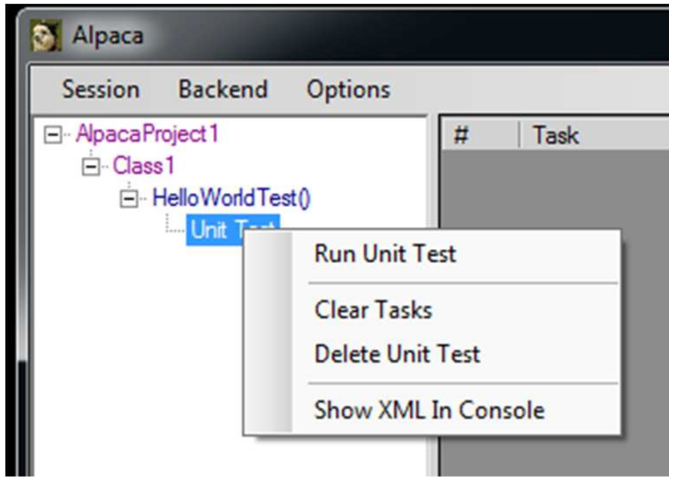
- Test Assembly Container
- Test Class Name
- Test Method
- Test Type(s)
  - There may be more than one type of test specified for a test method



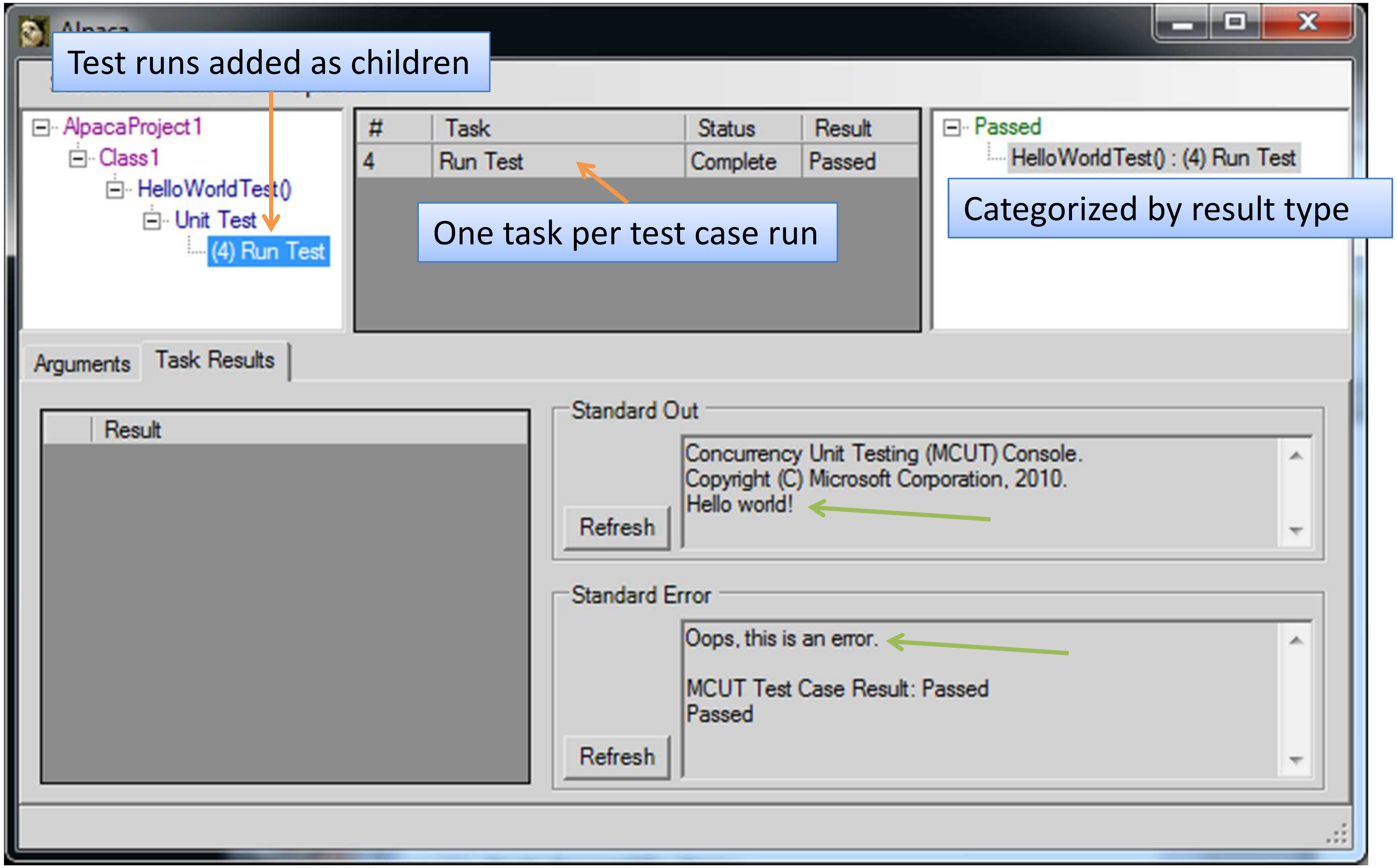
Run a Test

1. Right-click on the test method or the "Unit Test" test type node
2. Select the "Run" or "Run Unit Test" menu option

A task will be created in the tasks list.



Test Results and Output



Test runs added as children

One task per test case run

Categorized by result type

#	Task	Status	Result
4	Run Test	Complete	Passed

```
Concurrency Unit Testing (MCUT) Console.
Copyright (C) Microsoft Corporation, 2010.
Hello world!
```

```
Oops, this is an error.

MCUT Test Case Result: Passed
Passed
```

Test Assertions

- Assert class
  - Static class containing methods that will assert conditions.
- Examples

```
Assert.IsTrue(a != null);
Assert.IsNull(a);
Assert.AreEqual(10, n);
```

