

Test

Background:

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

Just to give a feeling of testing is a broad approach to ensure the system is well working.

Different levels of Testing

- Developer Testing (technical 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 (the subject of this paper)**
 - Systematic automatic test of a unit (testing from a black box view or white box)
- Integration Test
 - integration testing is performed to ensure that the components in combination do work (e.g. that classes across packages or subsystems do work)
- System Test
 - System testing is done when the software is functioning as a whole. Does 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 Test

A more detailed description of UnitTest is following, but first 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. See [Wiki : whitebox testing](#)

Black box testing – where you check all methods have been executed and all parameter boundaries have been checked – of cause again with an accepted result. See [Wiki : blackbox testing](#)

Additional reading to better understand Blackbox and Whitebox testing :

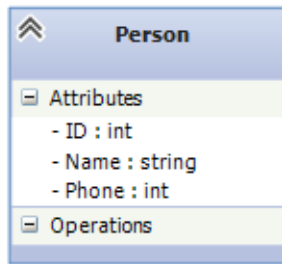
http://www.cs.unh.edu/~it666/reading_list/Defense/blackbox_vs_whitebox_testing.pdf .

How to do UnitTest In Visual Studio

<https://docs.microsoft.com/en-us/visualstudio/test/walkthrough-creating-and-running-unit-tests-for-managed-code?view=vs-2019>

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 **just 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,“Jakob”,12345678)	ID == 2222 Name == “Susanne” Phone == 12345678	
20	Constructor(00999,“ Jakob”,12345678)	ArgumentException // error	
21	Constructor(2222,null,12345678)	ArgumentException // error	
22	Constructor(2222,“Per”,12345678)	ArgumentException // error	
23	Constructor(2222,“ Jakob”,1234567890)	ArgumentException // error	

We could furthermore have a repository of persons e.g. 'PersonRepository'

This repository has e.g. five methods covering CRUD functionality

- `public List<Person> GetAll()`
- `public Person GetById(int id)`
- `public Person Add(Person person)`
- `public Person Delete(int id)`
- `public Person Update(int id, Person person)`

Remember here to test for different parameters as well as different return values

Test case #	Description of test case + parameter	Expected return value	Passed successfully
1	GetAll	Empty list	
2	GetAll	List of one person	
3	GetAll	List of several persons	
4	GetById id exist	the person object with the id	
5	GetById id do not exist	KeyNotFoundException	
6	Add person is null	ArgumentNullException the list is the same	
7	Add person is valid	The person object possible with autogenerated id The list is one element enlarged	
8	Delete id exist	the person object with the id the list is one person short	
9	Delete id do not exist	KeyNotFoundException the list is the same	
10	Update id do not exist	KeyNotFoundException the list is the same	
11	Update id exists but person is null	ArgumentNullException the list is the same	
12	Update id exists and person is valid	the person object with the id, and the updated values the object in the list are updated	