

## Testing (Δοκιμές / Έλεγχοι)



However it is a bad practice to leave testing for the end



Testing is a form of insurance.

It costs more to repair software bugs when people are depending on the programs than in earlier stages before the systems are in use.

- The purpose is not to demonstrate that the system is free of errors;
- The purpose is to detect as many errors as possible ... :)

#### **Testing Philosophy**

- It is dangerous to test early modules without an overall testing plan
- · It may be difficult to reproduce sequence of events causing an error
- Testing must be done systematically and results documented carefully

U. of Crete, Information Systems Analysis and Design

Yannis Tzitzikas

3



# Types/Stages of Testing

- Unit testing
  - Tests each module to assure that it performs its function

### Integration testing

- Tests the interaction of modules to assure that they work together

### System testing

 Tests to assure that the software works well as part of the <u>overall</u> system

### Acceptance testing

- Tests to assure that the system serves <u>organizational needs</u>



- Black Box Testing
  - Focuses on whether the unit meets requirements stated in specification
- <u>White-Box</u> Testing
  - Looks inside the module to test its major elements

### **Integration** Testing

- User interface testing
  - Tests each interface function
- Use-case testing
  - Ensures that each use case works correctly
- Interaction testing
  - Tests each process in a step-by-step fashion
- System interface testing
  - Ensures data transfer between systems

U. of Crete, Information Systems Analysis and Design

Yannis Tzitzikas



•

## System Testing

- <u>Requirements</u> Testing
  - Ensures that integration did not cause new errors
- Usability Testing
  - Tests how easy and error-free the system is in use
- Security Testing
  - Assures that security functions are handled properly
  - Performance Testing
    - Assures that the system works under high volumes of activity
- Documentation Testing
  - Analysts check that documentation and examples work properly

### Acceptance Testing

- <u>Alpha</u> Testing
  - Repeats tests by users to assure they accept the system
- Beta Testing
  - Uses real data, not test data



### Case study: Java Unit Testing



### Case: Testing Java Code

 JUnit is a freely available Java unit test framework. extensively used in the Java community because of its elegance of design and ease of use.

#### • Why use a testing framework?

- Using a testing framework is beneficial because it forces you to explicitly declare the expected results of specific program execution routes.
- When debugging it is possible to write a test which expresses the result you are trying to achieve and then debug until the test comes out positive.
- By having a set of tests that test all the core components of the project it is possible to modify specific areas of the project and immediately see the effect the modifications have on the other areas by the results of the test, hence, side-effects can be quickly realized.



– JUnit promotes the idea of first testing then coding, in that it is possible to setup test data for a unit which defines what the expected output is and then code until the tests pass. It is believed by some that this practice of "test a little, code a little, test a little, code a little..." increases programmer productivity and stability of program code whilst reducing programmer stress and the time spent debugging. It also integrates with Ant (http://jakarta.apache.org/ant)

U. of Crete, Information Systems Analysis and Design	Yannis Tzitzikas	
Short Introduction to JUnit		
Suppose you want to test the following code		
<pre>public class Math {     static public int add(int a, int b) {         return a + b;     } }</pre>	To test this code, you need a second Java class that will	
<pre>import j unit.framework.*; public class TestMath extends TestCase {     public void testAdd() {         int num1 = 3;         int num2 = 2;         int total = 5;         int sum = 0;         sum = Math.add(num1, num2);         assertEquals(sum, total);     } }</pre>	<ol> <li>import junit.framework.*</li> <li>extend TestCase.</li> </ol>	

#### U. of Crete, Information Systems Analysis and Design



### Short Introduction to JUnit

- Notice that the routine is named testAddNumbers. This convention tells you that the routine is supposed to be a test and that it's targetting the "add" functionality.
- The last step is how to run your JUnit tests. You can do this several ways, including your command line, Eclipse or the JUnit Test Runner, or plain Ant. To run a Junit test with an Ant script, add this to your Ant script:

<pre><junit haltonfailure="yes" printsummary="yes" showoutput="yes"></junit></pre>
<cl asspath=""></cl>
<pre><pathelement path="\${build}"></pathelement></pre>
<pre><batchtest fork="yes" todir="\${reports}/raw/">         <formatter type="xml"></formatter>         <fileset dir="\${src}"></fileset></batchtest></pre>

U. of Crete, Information Systems Analysis and Design



### Assertion Statements

There is a list of the different types of assertion statements that are used to test your code. Any Java data type or object can be used in the statement. These assertions are taken from the JUnit API.