- When an assertion fails, an exception is thrown and Alpaca is notified

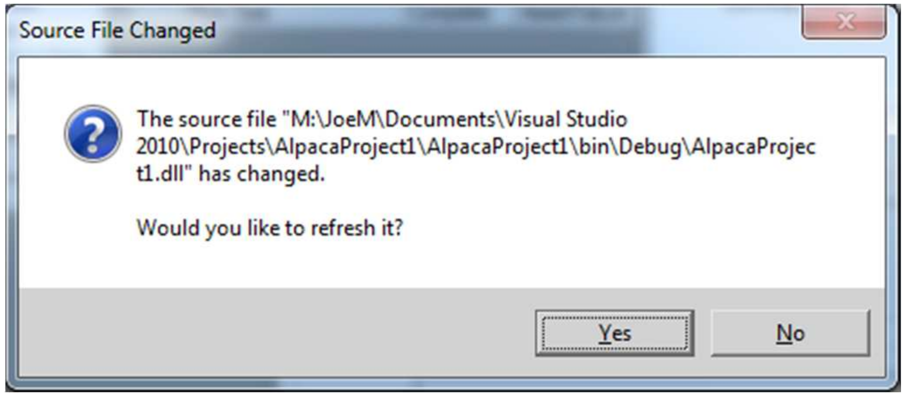
SortArrayTest

```
[UnitTestMethod]
public void SortArrayTest()
{
   int n = 1000;
   Random rand = new Random();

   // Create array of random integers
   int[] a = new int[n];
   for (int i = 0; i < n; i++)
      a[i] = rand.Next();

   // Sort the array
   SortArray(a);

   // Verify Correctness
   for (int i = 1; i < n; i++)
      Assert.IsTrue(a[i] >= a[i - 1]
         , "Element {0} is not sorted correctly", i);
}
```

```
private void SortArray(int[] a)
{
   //Array.Sort(a);
}
```

Refresh Assembly

- After adding the new test method, rebuild and switch over to Alpaca


Source File Changed

The source file "M:\JoeM\Documents\Visual Studio 2010\Projects\AlpacaProject1\AlpacaProject1\bin\Debug\AlpacaProject1.dll" has changed.

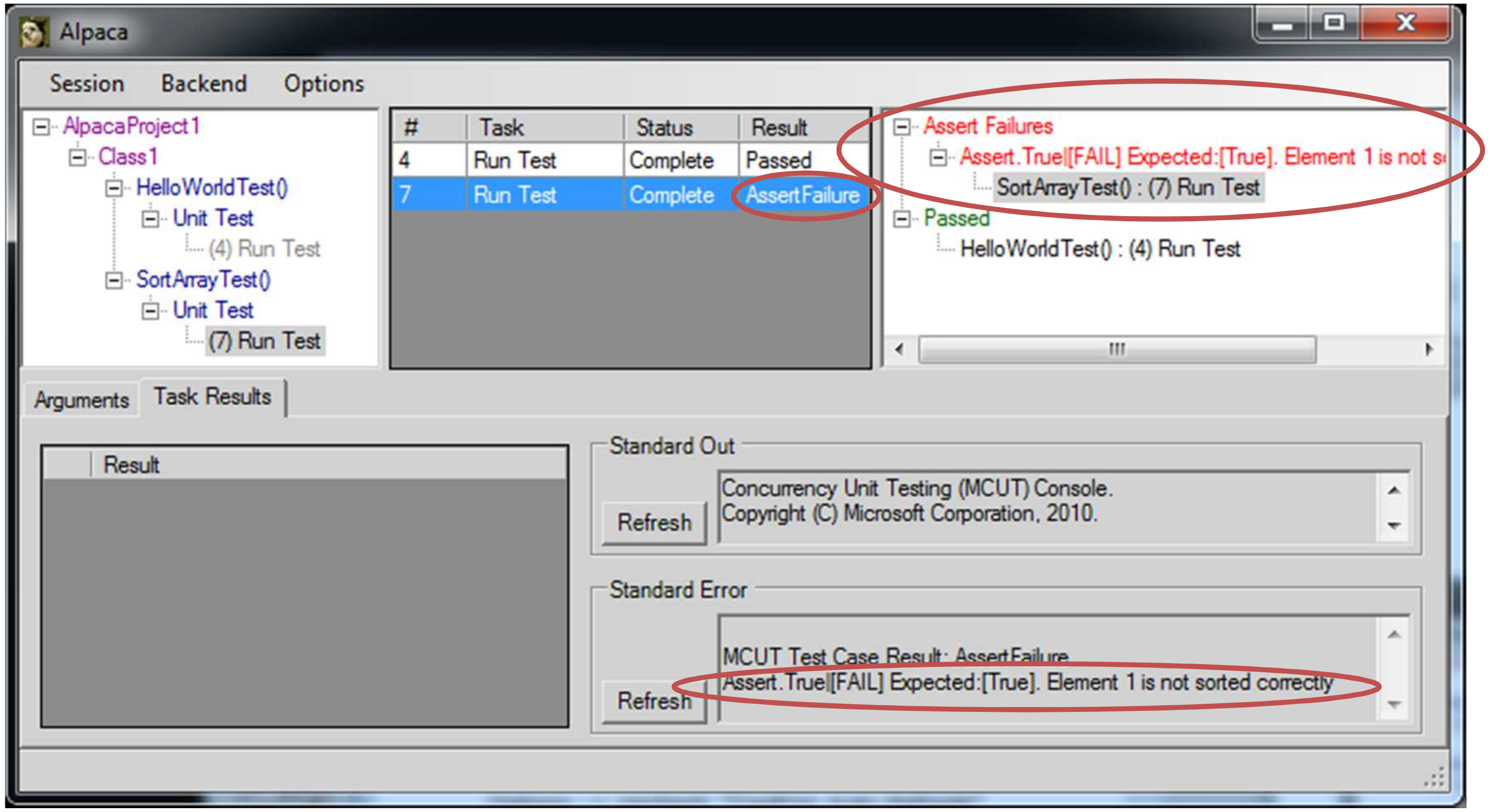
Would you like to refresh it?

- Alpaca watches your test assembly for changes
- To disable the notification:

  Options -> Uncheck “Confirm Auto Refresh”

Run Failing Test

- Run the SortArrayTest



#	Task	Status	Result
4	Run Test	Complete	Passed
7	Run Test	Complete	AssertFailure

```
Assert Failures
  Assert.True|[FAIL] Expected:[True]. Element 1 is not s
    SortArrayTest() : (7) Run Test
Passed
  HelloWorldTest() : (4) Run Test
```

Standard Out:

```
Concurrency Unit Testing (MCUT) Console.
Copyright (C) Microsoft Corporation, 2010.
```

Standard Error:

```
MCUT Test Case Result: AssertFailure
Assert.True|[FAIL] Expected:[True]. Element 1 is not sorted correctly
```

View Exception Stack Trace

- Currently, no way to debug into a unit test in VS from Alpaca ☹
- But we can see the exception with it's stack trace:
  1. Click/select on the result node in the results tree
  2. View the "Arguments" tab to see the xml
  3. Inside the xml is an error element that tells you the exceptionType and stackTrace

