S/W 품질관리 및 테스팅

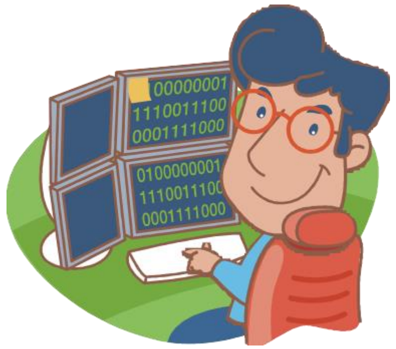
설계 #6: Mock Object Testing

CSE4061

2023

설계 목표

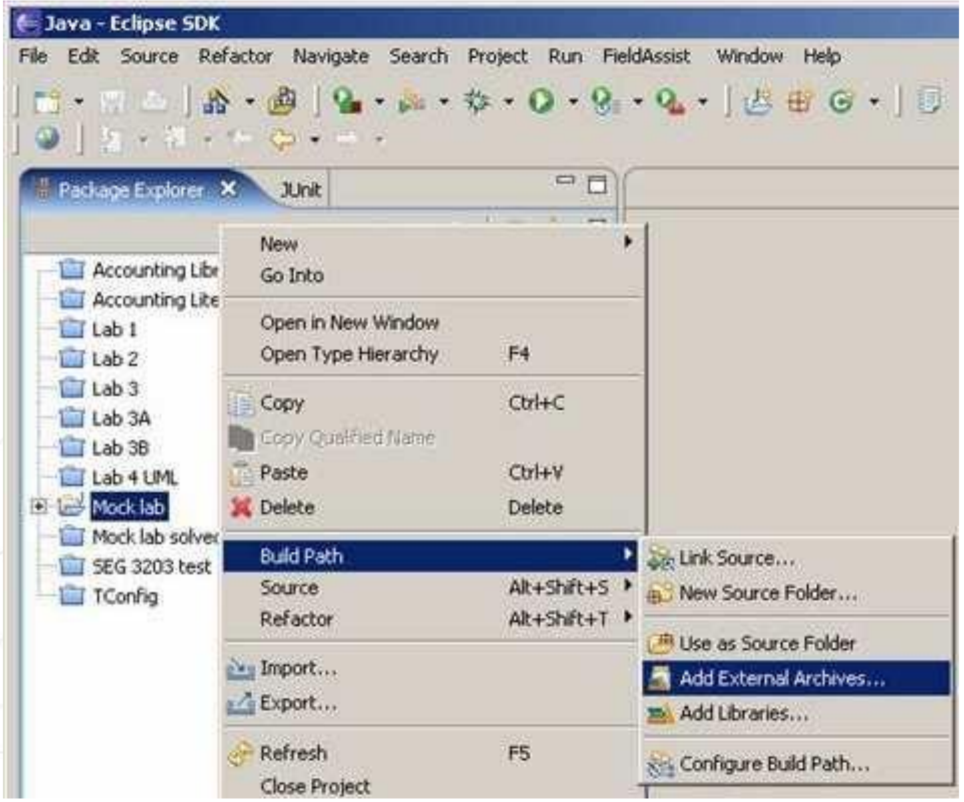
- EasyMock 셋업
- Mock 객체 테스팅 사례
- 설계 문제
  - Mock 객체를 사용하여 테스팅 진행



