Testgesteuert Entwickeln mit JUnit4, © Christoph Knabe 2002, BHT Berlin, FB VI, 2014-04-15

Beispiel: Schrittweise Entwicklung einer Klasse Stack für Strings

1. Klasse Spezifizieren

```
public class Stack {
    /**Puts element onto the stack*/
    public void push(final String element){}

    /**Returns and removes the youngest element of the stack.
    @throws EmptyExc No element is on the stack*/
    public String pop()throws EmptyExc {return null;}

    /**No element is on the stack*/
    public class EmptyExc extends Exception {}
}
```

2. Testreiber für „last in – first out" (LIFO)-Funktionalität schreiben

```
import static org.junit.Assert.* ;
import org.junit.Before;
import org.junit.Test;
public class StackTest {

    private Stack _stack;

    @Before public void setUp(){
        _stack = new Stack();
    }

    @Test public void popInvertsPush()throws Stack.EmptyExc{
        final String[] elements = {"", "a", "XY", "749398"};
        //Push all:
        for(int i=0; i<elements.length; i++){
22          _stack.push(elements[i]);
        }
        //Pop all:
        for(int i=elements.length-1; i>=0; i--){
26          assertEquals(elements[i], _stack.pop());
        }
    }
}
```

Test-Ausgabe von JUnit bei Aufruf: `java org.junit.runner.JUnitCore StackTest`↵

```
There was 1 failure:
1) popInvertsPush(StackTest)
java.lang.AssertionError: expected:<749398> but was:<null>
    at org.junit.Assert.fail(Assert.java:71)  ...
    at org.junit.Assert.assertEquals(Assert.java:116)
    at StackTest.popInvertsPush(StackTest.java:26)
```

Code-Korrektur in Stack.java:

```
    private final String[] _entries = new String[10];
    private int _fill = 0;
    public void push(final String element){
7       _entries[_fill] = element;
        _fill++;
    }
    public String pop()throws EmptyExc {
        _fill--;
16       return _entries[_fill];
    }
```

3. **Testtreiber ergänzen: Nach 4 push()s und 4 pop()s muss Stack wieder leer sein**

```
    @Test public void emptyAfterPushsAndPops()throws Stack.EmptyExc{
        popInvertsPush();
        try{
33          _stack.pop();
            fail("Stack.EmptyExc expected");
        } catch ( Stack.EmptyExc expected ){}
    }
```

Test-Ausgabe von JUnit:

There was 1 failure:
1) emptyAfterPushsAndPops(StackTest)
java.lang.ArrayIndexOutOfBoundsException: -1
    at Stack.pop(Stack.java:**16**)
    at StackTest.emptyAfterPushsAndPops(StackTest.java:**33**)

Code-Korrektur in Stack.java:

```
    public String pop()throws EmptyExc {
        _fill--;
        if(_fill<0){throw new EmptyExc();}
        return _entries[_fill];
    }
```

4. **Testtreiber ergänzen: pop() auf leerem Stack läßt ihn leer**

```
    @Test public void popOnEmptyDoesNotModify()throws Stack.EmptyExc{
        try{_stack.pop(); fail("Stack.EmptyExc expected");}
        catch(Stack.EmptyExc expected){}
41      popInvertsPush();
        try{_stack.pop(); fail("Stack.EmptyExc expected");}
        catch(Stack.EmptyExc expected){}
    }
```

Test-Ausgabe von JUnit:

There was 1 failure:
1) popOnEmptyDoesNotModify(StackTest)
java.lang.ArrayIndexOutOfBoundsException: -1
    at Stack.push(Stack.java:**7**)
    at StackTest.popInvertsPush(StackTest.java:**22**)
    at StackTest.popOnEmptyDoesNotModify(StackTest.java:**41**)

Code-Korrektur in Stack.java:

```
    public String pop()throws EmptyExc {
        if(_fill<=0){throw new EmptyExc();}
        _fill--;
        return _entries[_fill];
    }
```

Best Practice: Testsuite in eigenem Verzeichnisbaum parallel zum Quellverzeichnisbaum

Maven führt bei `mvn test` alle Klassen, deren Name auf `Test` endet, als Testtreiber aus.
Eclipse führt mittels *MausRechts* > *Run As* > *JUnit Test* alle Testtreiber im Verzeichnis aus.

Test-Suite zum Ausführen von Testtreibern in definierter Reihenfolge:

```
package fb6._any.ut;
import org.junit.runner.RunWith;
import org.junit.runners.Suite;
/**Testsuite zum zusammengefassten Ausführen bestimmter Tests.*/
@RunWith (Suite.class)
@Suite.SuiteClasses ({
    FileUtilTest.class,
    StringUtilTest.class
})
public class TestsAnyUt {}
```