Fix the Bug

- Uncomment the SortArray function's body

```
private void SortArray(int[] a)
{
    Array.Sort(a);
}
```

- Rebuild
- Rerun the test
- Test passes

Performance Tests

- Measure time taken to perform a certain block of code
- Intervals displayed in a graph
- TaskMeter
  - Provides Start/Stop methods for specifying the block of code to time

Create a Performance Test

1. Create TaskMeter instance(s)

```
TaskMeter initMeter = new TaskMeter("Initialize Data");
TaskMeter sortingMeter = new TaskMeter("Sort");
TaskMeter verificationMeter = new TaskMeter("Verify");
```

2. Mark our SortArrayTest method to also be a Performance test: `[PerformanceTestMethod]`
3. Add meter Start/Stop calls around blocks of code you wish to time

SortArrayTest

```
TaskMeter initMeter = new TaskMeter("Initialize Data");
TaskMeter sortingMeter = new TaskMeter("Sort");
TaskMeter verificationMeter = new TaskMeter("Verify");

[UnitTestMethod]
[PerformanceTestMethod]
public void SortArrayTest()
{
    int n = 1000;
    Random rand = new Random();

    // Create array of random integers
    initMeter.Start();
    int[] a = new int[n];
    for (int i = 0; i < n; i++)
        a[i] = rand.Next();
    initMeter.End();

    // Sort the array
    sortingMeter.Start();
    SortArray(a);
    sortingMeter.End();

    // Verify Correctness
    verificationMeter.Start();
    try
    {
        for (int i = 1; i < n; i++)
            Assert.IsTrue(a[i] >= a[i - 1]
                , "Element {0} is not sorted correctly", i);
    }
    finally
    {
        verificationMeter.End();
    }
}
```

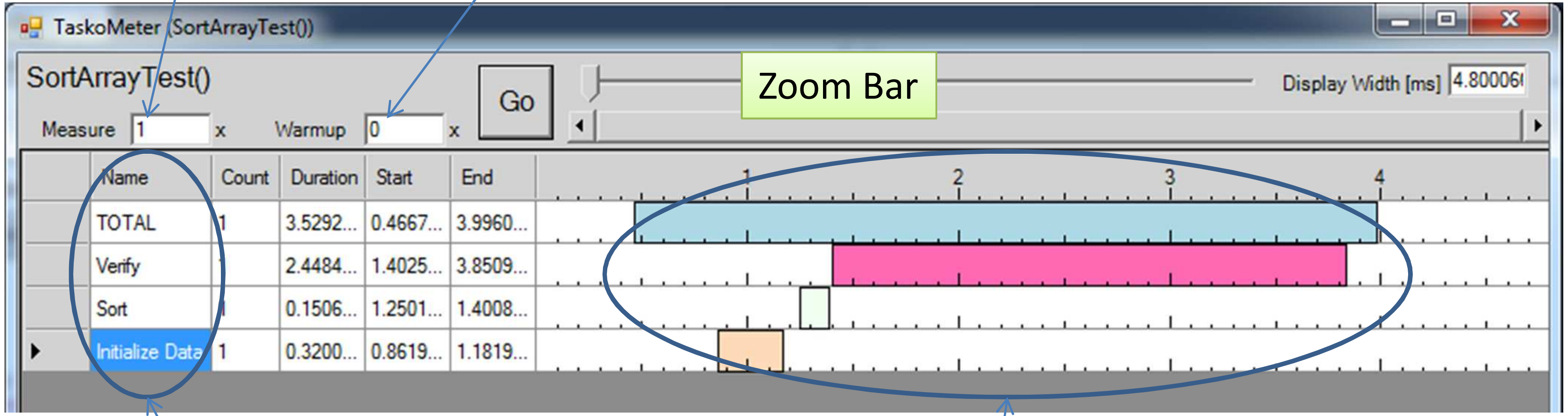
Run Performance Test Interactively

- In Alpaca, you will now see another node under the SortArrayTest() node called “Performance Test”
- Right-click and click “Run Performance Test Interactively”
- In the window that gets displayed, click the Go button.

TaskoMeter

# Repetitions to execute the test method

# Repetitions w/o timings (done first)


TaskoMeter (SortArrayTest())

SortArrayTest()

Measure 1 x   Warmup 0 x   Go

Zoom Bar

Display Width [ms] 4.80006

Name	Count	Duration	Start	End
TOTAL	1	3.5292...	0.4667...	3.9960...
Verify	1	2.4484...	1.4025...	3.8509...
Sort	1	0.1506...	1.2501...	1.4008...
Initialize Data	1	0.3200...	0.8619...	1.1819...

One row per task meter

Interval between a Start and Stop relative to other meters

Repetitions and Warmups

- You can specify…
  - Default # of time to run the test w/o timing
    - Has the effect of warming up the cache etc.
  - Default # of times to repeat the test and take timings

```
[PerformanceTestMethod(Repetitions=5, WarmupRepetitions=2)]
public void SortArrayTest()
…
```

Note about TaskoMeter

- When TaskoMeter runs, it loads the test assembly (and any of its dependencies) into memory. So while it’s open the OS keeps it from being modified.
- Therefore, you will not be able to rebuild those assemblies unless the TaskoMeter window that has it open has been closed
- When Alpaca launches a performance test, it does so by launching a new process for every test
- This helps prevent Alpaca from loading the assembly in its own app domain so you won’t have to close Alpaca to rebuild an assembly, just the TaskoMeter windows