동국대학교 dongguk university

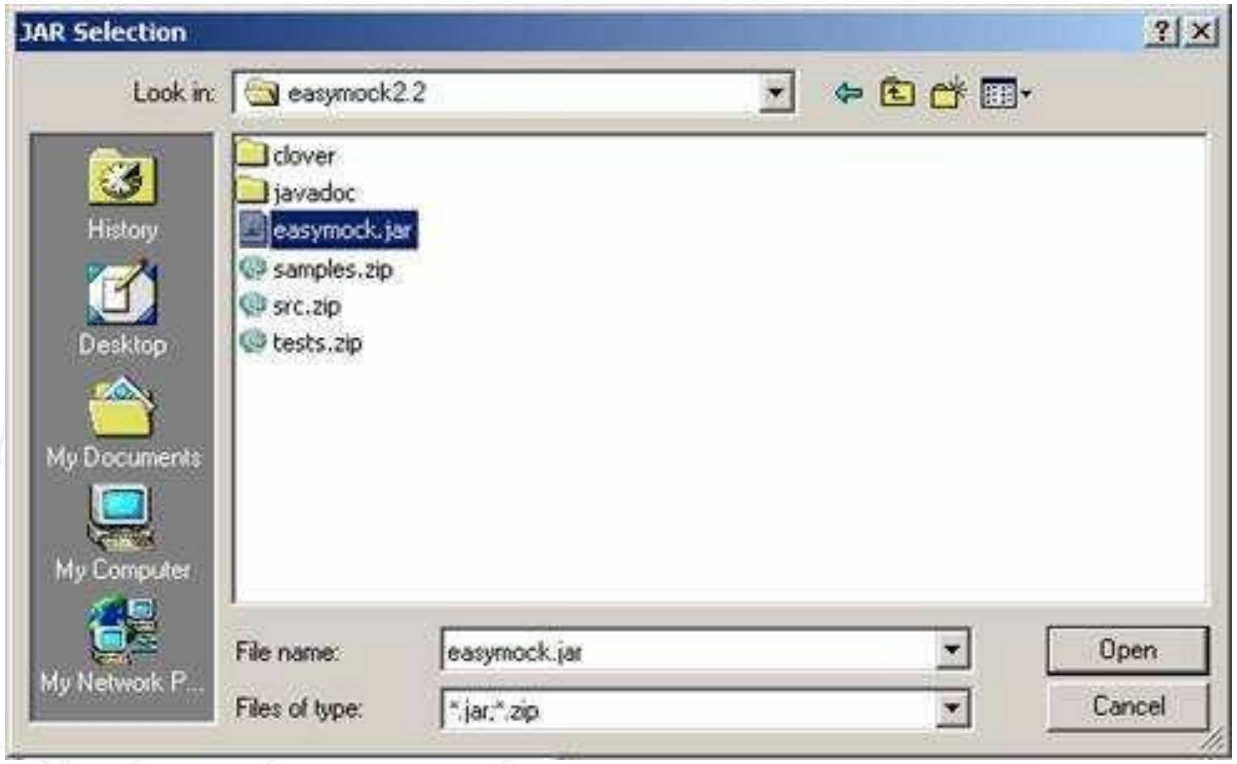
EasyMock 셋업

- 다음 링크로부터 다운 받기
  - https://easymock.org/
  - 최신 버전 easymock-5.1-bundle 을 다운 받아야 함
- 임시 폴더에서 풀면
  - easymock-5.1 폴더가 생김
  - 폴더 안에 easymock-5.1.jar 파일을 빌드 경로로 추가



프로젝트 생성

- 프로젝트 Mock Lab 생성하고
- Build Path > Add External Archives
- Easymock5.1 폴더 안의 easymock-5.1.jar을 선택

