

Test

Background:

Læs let Larman ch. 21 p. 385-389

Læs let UPEDU (www.upedu.org) the test disciplines [see testing](#)

Orientering Wiki: the V-model ([http://en.wikipedia.org/wiki/V-Model_\(software_development\)](http://en.wikipedia.org/wiki/V-Model_(software_development)))

Types (or Stages) of Testing

- Developer Testing (tech.walkthrough)
 - Normal testing by the developer / programmer – to see it do work
- Independent and Stakeholder Testing (reviews)
 - Independent Testing denotes the test design and implementation that it is most appropriate for someone independent from the team of developers to do.
- Unit Tests
 - Systematic automatic test of a unit (testing from a black box view)
- Integration Test
 - integration testing is performed to ensure that the components in combination do work (e.g. that classes across packages do work)
- System Test
 - System testing is done when the software is functioning as a whole. Do the whole system works
- Acceptance Test
 - The users do the testing and accepting as a final test action prior to deploying the software. Check that all use-cases and all non-functional requirements work

Unit testing

See the artifact: test case

http://www.upedu.org/process/artifact/ar_tstcs.htm

Especially you can see the guidelines of test case

http://www.upedu.org/process/gdlines/md_tstcs.htm

From the guidelines

You can see how to set up (or derive) test cases o test your **use-cases** as well as the **supplementary requirements** and for **unit test** and for **Acceptance test**

Below is the Unit testing discussed. When talking of unit tests you can divide them into

White box testing – where you check all programming lines have been executed with an accepted result

Black box testing – where you check all methods have been executed and all parameter boundaries have been checked – of course again with an accepted result

From [Upedu \(click on black box test\)](#) you can read in more detail how to construct (derive) different test cases.

Here is an example of the black box testing – which is the most common:

We have the class Person

ID a number between 1000-99999
Name a text which is not null and at least 4 character long
Phone a number of 8 digits

We have to set up all ‘possible’ input values
(normal values, values on the boundary, values outside boundary and illegal values)

Test case #	Description of test case	Expected value	Passed successfully
1	Default constructor	Object created	
2	Set ID – value 999	ArgumentException // error	
3	Set ID – value 1000	ID == 1000	
4	Set ID – value 99999	ID == 99999	
5	Set ID – value 100000	ArgumentException // error	
6	Set ID – value 5678	ID == 5678	
7	Set ID – value -5	ArgumentException // error	
8	Set Name – value null	ArgumentException // error	
9	Set Name – value empty (“”)	ArgumentException // error	
10	Set Name – value not empty but less than 4 value “123”	ArgumentException // error	
11	Set Name – value not empty and 4 value “1234”	Name == “1234”	
12	Set Name – value not empty and 15 value “123456789012345”	Name == “123456789012345”	
13	Set Phone – value 9999999	ArgumentException // error	
14	Set Phone – value 10000000	Phone == 10000000	
15	Set Phone – value 99999999	Phone == 99999999	
16	Set Phone – value 100000000	ArgumentException // error	
17	Set Phone – value 56781234	Phone == 56781234	
18	Set Phone – value -5	ArgumentException // error	
19	Constructor(2222,“Susanne”,12345678)	ID == 2222 Name == “Susanne” Phone == 12345678	
20	Constructor(00999,“Susanne”,12345678)	ArgumentException // error	

21	Constructor(2222,null,12345678)	ArgumentException // error	
22	Constructor(2222,"Per",12345678)	ArgumentException // error	
23	Constructor(2222,"Susanne",1234567890)	ArgumentException // error	