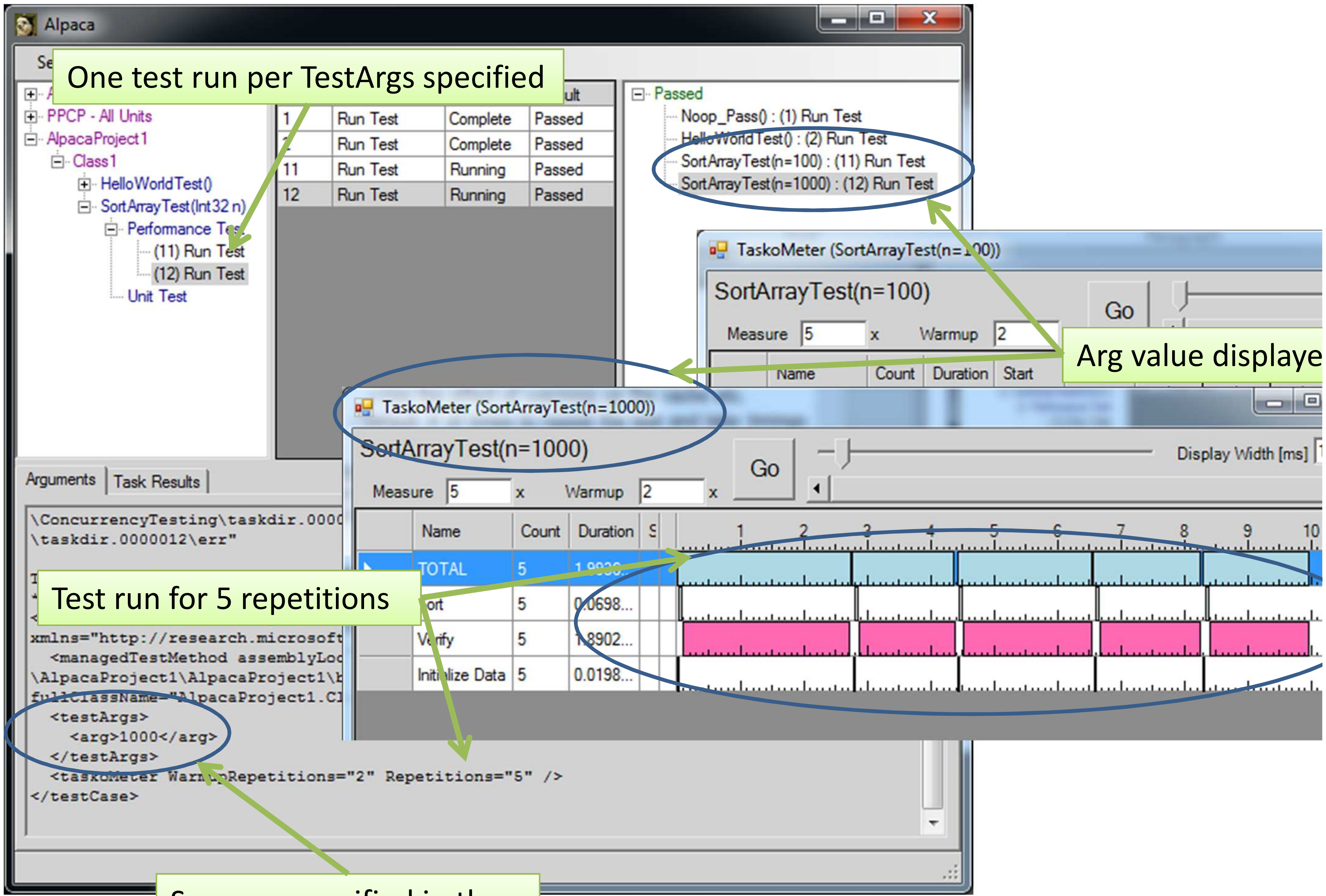
Test Arguments

- Run a test with a fixed set of arguments
- Specify using: [TestArgs(…)]
  - Number of values must match number of parameters for the method
  - Can declare more than one instance of the attribute; each represents the arguments for a single run of the test.

```
[TestArgs(100)]
[TestArgs(1000)]
public void SortArrayTest(int n)
{
   Random rand = new Random();
   …
```

TestArgs Limitations

- Only values that the complier can create at compile-time may be used
- Must be able to be converted to a string and then back to the parameter’s type



One test run per TestArgs specified

			ult
1	Run Test	Complete	Passed
2	Run Test	Complete	Passed
11	Run Test	Running	Passed
12	Run Test	Running	Passed

Passed
- Noop_Pass() : (1) Run Test
- HelloWorldTest() : (2) Run Test
- SortArrayTest(n=100) : (11) Run Test
- SortArrayTest(n=1000) : (12) Run Test

SortArrayTest(n=100)

Measure 5 x Warmup 2

Name	Count	Duration	Start

Arg value displayed

SortArrayTest(n=1000)

Measure 5 x Warmup 2 x

Name	Count	Duration	S
TOTAL	5	1.9930...	
Sort	5	0.0698...	
Verify	5	1.8902...	
Initialize Data	5	0.0198...	

Test run for 5 repetitions

```
\ConcurrencyTesting\taskdir.0000
\taskdir.0000012\err"

T
*
<
xmlns="http://research.microsoft
  <managedTestMethod assemblyLoc
\AlpacaProject1\AlpacaProject1\b
fullClassName="AlpacaProject1.Cl
  <testArgs>
    <arg>1000</arg>
  </testArgs>
  <taskoMeter WarmupRepetitions="2" Repetitions="5" />
</testCase>
```

See args specified in the test case's xml

Concurrency Unit Testing

- Different from regular unit testing
- Want to expose concurrency bugs
- May use various tools to analyze or instrument code to reason about correctness
- Often must be a limited set of execution code to speed up tests
- The nature of concurrency testing is different enough from normal unit testing that one generally doesn’t mark a test method as a unit test and a concurrency unit test method

Preemption Scheduling

- Preemption:

  The interruption of a running task with the intent to execute another task and return to the interrupted task at a later time.
- By controlling the scheduling of preemptions you can explore all possible schedules
- Certain optimizations can also be done to limit the exploration space.
- Alpaca uses the MS ChessTool to run schedule and data race tests.

