EASYMOCK - CREATESTRICTMOCK

EasyMock.createStrictMock creates a mock and also takes care of the order of method calls that the mock is going to make in due course of its action.

Syntax

```
calcService = EasyMock.createStrictMock(CalculatorService.class);
```

Example

Step 1: Create an interface called CalculatorService to provide mathematical functions

File: CalculatorService.java

```
public interface CalculatorService {
   public double add(double input1, double input2);
   public double subtract(double input1, double input2);
   public double multiply(double input1, double input2);
   public double divide(double input1, double input2);
}
```

Step 2: Create a JAVA class to represent MathApplication

File: MathApplication.java

```
public class MathApplication {
   private CalculatorService calcService;

   public void setCalculatorService(CalculatorService calcService){
      this.calcService = calcService;
   }

   public double add(double input1, double input2){
      return calcService.add(input1, input2);
   }

   public double subtract(double input1, double input2){
      return calcService.subtract(input1, input2);
   }

   public double multiply(double input1, double input2){
      return calcService.multiply(input1, input2);
   }

   public double divide(double input1, double input2){
      return calcService.divide(input1, input2);
   }
}
```

Step 3: Test the MathApplication class

Let's test the MathApplication class, by injecting in it a mock of calculatorService. Mock will be created by EasyMock.

Here we've added two mock method calls, add and subtract, to the mock object via expect. However during testing, we've called subtract before calling add. When we create a mock object using EasyMock.createStrictMock, the order of execution of the method does matter.

File: MathApplicationTester.java

```
import org.easymock.EasyMock;
```

```
import org.easymock.EasyMockRunner;

import org.junit.Assert;
import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;

@RunWith(EasyMockRunner.class)
public class MathApplicationTester {

   private MathApplication mathApplication;
   private CalculatorService calcService;

   @Before
   public void setUp(){
      mathApplication = new MathApplication();
      calcService = EasyMock.createStrictMock(CalculatorService.class);
      mathApplication.setCalculatorService(calcService);
   }

   @Test
   public void testAddAndSubtract(){

      //add the behavior to add numbers
      EasyMock.expect(calcService.add(20.0,10.0)).andReturn(30.0);

      //subtract the behavior to subtract numbers
      EasyMock.expect(calcService.subtract(20.0,10.0)).andReturn(10.0);

      //activate the mock
      EasyMock.replay(calcService);

      //test the subtract functionality
      Assert.assertEquals(mathApplication.subtract(20.0, 10.0),10.0,0);

      //test the add functionality
      Assert.assertEquals(mathApplication.add(20.0, 10.0),30.0,0);

      //verify call to calcService is made or not
      EasyMock.verify(calcService);
   }
}
```

Step 4: Execute test cases

Create a java class file named TestRunner in **C:\> EasyMock_WORKSPACE** to execute Test cases .

File: TestRunner.java

```
import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;

public class TestRunner {
   public static void main(String[] args) {
      Result result = JUnitCore.runClasses(MathApplicationTester.class);

      for (Failure failure : result.getFailures()) {
         System.out.println(failure.toString());
      }

      System.out.println(result.wasSuccessful());
   }
}
```

Step 5: Verify the Result

Compile the classes using **javac** compiler as follows:

```
C:\EasyMock_WORKSPACE>javac MathApplicationTester.java
```

Now run the Test Runner to see the result:

```
C:\EasyMock_WORKSPACE>java TestRunner
```

Verify the output.

```
testAddAndSubtract(com.tutorialspoint.mock.MathApplicationTester):
   Unexpected method call CalculatorService.subtract(20.0, 10.0):
      CalculatorService.add(20.0, 10.0): expected: 1, actual: 0
false
```