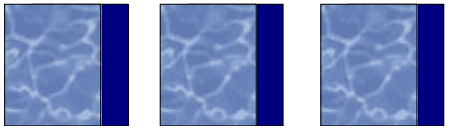
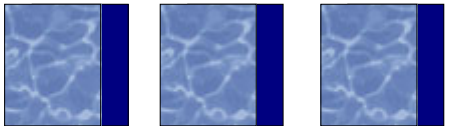
JUnit

Unit testing unter Java



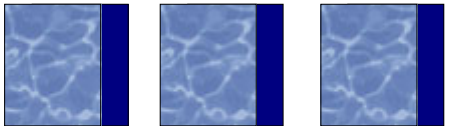
Was ist Junit?

- einfaches Framework zum Schreiben von wiederholbaren Tests
- Besonders geeignet für unit testing
- Erlaubt Hierarchie von Testsuites



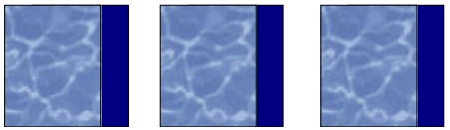
Schreiben eines Test Case

1. Definiere eine Subclass von `TestCase`
2. Überschreibe die `setUp()` Methode um Testobjekt(e) zu initialisieren und zustarten (z.B. Datenbankverbindung)
3. Überschreibe die `tearDown()` Methode um Testobjekt(e) nach dem Test wieder freizugeben bzw. zu beenden
4. Definiere eine oder mehrere `testXXX()` Methode(n)
5. Definiere eine suite() factory−Methode, die eine TestSuite kreiert, die alle `testXXX()` Methoden des `TestCase`
6. Definiere eine main() Methode, die den `TestCase` abarbeitet



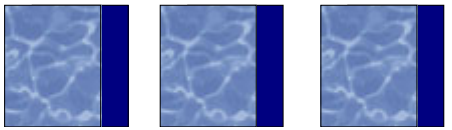
Beispiel

```
import junit.framework.Test;
import junit.framework.TestCase;
import junit.framework.TestSuite;

public class ShoppingCartTest extends TestCase {

        private ShoppingCart _bookCart;

        public ShoppingCartTest(String name) {
                super(name);
        }

        protected void setUp() {

                _bookCart = new ShoppingCart();

                Product book = new Product("Extreme Programming", 23.95);
                _bookCart.addItem(book);
        }

        protected void tearDown() {
                _bookCart = null;
        }
```



Beispiel

```
public void testEmpty() {
        _bookCart.empty();
        assert(_bookCart.isEmpty());
}


public void testProductAdd() {
        Product book = new Product("Refactoring", 53.95);
        _bookCart.addItem(book);
        double expectedBalance = 23.95 + book.getPrice();
        double currentBalance = _bookCart.getBalance();
        double tolerance = 0.0;
        assertEquals(expectedBalance, currentBalance, tolerance);
        int expectedItemCount = 2;
        int currentItemCount = _bookCart.getItemCount();
        assertEquals(expectedItemCount, currentItemCount);
}

public void testProductNotFound() {
        try {
                Product book = new Product("Ender's Game", 4.95);
                _bookCart.removeItem(book);
                fail("Should raise a ProductNotFoundException");
        } catch(ProductNotFoundException pnfe) {
                // should never get here   }
}
```

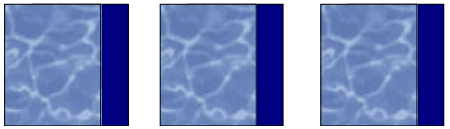


Beispiel

```
public static Test suite() {
    TestSuite suite = new TestSuite(ShoppingCartTest.class);

    // Alternatively, but prone to error when adding more
     // test case methods...
     //
    // TestSuite suite = new TestSuite();
    // suite.addTest(new ShoppingCartTest("testEmpty"));
    // suite.addTest(new ShoppingCartTest("testProductAdd"));
    // suite.addTest(new ShoppingCartTest("testProductRemove"));
    // suite.addTest(new ShoppingCartTest("testProductNotFound"));
    //

    return suite;
  }

     public String toString() {
            return name();
     }

     public static void main(String args[]) {
            String[] testCaseName = {ShoppingCartTest.class.getName()};
            //junit.textui.TestRunner.main(testCaseName);
            junit.swingui.TestRunner.main(testCaseName);
            //junit.ui.TestRunner.main(testCaseName);
     }
}
```

Schreiben einer Test Suite

1. Definiere eine Subclass von `TestCase`
2. Definiere eine `suite()` factory−Methode, die eine `TestSuite` kreiert, die alle `TestCase` Instanzen und alle `TestSuite` Instanzen beinhaltet
3. Definiere eine `main()` Methode, die den `TestCase` abarbeitet



Beispiel

```
public class EcommerceTestSuite extends TestCase {

    public EcommerceTestSuite(String name) {
        super(name);
    }

    public static Test suite() {

        TestSuite suite = new TestSuite();

        //
        // The ShoppingCartTest we created above.
        //
        suite.addTest(ShoppingCartTest.suite());

        //
        // Another example test suite of tests.
        //
        suite.addTest(CreditCartTestSuite().suite());

        return suite;
    }

    public static void main(String args[]) {
        String[] testCaseName = {EcommerceTestSuite.class.getName()};
        //junit.textui.TestRunner.main(testCaseName);
        //junit.swingui.TestRunner.main(testCaseName);
        junit.ui.TestRunner.main(testCaseName);
    }
}
```

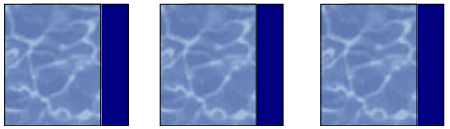
Starten eines Tests

- Mehrere UserInterfaces:
  - AWT−basiert: `java.ui.TestRunner`
  - Swing−basiert: `java.swingui.TestRunner`
  - Textuell: `java.textui.TestRunner`
- Entweder TestCase starten:

```
java ShoppingCartTest
```

- Oder TestSuite starten:

```
java EcommerceTestSuite
```



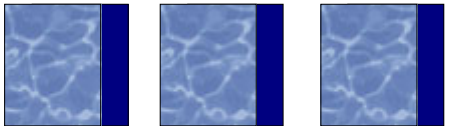
Organisation von Tests

1. Erstelle Test cases im gleichen package wie der Code, auf den sich der Test bezieht

2. Erstelle für jedes Java−Package eine TestSuite−class, die alle Tests für das Package enthält

3. Erstelle higher−level und lower−level TestSuites in den anderen Packages bzw. in den Subpackages der Applikation

4. Der build process sollte das kompilieren der TestSuites beinhalten. Dadurch sind die Tests immer up−to−date!



Beispiel

- `MasterTestSuite` – The top–level test suite
  - `SmokeTestSuite` – Structural integrity tests
    - `EcommerceTestSuite`
      - `ShoppingCartTestCase`
      - `CreditCartTestSuite`
        - `AuthorizationTestCase`
        - `CaptureTestCase`
        - `VoidTestCase`
      - `UtilityTestSuite`
        - `MoneyTestCase`
    - `DatabaseTestSuite`
      - `ConnectionTestCase`
      - `TransactionTestCase`
  - `LoadTestSuite` – Scalability tests
    - `DatabaseTestSuite`
      - `ConnectionPoolTestCase`
    - `ThreadPoolTestCase`