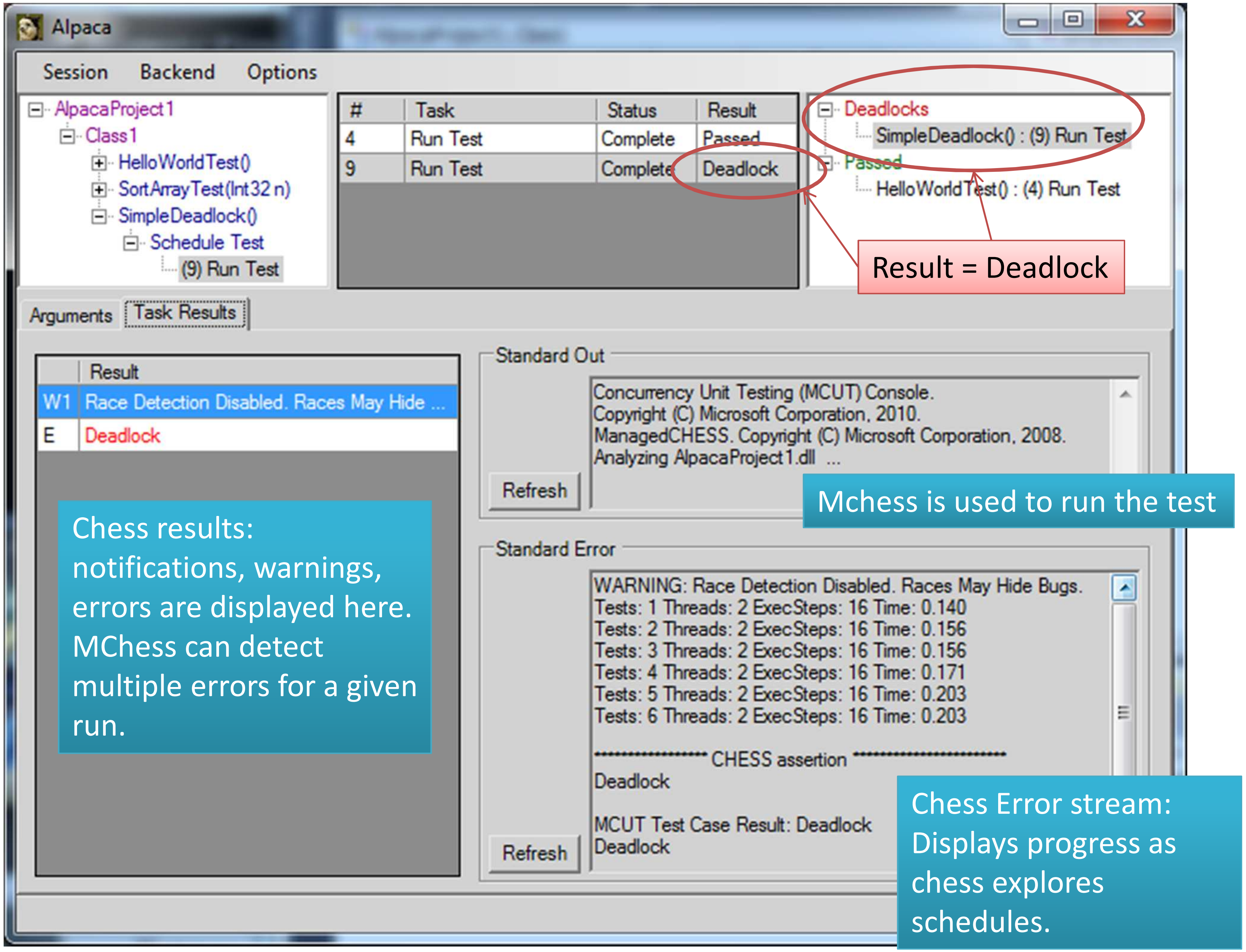
Schedule Tests

- Alpaca runs MChess in the background to instrument and explore preemption schedules
- Can be used to find common concurrency bugs
  - Deadlocks

Create a Schedule Test

- Create test method that simulates a deadlock
- Mark the method as a schedule test using the attribute [ScheduleTestMethod]
- Build and refresh assembly in Alpaca
- Run the test in Alpaca by right-clicking and selecting "Run Schedule Test"

```
[ScheduleTestMethod]
public void SimpleDeadlock()
{
    object syncObj1 = new object();
    object syncObj2 = new object();

    Parallel.Invoke(
        () => {
            lock (syncObj1)
                lock (syncObj2)
                    { }
        },
        () => {
            lock (syncObj2)
                lock (syncObj1)
                    { }
        });
}
```



#	Task	Status	Result
4	Run Test	Complete	Passed
9	Run Test	Complete	Deadlock

Result = Deadlock

	Result
W1	Race Detection Disabled. Races May Hide ...
E	Deadlock

Chess results: notifications, warnings, errors are displayed here. MChess can detect multiple errors for a given run.

Standard Out

```
Concurrency Unit Testing (MCUT) Console.
Copyright (C) Microsoft Corporation, 2010.
ManagedCHESS. Copyright (C) Microsoft Corporation, 2008.
Analyzing AlpacaProject1.dll ...
```

Mchess is used to run the test

Standard Error

```
WARNING: Race Detection Disabled. Races May Hide Bugs.
Tests: 1 Threads: 2 ExecSteps: 16 Time: 0.140
Tests: 2 Threads: 2 ExecSteps: 16 Time: 0.156
Tests: 3 Threads: 2 ExecSteps: 16 Time: 0.156
Tests: 4 Threads: 2 ExecSteps: 16 Time: 0.171
Tests: 5 Threads: 2 ExecSteps: 16 Time: 0.203
Tests: 6 Threads: 2 ExecSteps: 16 Time: 0.203

****************** CHESS assertion ***********************
Deadlock

MCUT Test Case Result: Deadlock
Deadlock
```

Chess Error stream: Displays progress as chess explores schedules.

Finding the Deadlock

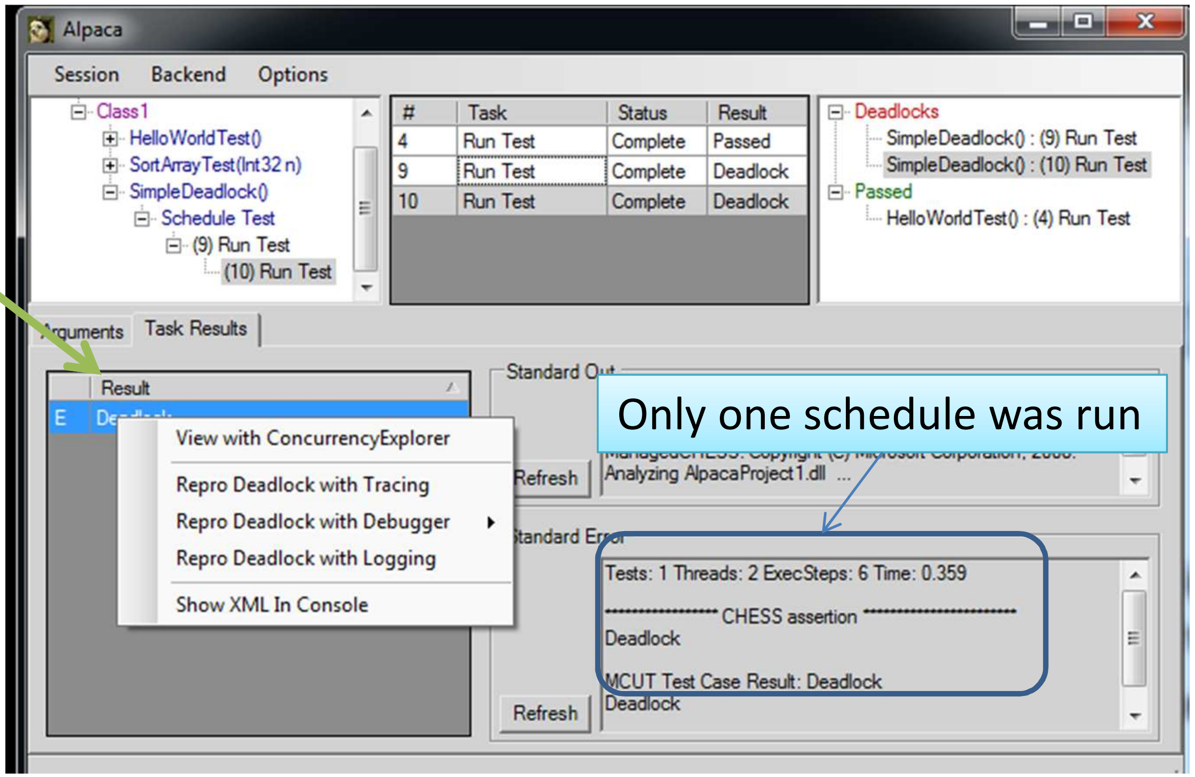
- Alpaca also exposes features of MChess
  - Run MChess again with the same schedule that found an error with Tracing to enable more diagnostics
  - Can view the operations that occurred in the schedule in the code using Concurrency Explorer