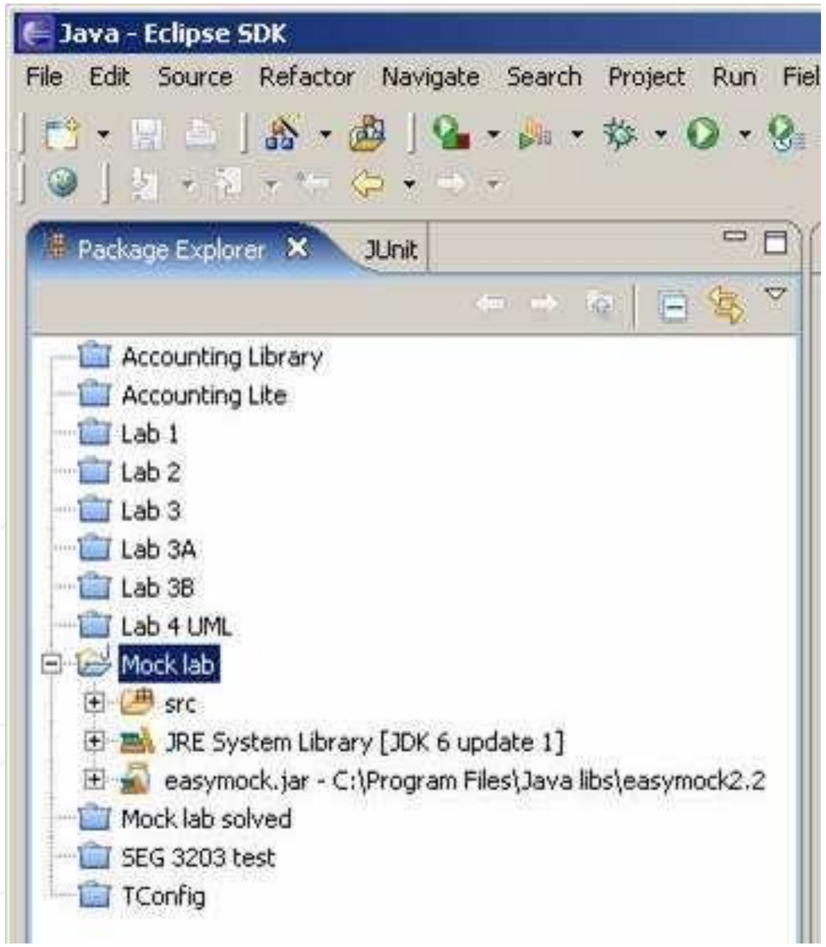


File name: easymock.jar

Files of type: *.jar;*.zip

변경된 빌드 경로

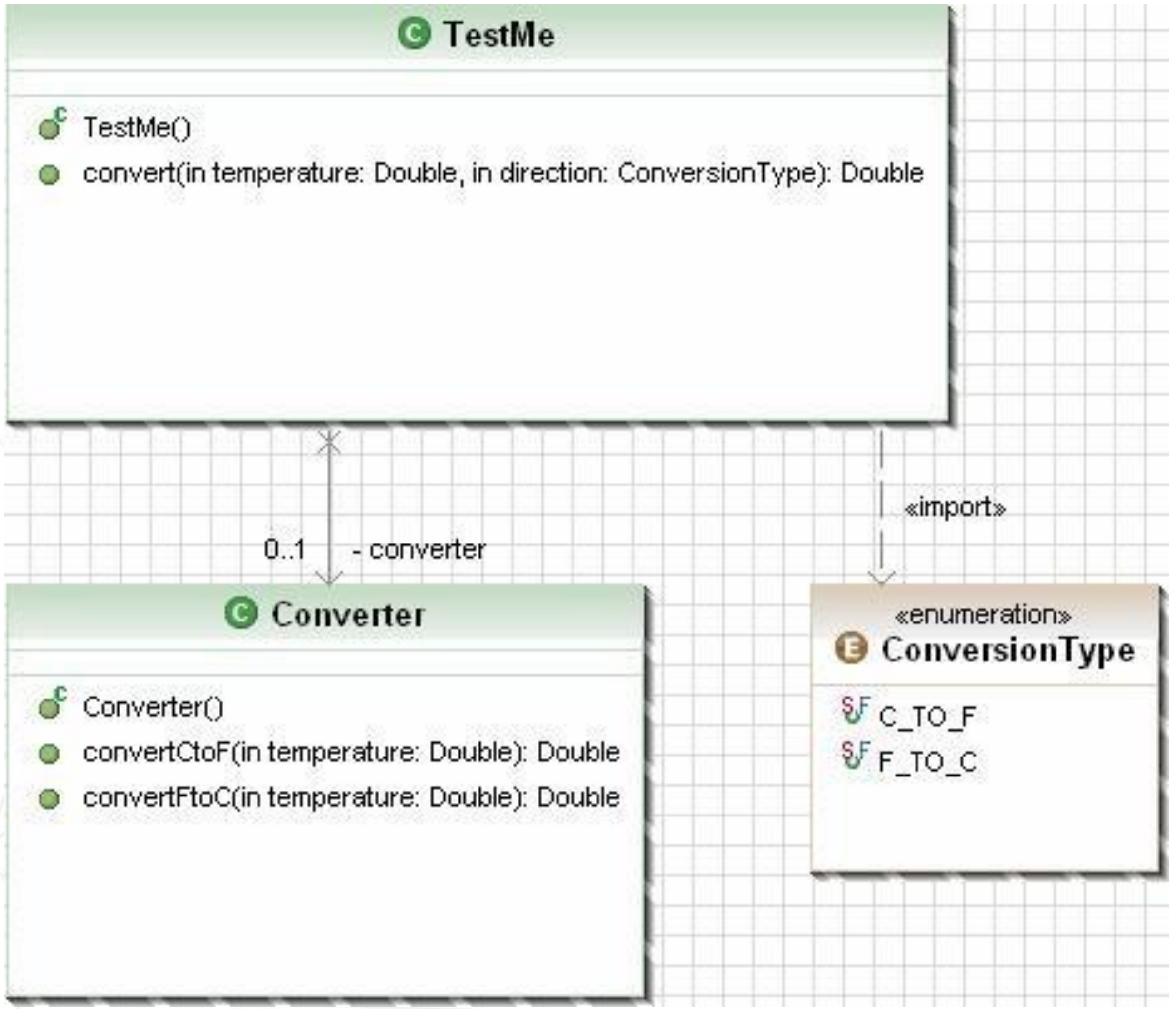
- .jar 파일을 포함하기 위하여 open 버튼을 누르면
- 생성된 Project에 변경된 빌드 경로 포함됨



Accounting Library
Accounting Lite
Lab 1
Lab 2
Lab 3
Lab 3A
Lab 3B
Lab 4 UML
Mock lab
src
JRE System Library [JDK 6 update 1]
easymock.jar - C:\Program Files\Java libs\easymock2.2
Mock lab solved
SEG 3203 test
TConfig

테스트할 클래스와 helper 클래스 생성

- 테스트할 클래스 – TestMe
- 연관된 클래스 – Converter


TestMe
TestMe()
convert(in temperature: Double, in direction: ConversionType): Double
0..1
- converter
«import»
Converter
Converter()
convertCtoF(in temperature: Double): Double
convertFtoC(in temperature: Double): Double
«enumeration»
ConversionType
C_TO_F
F_TO_C

Converter 클래스 생성

```
package original;

public class Converter
{

   public Converter( )
   {
   }

   public double convertCtoF( double temperature )
   {
      return (double)( temperature * 9.0 / 5.0 + 32.0 );
   }

   public double convertFtoC( double temperature )
   {
      return (double)( ( temperature - 32.0 ) * 5.0 / 9.0 );
   }

}
```