- **assertEquals**(expected, actual)
- assertEquals(message, expected, actual)
- assertEquals(expected, actual, delta) used on doubles or floats, where delta is the difference in precision
- **assertEquals**(message, expected, actual, delta) used on doubles or floats, where delta is the difference in precision
- **assertFalse**(condition)
- assertFalse(message, condition)
- assertNotNull(object)
- assertNotNull(message, object)

Yannis Tzitzikas

11



# Another example

```
// A Class that adds up a string based on the ASCII values of its
// characters and then returns the binary representation of the sum.
public class BinString {
  public BinString() {}
   public String convert(String s) {
      return bi nari se(sum(s));
   }
   public int sum(String s) {
      if(s=="") return 0;
      if(s.length()==1) return ((int)(s.charAt(0)));
      return ((int)(s.charAt(0)))+sum(s.substring(1));
   }
   public String binarise(int x) {
      if(x==0) return "";
      if(x%2==1) return "1"+bi nari se(x/2);
      return "0"+bi nari se(x/2);
   }
}
```



```
import junit.framework.*;
public class BinStringTest extends TestCase {
   private BinString binString;
   public BinStringTest(String name) {
      super(name);
   }
   protected void setUp() {
      binString = new BinString();
   }
   public void testSumFunction() {
      int expected = 0;
      assertEqual s(expected, binString.sum(""));
      expected = 100;
      assertEqual s(expected, binString.sum("d"));
      expected = 265;
      assertEqual s(expected, binString.sum("Add"));
   }
```

U. of Crete, Information Systems Analysis and Design

Yannis Tzitzikas

```
15
```



```
the test class (cont)
```

```
public void testBinariseFunction() {
    String expected = "101";
    assertEquals(expected, binString.binarise(5));
    expected = "11111100";
    assertEquals(expected, binString.binarise(252));
  }
  public void testTotalConversion() {
    String expected = "1000001";
    assertEquals(expected, binString.convert("A"));
  }
}
```



- TestCase allows us to use the method setUp to set up any necessary variables or objects (setUp is called before the evaluation of each test)
  - Note that *bi nStri ng* was already declared at the top of the file like usual with pri vate Bi nStri ng bi nStri ng;
- setUp has a brother called tearDown which is called after the evaluation of each test and can be used to dereference variables or whatever so that each test may be performed without issuing side-effects that may affect the outcome of the other tests.

U. of Crete, Information Systems Analysis and Design

Yannis Tzitzikas

17



- Using an IDE (like Eclipse, NetBeans) you can create tests very quickly and easily
- Tests running can be automated



🗑 NetBeans IDE 4.0 - cse409netbeans			-0×
File Edit View Build Run Refactor Versioning	Tools Window Help		
🖻 🖻 🖻 💕 🖆 📁	Javadoc Index Search Shift+F1 Internationalization		
Projects Files () ×	Junit Tests 🔹 🕨	Create Tests Ctrl+Alt+J	
Contractions	Auto Comment	Open Test Ctrl+Alt+K	
E dist	Add to Favorites		
🕀 🛅 nbproject	Apply Diff Patch		
test     imanifest.mf     ise409netbeans - Source Package	Palette Manager Java Platform Manager Library Manager		
E cse409	Setup Wizard Keyboard Shortcuts Options		
B C jce B C uti keyfile.txt	Update Center		
<u>د الا المعالمة المعا</u>			
Output - cse409netbeans (jar)			<b>₽</b> ×
			101



### case NetBeans creating test 2/2

Example - NetBeans ID	DE 6.0.1								
<u>Edit View N</u> avigate	Source Refactor Build	<u>R</u> un	Profile Versioning Tools Window	v <u>H</u> elp					
<u>*</u> * * * * * *	🔀 💾 🗐 🏓 (	G :	(default config)	<u>i</u>	3	\$ - 🚯 -			
rojects	s								
🕀 🔓 Libraries	New	• Ju	JUnit Test						
🕀 🍯 Test Librarie:	Local History	•	Persistence Unit Java Class						
	Eind Ctrl+		Java Package						
	Paste Ctrl+	/	Java Interface JPanel Form	C	🕽 Nev	v JUnit Test			×
	Tools	• 💼	JFrame Form		Steps	<u>.</u>	Name and I	Location	
	Properties	 	Entity Class Entity Classes from Database		1. ( 2. j	Choose File Type Name and Location	Class Name:	NewEmptyJUnitTest	
		9	Web Service Client				Project:	Example	
		2	Other				Location:	Test Packages	•
							Package:	ExampleTesting	•
							Created File:	$\label{eq:c:NetBeansWorkspace} C: \end{tabular} C: \end{tabular} Example \end{tabular} to the the the test \end{tabular} to the test \end{tabular}$	
							Generated C	ode Generated Comments	
							🔽 Test	Initializer 📝 Source Code Hints	
							🗸 Test	Finalizer	
								<back next=""> Finish Cancel Help</back>	