Run Deadlocked Schedule with Tracing

- In the “Task Results” tab, right click the “E|Deadlock” result item from the Chess Results grid
  - This is not the same as the one from the task list
- Click “Repro Deadlock with Tracing”
  - Note other options here for debugging ;-)
- The test will run with MChess again and return with another Deadlock result. Only this time just the schedule that produced the deadlock will be instrumented.

Open Schedule in Concurrency Explorer

- Right-click the new E|Deadlock row in the run that had tracing enabled
- Select "View in Concurrency Explorer"



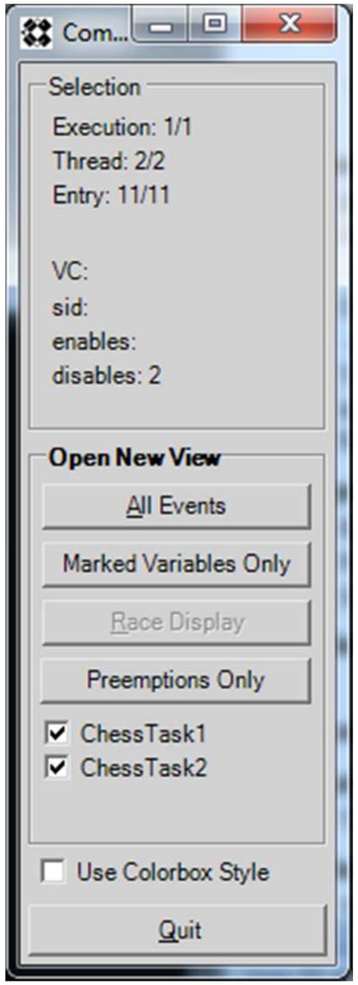
#	Task	Status	Result
4	Run Test	Complete	Passed
9	Run Test	Complete	Deadlock
10	Run Test	Complete	Deadlock

```
ManagedCHESS: Copyright (c) Microsoft Corporation, 2008.
Analyzing AlpacaProject1.dll ...
```

```
Tests: 1 Threads: 2 ExecSteps: 6 Time: 0.359
**************** CHESS assertion **********************
Deadlock
MCUT Test Case Result: Deadlock
Deadlock
```

