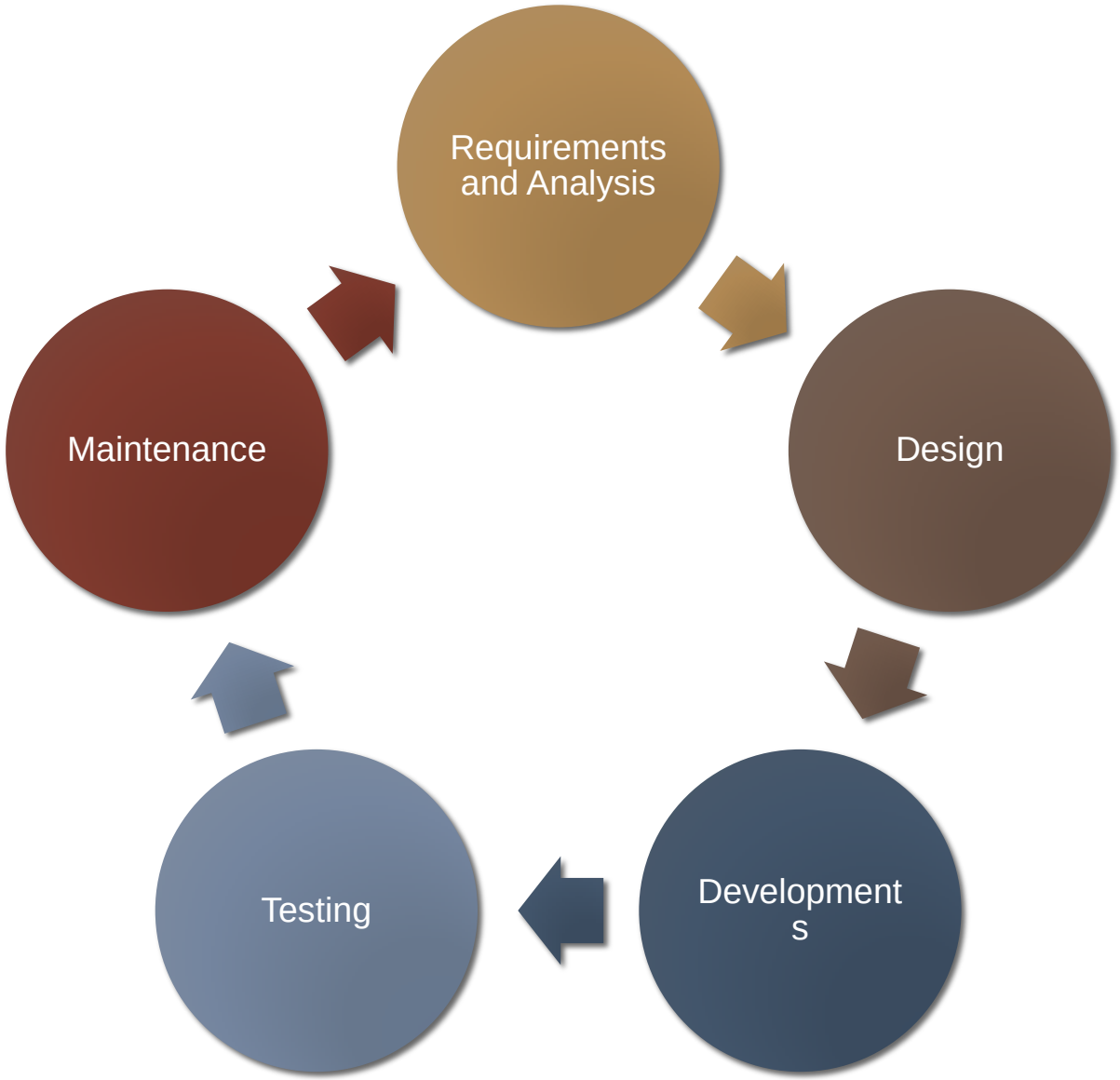
(11224) INTRODUCTION TO SOFTWARE ENGINEERING