Conversion Type 생성

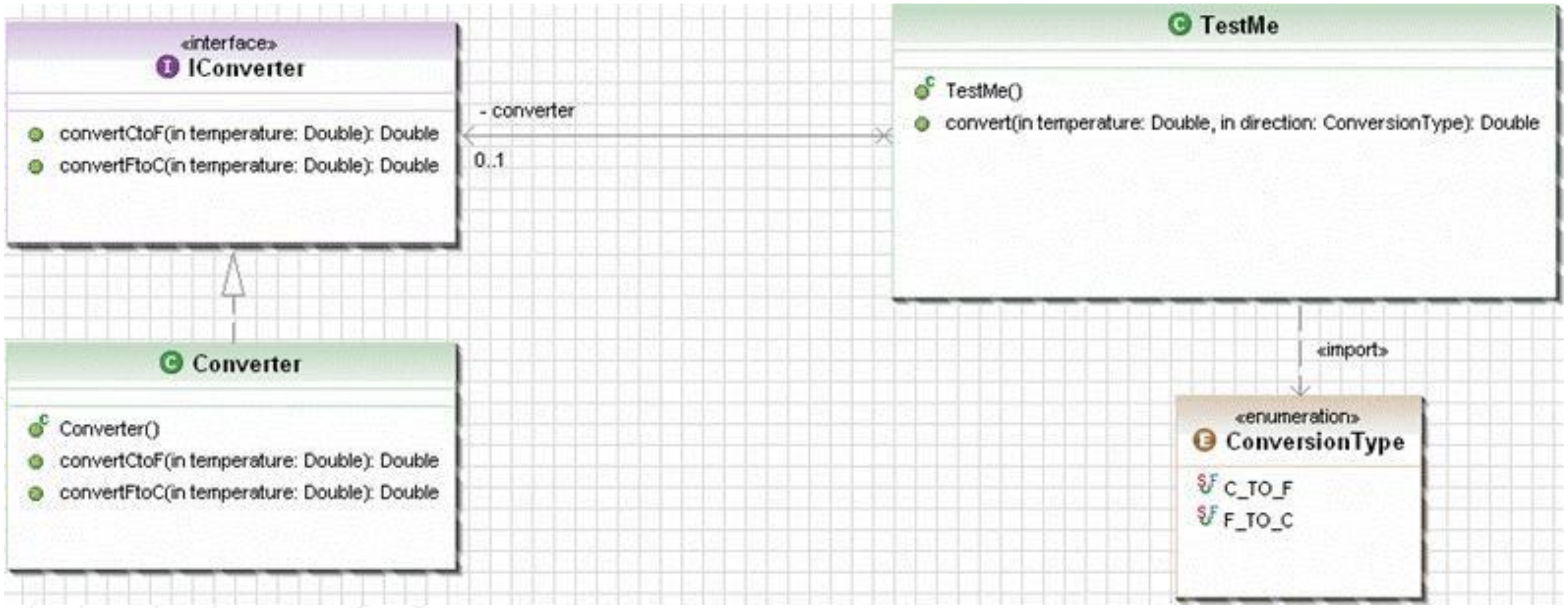
```
package original;

public enum ConversionType
{
    C_TO_F, F_TO_C
}
```

TestMe 클래스 생성

```
package original;
public class TestMe
{
   private Converter converter;
   public TestMe( )
   {
      converter = new Converter( );
   }
   public Double convert( Double temperature, ConversionType direction )
   {
      Double result = null;
      if ( converter != null )
      {
         switch ( direction )
         {
            case C_TO_F:
            {
               result = converter.convertCtoF( temperature );
               break;
            }
            case F_TO_C:
            {
               result = converter.convertFtoC( temperature );
               break;
            }
         }
      }
      return result;
   }
}
```

제어 역전 패턴 적용



«interface»
IConverter
convertCtoF(in temperature: Double): Double
convertFtoC(in temperature: Double): Double
- converter
0..1
TestMe
TestMe()
convert(in temperature: Double, in direction: ConversionType): Double
Converter
Converter()
convertCtoF(in temperature: Double): Double
convertFtoC(in temperature: Double): Double
«import»
«enumeration»
ConversionType
C_TO_F
F_TO_C

EasyMock 이용

```
import org.easymock.EasyMock;
//...
// Create the mock object
IConverter converter = EasyMock.createMock( IConverter.class );

// Tell the mock object to expect a method call with specified parameter
  and return value.
EasyMock.expect( converter.convertFtoC( new Double( 32.0 ) )
  ).andReturn( new Double( 0.0 ) );
// More expect() calls can be provided here if necessary.

// Activate the mock object.  From this point on, it will be functioning
  as if it was a real object.
EasyMock.replay( converter );

//...
// Do something with should call convertFtoC( 32.0 ) using converter;
//...

// Verify that the mock object was called.
EasyMock.verify( converter );
```