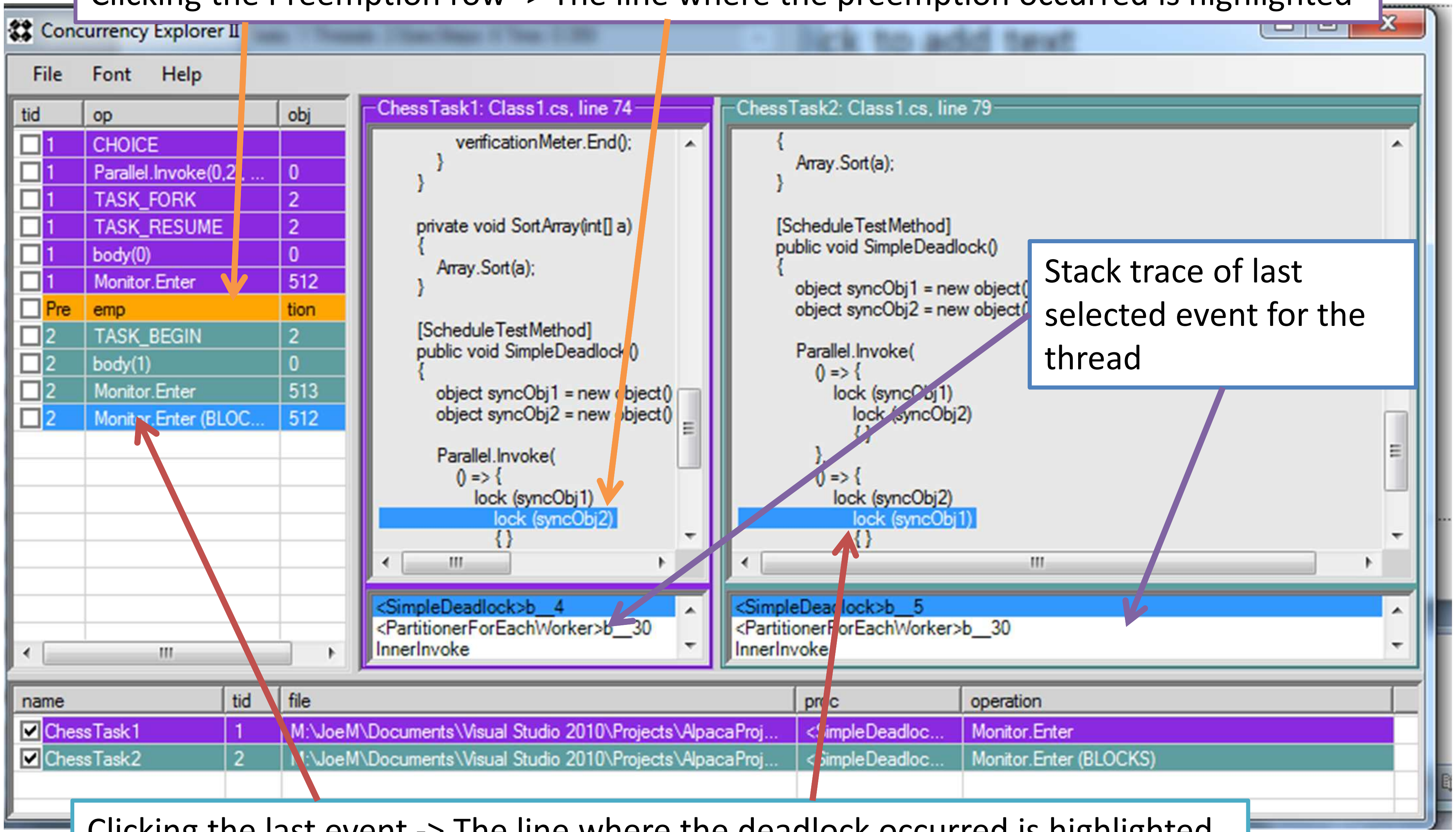
View Deadlock in Concurrency Explorer

- When opening the Concurrency Explorer (CE) from a deadlocked schedule it automatically opens the “All Events” window.
- By clicking on the orange “Preemption” item you see where the first thread preempts just before obtaining the 2nd lock.
- Clicking next on the 2nd thread’s last instruction (text in orange) you see the 2nd thread preempted just before optaining it’s 2nd lock.
- Both threads are waiting on the lock for which the other thread has obtained.


Selection
Execution: 1/1
Thread: 2/2
Entry: 11/11

VC:
sid:
enables:
disables: 2

Concurrency Explorer



Clicking the Preemption row -> The line where the preemption occurred is highlighted

Stack trace of last selected event for the thread

Clicking the last event -> The line where the deadlock occurred is highlighted

tid	op	obj
1	CHOICE	
1	Parallel.Invoke(0,2, ...	0
1	TASK_FORK	2
1	TASK_RESUME	2
1	body(0)	0
1	Monitor.Enter	512
Pre	emp	tion
2	TASK_BEGIN	2
2	body(1)	0
2	Monitor.Enter	513
2	Monitor.Enter (BLOC...	512

ChessTask1: Class1.cs, line 74

```
            verificationMeter.End();
        }
    }

    private void SortArray(int[] a)
    {
        Array.Sort(a);
    }

    [ScheduleTestMethod]
    public void SimpleDeadlock()
    {
        object syncObj1 = new object()
        object syncObj2 = new object()

        Parallel.Invoke(
          () => {
            lock (syncObj1)
              lock (syncObj2)
              { }
```

```
<SimpleDeadlock>b__4
<PartitionerForEachWorker>b__30
InnerInvoke
```

ChessTask2: Class1.cs, line 79

```
    {
        Array.Sort(a);
    }

    [ScheduleTestMethod]
    public void SimpleDeadlock()
    {
        object syncObj1 = new object(
        object syncObj2 = new object(

        Parallel.Invoke(
          () => {
            lock (syncObj1)
              lock (syncObj2)
              { }
          },
          () => {
            lock (syncObj2)
              lock (syncObj1)
              { }
```

```
<SimpleDeadlock>b__5
<PartitionerForEachWorker>b__30
InnerInvoke
```

name	tid	file	proc	operation
☑ ChessTask1	1	M:\JoeM\Documents\Visual Studio 2010\Projects\AlpacaProj...	<SimpleDeadloc...	Monitor.Enter
☑ ChessTask2	2	M:\JoeM\Documents\Visual Studio 2010\Projects\AlpacaProj...	<SimpleDeadloc...	Monitor.Enter (BLOCKS)

Fix the Deadlock

- Fix the deadlock by swapping the order of lock acquisition for one of the threads.
- Run the test again.
- The test should now pass.

Data Race Tests

- Data Race: When two concurrent threads access the same memory location and one of the accesses is a write
- Notice the warning given by Mchess when we ran the previous schedule test:
  WARNING: Race Detection Disabled. Races May Hide Bugs.
- When data race detection is enabled Mchess verifies that preempted threads don't cause a data race
- Same number of schedules explored as a regular Schedule Test

Create a Test With a Data Race

- Create a basic data race test where two threads read/write to the same shared variable
- Mark the test with the [DataRaceTestMethod] attribute
- Run the test in Alpaca

```
[DataRaceTestMethod]
public void SimpleDataRace()
{
   int cnt = 0;
   Parallel.Invoke(
      () => cnt++,
      () => cnt++
      );
   Assert.AreEqual(2, cnt);
}
```