Lecture 3: Software Testing

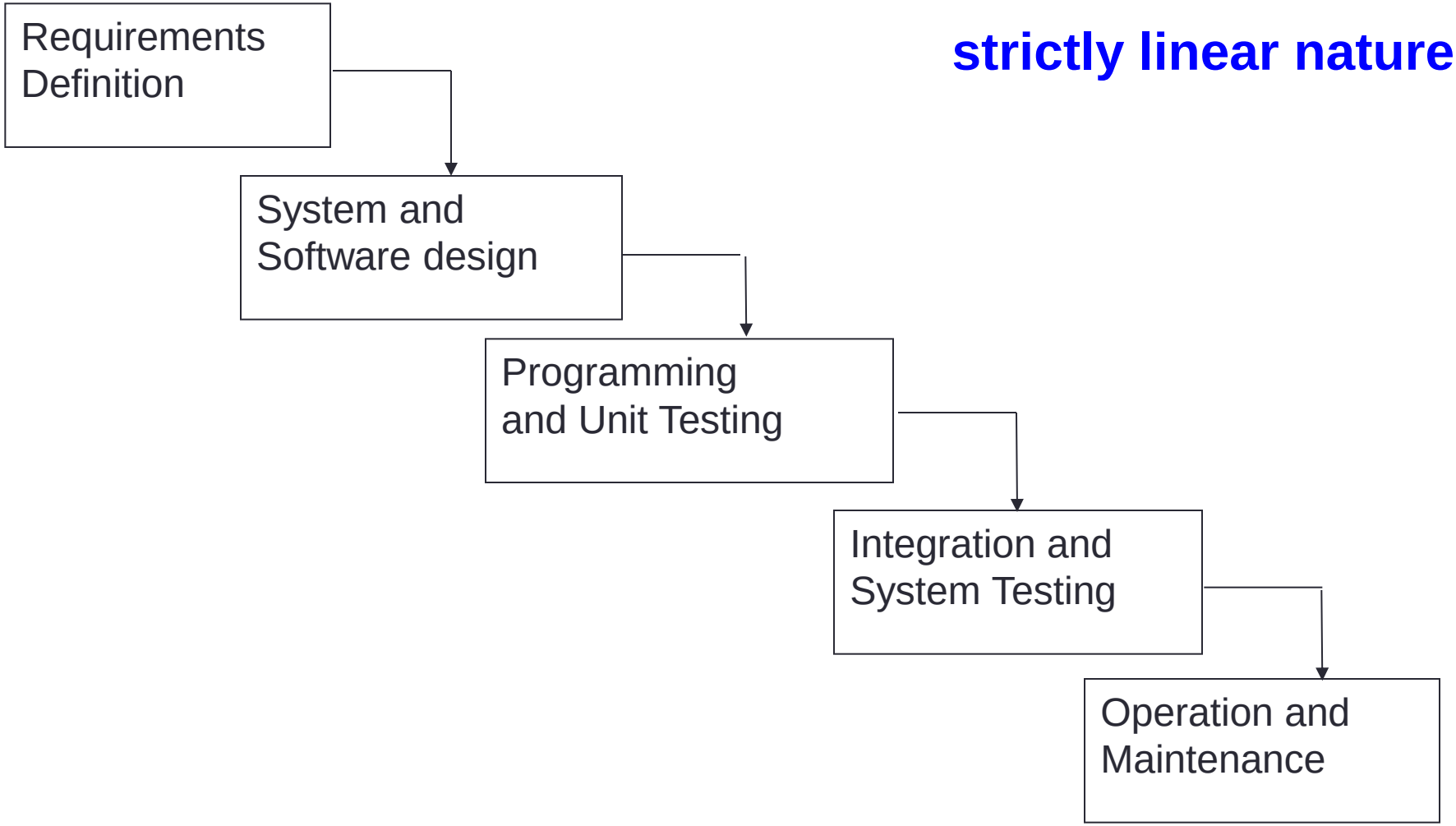
Shereen Fouad

Reminder of Previous Lecture