사례 – Electronic Store

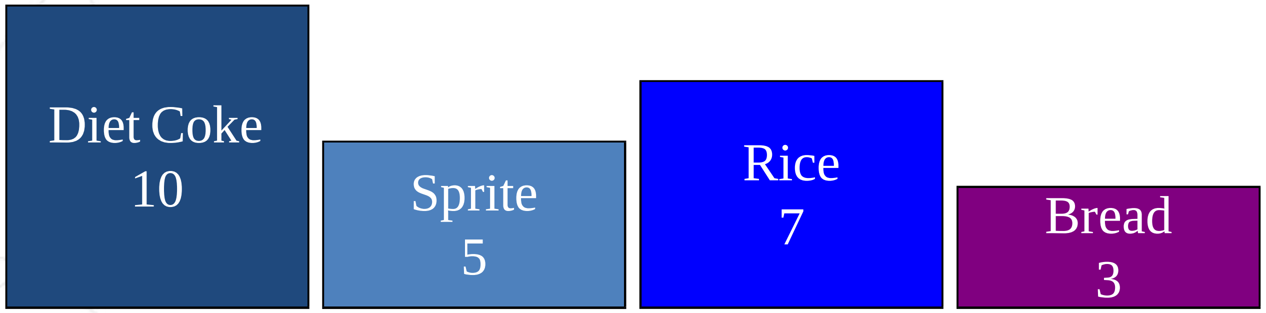
- **주문(Order)과 창고(Warehouse)**

Order1: Diet Coke - 5

Order2: Diet Coke - 2

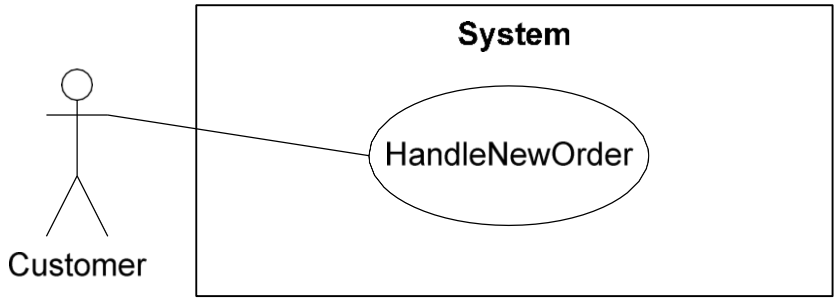
Order3: Sprite - 3

Order4: Bread - 1

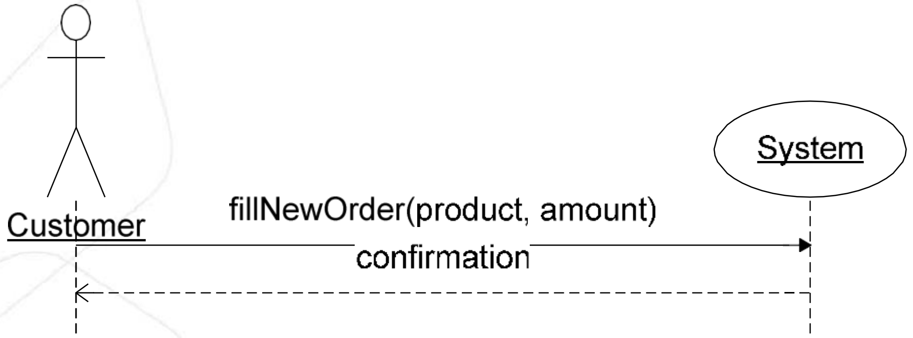


사례 – Electronic Store

- 사용 사례 모델

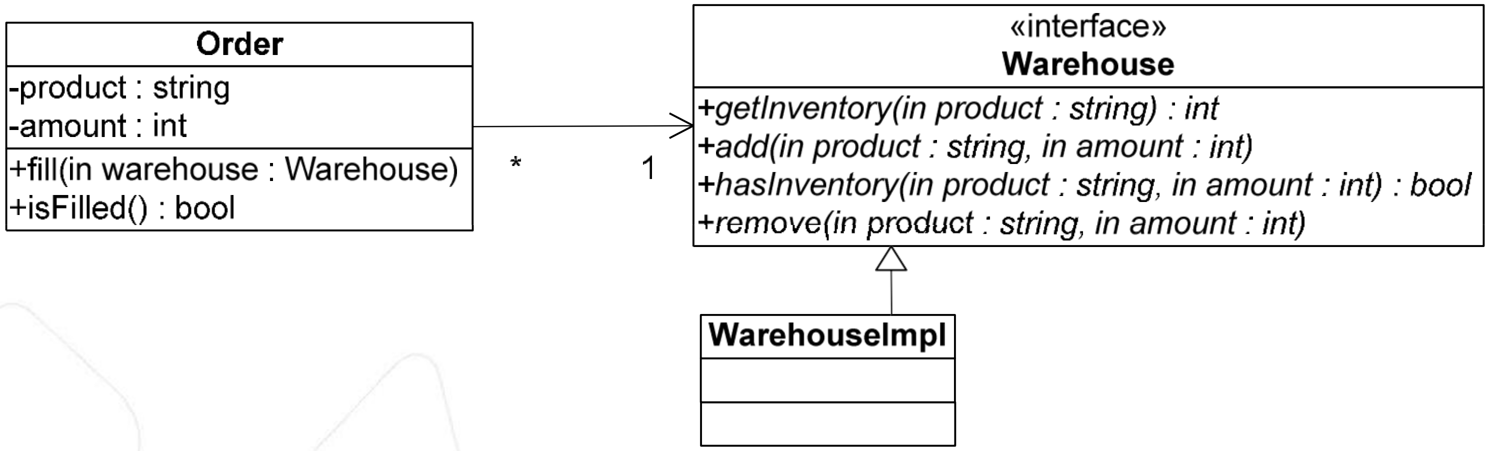


- 시스템 구현



사례 – Electronic Store

- 클래스 모델


Order
-product : string
-amount : int
+fill(in warehouse : Warehouse)
+isFilled() : bool
*
1
«interface»
Warehouse
+getInventory(in product : string) : int
+add(in product : string, in amount : int)
+hasInventory(in product : string, in amount : int) : bool
+remove(in product : string, in amount : int)
WarehouseImpl

사례 – Electronic Store

- **Order 클래스 테스팅:**

```
public class OrderStateTester extends TestCase {
       private static String DIET_COKE = "Diet Coke";
       private static String SPRITE = "Sprite";
       Warehouse warehouse;

       protected void setUp() throws Exception {
              //Fixture with secondary object(s)
              warehouse = new WarehouseImpl();
              warehouse.add(DIET_COKE,5);
              warehouse.add(SPRITE,10);
       }
       …
```

사례 – Electronic Store

```
public class OrderStateTester extends TestCase {
        …
        public void testOrderIsFilledIfEnoughInWarehouse(){ Order
                order = new Order(DIET_COKE,5);
                order.fill(warehouse);
                // Primary object test
                assertTrue(order.isFilled());
                // Secondary object test(s)
                assertEquals(0,warehouse.getInventory(DIET_COKE));
        }

        public void testOrderDoesNotRemoveIfNotEnough(){ Order
                order = new Order(SPRITE,11);
                order.fill(warehouse);
                // Primary object test
                assertFalse(order.isFilled());
                // Secondary object test(s)
                assertEquals(10, warehouse.getInventory(SPRITE));
        }
}
```

사례 – Electronic Store

- 테스팅을 위하여 스텁을 사용하는 경우 -
  - 메소드 호출에 대하여 간단히 준비된 자료를 리턴:

```
public    class WarehouseImpl implements Warehouse {

          public void add(String product, int i) {}

          public int getInventory(String product) {
                  return 0;
          }

          public boolean hasInventory(String product, int amount)    {
                  return false;
          }

          public void remove(String product, int i) {                         }
}
```