View The Data Race

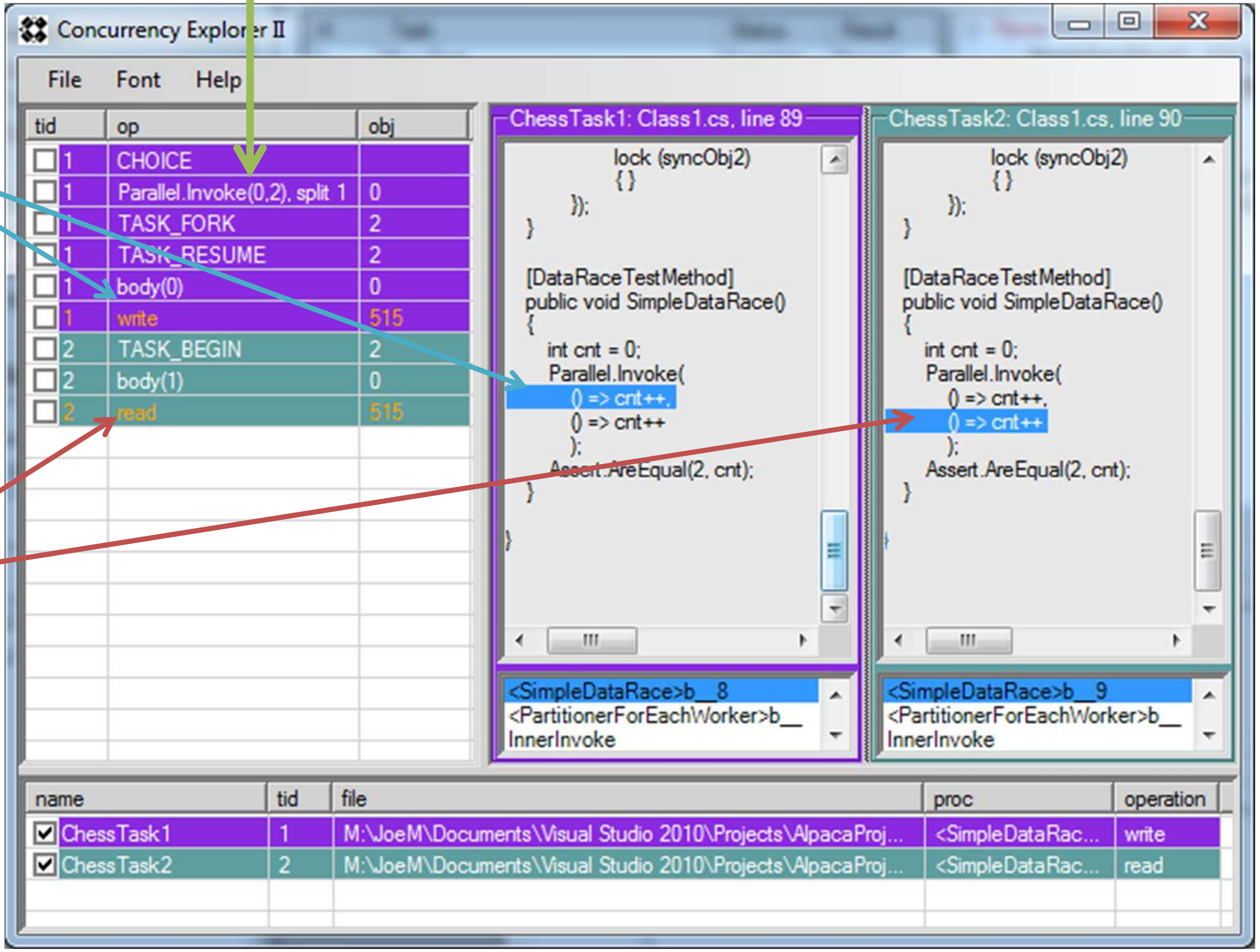
- Rerun the erroring schedule by right-clicking on [one of] the data race Chess errors and clicking “Repro Race Rx with Tracing”
- After this test finishes, right-click the R1 row and click “View Race with Concurrency Explorer”

Data Race in Concurrency Explorer

Each color represents a different thread's operations

Thread 1: Write

Thread 2: Read



tid	op	obj
1	CHOICE	
1	Parallel.Invoke(0,2), split 1	0
1	TASK_FORK	2
1	TASK_RESUME	2
1	body(0)	0
1	write	515
2	TASK_BEGIN	2
2	body(1)	0
2	read	515

ChessTask1: Class1.cs, line 89

```
                lock (syncObj2)
                { }
            });
    }

    [DataRaceTestMethod]
    public void SimpleDataRace()
    {
        int cnt = 0;
        Parallel.Invoke(
            () => cnt++,
            () => cnt++
        );
        Assert.AreEqual(2, cnt);
    }

}
```

```
<SimpleDataRace>b__8
<PartitionerForEachWorker>b__
InnerInvoke
```

ChessTask2: Class1.cs, line 90

```
                lock (syncObj2)
                { }
            });
    }

    [DataRaceTestMethod]
    public void SimpleDataRace()
    {
        int cnt = 0;
        Parallel.Invoke(
            () => cnt++,
            () => cnt++
        );
        Assert.AreEqual(2, cnt);
    }
```

```
<SimpleDataRace>b__9
<PartitionerForEachWorker>b__
InnerInvoke
```

name	tid	file	proc	operation
☑ ChessTask1	1	M:\JoeM\Documents\Visual Studio 2010\Projects\AlpacaProj...	<SimpleDataRac...	write
☑ ChessTask2	2	M:\JoeM\Documents\Visual Studio 2010\Projects\AlpacaProj...	<SimpleDataRac...	read

Chess Tests

- The ScheduleTestMethod and DataRaceTestMethod attributes are just abstractions of the attributes available to create a test that runs using MChess.
- Namespace: `Microsoft.Concurrency.TestTools.UnitTesting.Chess`
- There is some documentation already via intellisense in Visual Studio.
- Note an MChess test is for managed code only. Thus the attribute is simply ChessTestMethod since the ‘M’ is implied.
- We’ll just provide a pretty complex example of an MChess test.

MChess Test Sample

```
1:[ChessTestMethod()]
2:[ChessTestContext(
3:     PreemptAllAccesses = true
4:     , ExtraCommandLineArgs = new[]{
5:          "/dpt:AntisocialRobots.RobotSimulationBase"
6:          ,"/dpt:AlpacaProject.RobotSimulationFixes"
7:     })]
8:public void NaiveParallel_ScheduleTest()
```

Line 1: Simply marks the method as an MChess test method.
Line 2: Specifies a context under which to run the test. There can be multiple contexts specified for a test. When the test is run, Alpaca executes one run per cross product of context and specified TestArgs (if any). If no name is specified, then it’s the default context. If more than one contexts are specified, they must have unique names.

MChess Test Sample (cont.)

- **Line 3**: This particular test doesn’t make use of any threading/locking primitives and thus there would be no preemptions. By specifying this property, MChess will explore schedules where it will preempt on all memory accesses. The problem with this (as you may guess) is that the number of schedules explored is huge.
- **Line 4**: While the most common options are implemented via attribute properties sometimes you need to specify an option on the command-line yourself. This array of strings (one string per arg) allows you to do just that.
- **Lines 5-6**: The /dp[tmn] command line option tells MChess “Don’t Preempt [Type, Method, Namespace]” This helps to minimize the schedule exploration space for this test so it takes much less time to run.

Other Features

- [ExpectedException(typeof(…Exception))]
  - Asserts that the method throws an exception of the specified type
- [ExpectedResult(TestResultType.AssertFailure)]
  - Indicates that for a test to pass in Alpaca, the test should produce the specified result
  - E.g. When you expect a test to produce a deadlock but don’t want Alpaca to report one
- You can right-click a chess error and Debug that schedule
  - Allows you to attach a debugger to the process

Regression Testing

- Another layer after the ExpectedResult attribute that is only run when regression test assertions are enabled
- Allows a test to pass regression testing if the result is any of those specified
- Ignored by Alpaca
- Enabled for the following mcut command:

  ```
  >mcut runAllTests [testAssembly]
  ```

  This runs all tests and prints results to the Console.

```
/// This test may or may not find the deadlock since we aren't using MChess
[UnitTestMethod]
[RegressionTestExpectedResult(TestResultType.Passed, TestResultType.DeadLock)]
public void MyLoadTest()
```