Software Processes

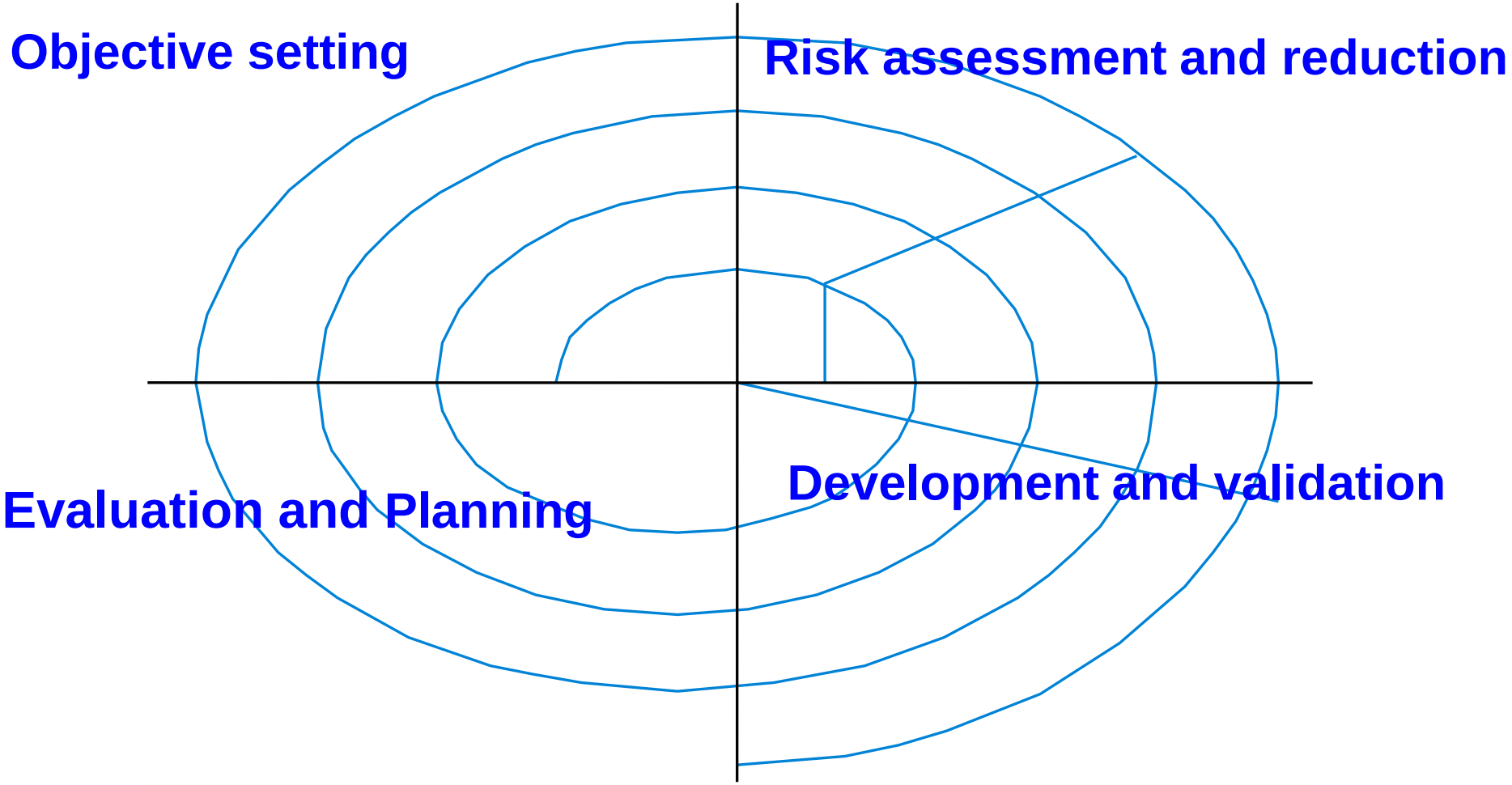


Waterfall Model



strictly linear nature !