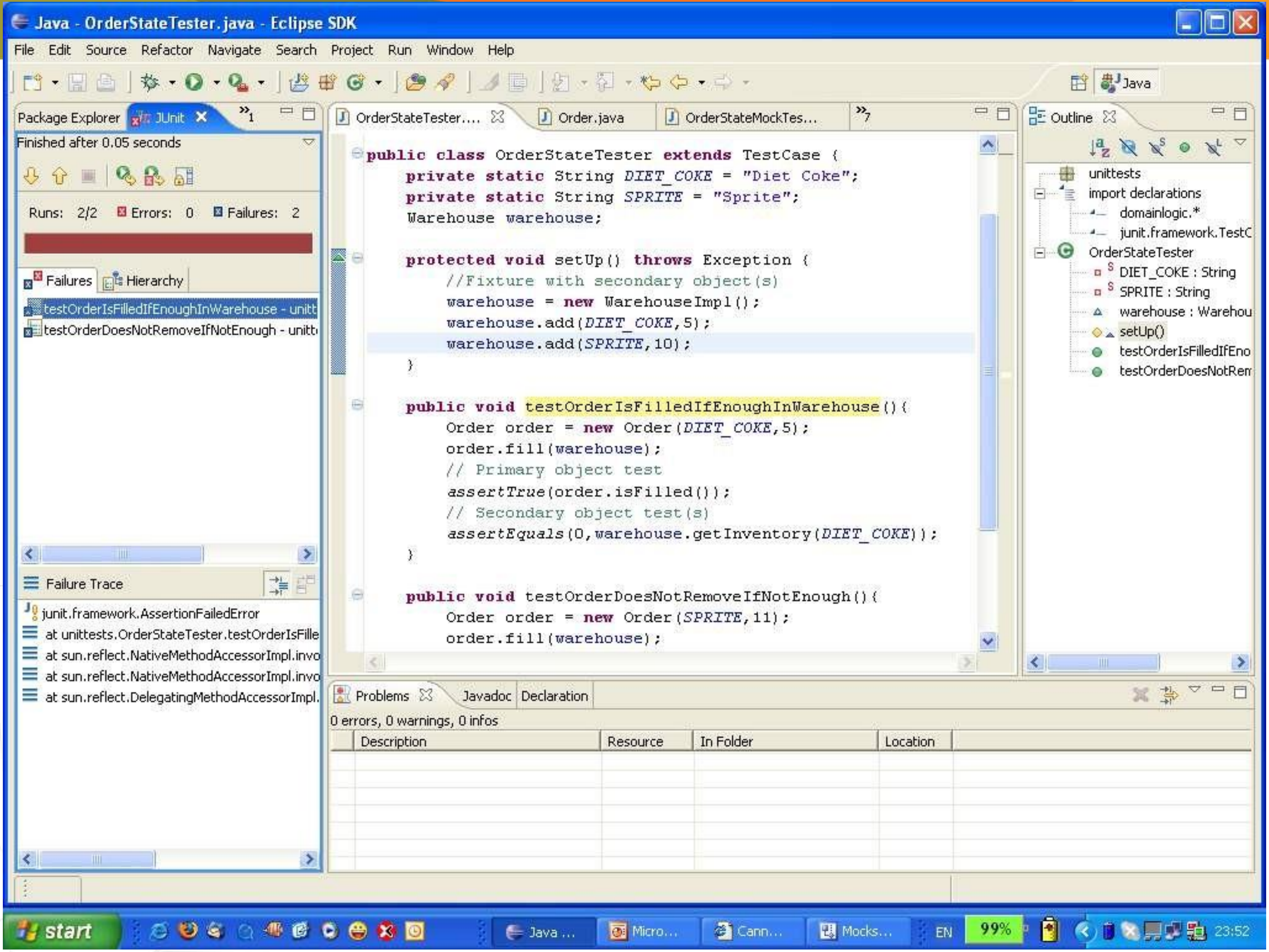
사례 – Electronic Store

```
public class Order {
        …
        public Order(String product, int i) {
                this.product = product;
                this.amount =  i;
                this.isFilled = false;
        }

        public void fill(Warehouse warehouse) {
                if (warehouse.hasInventory(product,amount)) {
                        warehouse.remove(product,amount);
                        isFilled = true;
                }
        }

        public boolean isFilled() {
                return isFilled;
        }
}
```

- **CUT 테스트 통과할까?**



Finished after 0.05 seconds

Runs: 2/2 Errors: 0 Failures: 2

testOrderIsFilledIfEnoughInWarehouse - unitt
testOrderDoesNotRemoveIfNotEnough - unitt