### case NetBeans selecting the elements for which tests should be created

We are now presented with a Create Tests window. You will notice several check boxes. If the methods that you are trying to test are not static meaning that they need an instance of the object to be called then you will probably want to specify a **setUp** method for the test case. This will be where you can instantiate your object before running the tests. Also if your tests allocate any special resources you may want to select **tearDown** so you can properly dispose of the resources. After we are done with this you may click the next button.

ethod Access Levels	Optional Classes		
Public	Generate Test Suites		
Protected	Optional Code		
Package Private	l⊽ set <u>U</u> p		
lass Types	I⊽ tear <u>D</u> own		
Package Private Classes	Default Method Bodies		
Abstract Classes	Optional Comments		
Exception Classes	☑ JavaDoc Comments		
	Source Code Hints		

• An alternate way to create tests for individual classes is to right click on the file that you wish to create a test for then goto JUnit Tests and then Create Tests. This will allow you to create a single test for a particular class.

U. of Crete, Information Systems Analysis and Design

Yannis Tzitzikas

21





### case Eclipse creating test 1/2

틎 Java - Eclipse	Plat	orm			
File Edit Sou	irce	Refactor Navigate Search	Project Run Window	Help	
		ᅓ ▾ () ▾ 🂁 ▾ 🛛 🖑 📽	G • 😕 🖨 🔗 🔮	•	
🛱 Packa 🙁	L.	Hierar 🗖			
$\Leftrightarrow \Rightarrow $		New	•	<mark>₁}</mark> ♦	Java Project
▲ 😂 Example		Go Into			Project
D 🛋 JRE S		Open in New Window		₿	Package
⊳ 🎏 GapMar		Open Type Hierarchy	F4	C	Class
		Show In	Alt+Shift+W ►	G	Interface
		Сору	Ctrl+C	G	Enum
		Copy Qualified Name		@	Annotation
	Ê	Paste	Ctrl+V	₩	Source Folder
	×	Delete	Delete		Folder
	<u>.</u>	Remove from Context	Ctrl+Alt+Shift+Down		Untitled Text File
		Build Path	•	E	JUnit Test Case
		Source	Alt+Shift+S ►	Ċ	Task
		Refactor	Alt+Shift+1 ►	<b>1</b>	Example
	è	Import		=9	Other
	4	Export			
	4	n r 1			

### In Eclipse we must add the JUnit Library explicitly

Properties for Example		
type filter text	Java Build Path	⇔ • ⇔ •
Resource Builders Java Build Path Java Code Style Java Code Style Java Editor Javadoc Location Project References Run/Debug Settings Task Repository Task Tags Validation		Add JARs Add External JARs Add Variable Add Library Add Class Folder Edit Remove Migrate JAR File
0		OK Cancel
	Yannis Tzitzikas	23

U. of Crete, Information Systems Analysis and Design



### case Eclipse creating test 2/2

😂 New JUnit Test	Case	_ <b>D _</b> X
JUnit Test Ca This package r start with a lov	<b>Se</b> name is discouraged. By convention, package names usuall vercase letter	y E
New JUnit 3 te	st 💿 New JUnit 4 test	
Source folder:	Example/src	Browse
Package:	ExampleTesting	Browse
Name:	NewJunitTest	
Superclass:	junit.framework.TestCase	Browse
Do you want to a	Ups would you like to create: setUpBeforeClass() tearDownAfterClass() setUp() V tearDown() constructor dd comments as configured in the <u>properties</u> of the curren Generate comments	t project?
Class under test:		Browse
0	< Back Next > Finish	Cancel

0	• Q. • 🖄 🕸 🎯 •	ا 🤔	🖻 🛷 🛛 🖢 🔻 🖗	• * <b>\$</b> \$
Ju	1 NewJunitTest			
J	2 Simple			
	Run As	2	1 Java Applet	Alt+Shift+X, A
0	Open Run Dialog	J	2 Java Application	Alt+Shift+X, J
	Organize Favorites	JU	3 JUnit Test	Alt+Shift+X, T