Failure Trace

```
junit.framework.AssertionFailedError
at unittests.OrderStateTester.testOrderIsFille
at sun.reflect.NativeMethodAccessorImpl.invo
at sun.reflect.NativeMethodAccessorImpl.invo
at sun.reflect.DelegatingMethodAccessorImpl.
```

```java
public class OrderStateTester extends TestCase {
    private static String DIET_COKE = "Diet Coke";
    private static String SPRITE = "Sprite";
    Warehouse warehouse;

    protected void setUp() throws Exception {
        //Fixture with secondary object(s)
        warehouse = new WarehouseImpl();
        warehouse.add(DIET_COKE,5);
        warehouse.add(SPRITE,10);
    }

    public void testOrderIsFilledIfEnoughInWarehouse(){
        Order order = new Order(DIET_COKE,5);
        order.fill(warehouse);
        // Primary object test
        assertTrue(order.isFilled());
        // Secondary object test(s)
        assertEquals(0,warehouse.getInventory(DIET_COKE));
    }

    public void testOrderDoesNotRemoveIfNotEnough(){
        Order order = new Order(SPRITE,11);
        order.fill(warehouse);
```

```
unittests
import declarations
    domainlogic.*
    junit.framework.TestC
OrderStateTester
    DIET_COKE : String
    SPRITE : String
    warehouse : Warehou
    setUp()
    testOrderIsFilledIfEno
    testOrderDoesNotRem
```

0 errors, 0 warnings, 0 infos

Description	Resource	In Folder	Location

사례 – Electronic Store

- Mock 클래스

```
public class WarehouseMock implements Warehouse {

        int inventoryResult;
        boolean hasInventoryResult;
        int expectedCalls,actualCalls;
        …
        public int getInventory(String product) {
                actualCalls++;
                return inventoryResult;
        }

        public void setGetInventoryResult(int result) {
                this.inventoryResult = result;
                expectedCalls++;
        }

        public boolean verify(){
                return expectedCalls == actualCalls;
        }
}
```

사례 – Electronic Store

- **인터랙션 기반 테스팅:**

```
public class OrderInteractionMockTester extends TestCase {
        …
        protected void setUp() throws Exception {
                //Fixture with secondary object(s)
                warehouse = new WarehouseMock();
        }

        public void testOrderIsFilledIfEnoughInWarehouse(){
                //Expectations
                warehouse.setHasInventoryResult(true);
                warehouse.setRemoveInvoked();

                Order order = new Order(DIET_COKE,5);
                order.fill(warehouse);
                // Primary object test
                assertTrue(order.isFilled());
                // Secondary object test(s)
                assertTrue(warehouse.verify());
        }
        …
```

사례 – Electronic Store

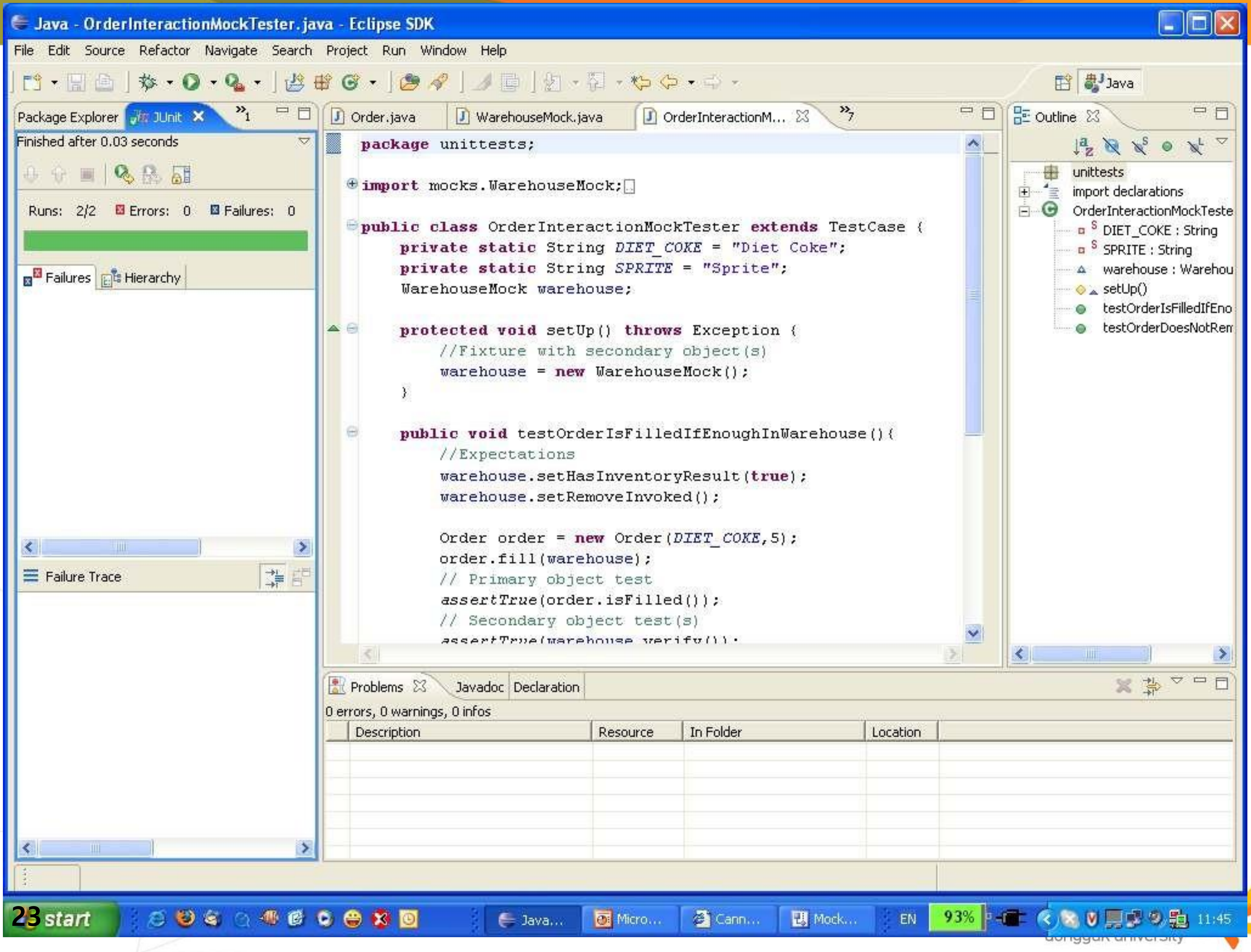
- **인터랙션 기반 테스팅:**

```
public class OrderInteractionMockTester extends TestCase {
        …

        public void testOrderDoesNotRemoveIfNotEnough(){
                //Expectations
                warehouse.setHasInventoryResult(false);

                Order order = new Order(SPRITE,11);
                order.fill(warehouse);
                // Primary object test
                assertFalse(order.isFilled());
                // Secondary object test(s)
                assertTrue(warehouse.verify());
        }

}
```



Finished after 0.03 seconds

Runs: 2/2 Errors: 0 Failures: 0

```
package unittests;

import mocks.WarehouseMock;

public class OrderInteractionMockTester extends TestCase {
    private static String DIET_COKE = "Diet Coke";
    private static String SPRITE = "Sprite";
    WarehouseMock warehouse;

    protected void setUp() throws Exception {
        //Fixture with secondary object(s)
        warehouse = new WarehouseMock();
    }

    public void testOrderIsFilledIfEnoughInWarehouse(){
        //Expectations
        warehouse.setHasInventoryResult(true);
        warehouse.setRemoveInvoked();

        Order order = new Order(DIET_COKE,5);
        order.fill(warehouse);
        // Primary object test
        assertTrue(order.isFilled());
        // Secondary object test(s)
        assertTrue(warehouse.verify());
```

```
unittests
import declarations
OrderInteractionMockTeste
    DIET_COKE : String
    SPRITE : String
    warehouse : Warehou
    setUp()
    testOrderIsFilledIfEno
    testOrderDoesNotRem
```

0 errors, 0 warnings, 0 infos

Description	Resource	In Folder	Location

EasyMock 사용

- Mock 객체의 인터페이스만 정의:

```
public interface Warehouse {

        void add(String product, int i);
        int getInventory(String product);
        boolean hasInventory(String product,int amount);
        void remove(String product, int i);
}
```

EasyMock 사용

- **classpath에 EasyMock jar 파일(easymock.jar) 추가**
- ***import static org.easymock.EasyMock.*;***
- **Mock 객체 생성:**

```
protected void setUp() throws Exception {
        //Fixture with secondary object(s)
        mock = createMock(Warehouse.class);
}
```

EasyMock 사용

- **예상 결과를 주고 테스트 실행:**

```
public void testOrderIsFilledIfEnoughInWarehouse(){
        //Expectations
        expect(mock.hasInventory(DIET_COKE,5)).andReturn(true);
        mock.remove(DIET_COKE,5);
        replay(mock);

        Order order = new Order(DIET_COKE,5);
        order.fill(mock);
        // Primary object test
        assertTrue(order.isFilled());
        // Secondary object test(s)
        verify(mock);
}
```

EasyMock 사용

- 테스트 결과를 검증
  - Mock 객체의 메소드를 호출하지 않는 경우:

```
public void testDemo(){
        mock.remove("cola",2);
        replay(mock);

        verify(mock);
}
```

```
java.lang.AssertionError:
   Expectation failure on verify:
      remove("cola", 2): expected: 1, actual: 0
```

EasyMock 사용

- 테스팅 동작을 검증
  - 호출 및 예외 처리 횟수를 예상할 수 있음

```
expect(mock.foo("input"))
        .andReturn(3).times(3)
        .andThrow(new RuntimeException()).times(4)
        .andReturn(1);
```

  - 융통성 있는 예상

```
int MIN = 1,MAX = 3;

expect(mock.foo("input"))
        .andReturn(3).times(MIN,MAX)
        .andThrow(new RuntimeException()).times(4)
        .andReturn(1).atLeastOnce();
```

설계 #6 – Mock 객체 이용 테스팅

- **다음 Currency 클래스를 EasyMock을 이용하여 테스트 하라.**

```
import java.io.IOException;

public class Currency {

    private String units;
    private long amount;
    private int cents;


    public Currency(double amount, String code) {
        this.units = code;
        setAmount(amount);
    }

    private void setAmount(double amount) {
        this.amount = new Double(amount).longValue();
        this.cents = (int) ((amount * 100.0) % 100);
    }

    public Currency toEuros(ExchangeRate converter) {
        if ("EUR".equals(units)) return this;
        else {
            double input = amount + cents/100.0;
            double rate;
```

Mock 객체 이용 테스팅

```
try {
                rate = converter.getRate(units, "EUR");
                double output = input * rate;
                return new Currency(output, "EUR");
            } catch (IOException ex) {
                return null;
            }
        }
    }

    public boolean equals(Object o) {
        if (o instanceof Currency) {
            Currency other = (Currency) o;
            return this.units.equals(other.units)
                    && this.amount == other.amount
                    && this.cents == other.cents;
        }
        return false;
    }

    public String toString() {
        return amount + "." + Math.abs(cents) + " " + units;
    }
}
```

Mock 객체 이용 테스팅

- Step 1: Exchange 인터페이스를 정의하고
- Step 2: CurrencyTest 클래스를 작성한 후
- Step 3: 테스트 실행
- 제출할 결과물
  - 소스가 있는 프로젝트 파일
  - 실행 결과 캡처 파일