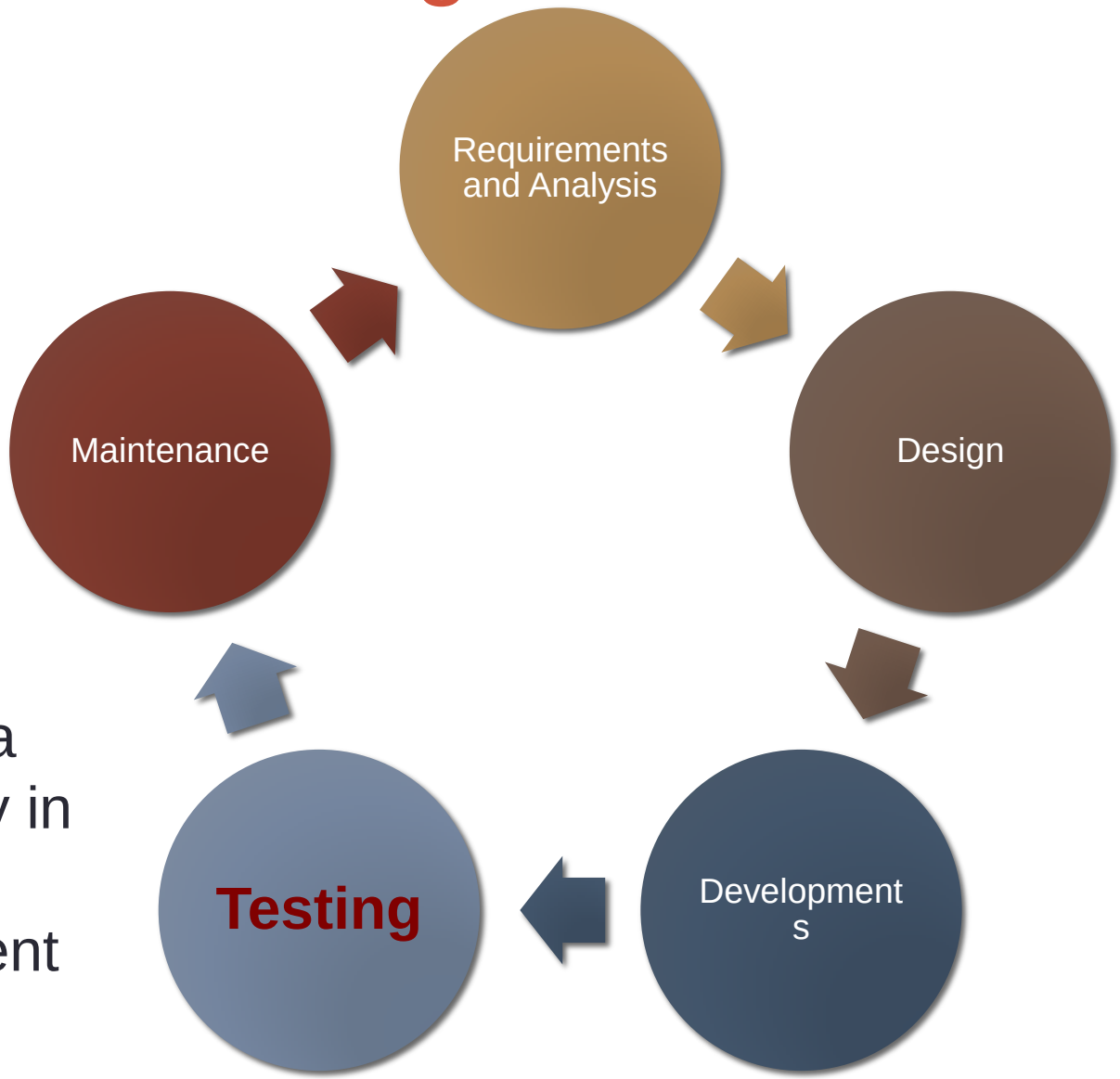
Spiral Model Sectors



Overview

- Software testing definition
- Why testing?
- Goals of testing
- Levels of testing
- Basic Steps of a Test
- Automated Testing (TestNG)

Software Testing



Testing is a key activity in software development process

Software Testing

- Discover program defects before it is put into use
- Check the functionality of the application whether it is working as per requirements
- Can reveal the presence of errors NOT their absence
- It involves executing the program using artificial data
- Part of a more general verification and validation process

- **Can you tell the difference between verification and validation??**

Note that !!

- For the purposes of this module, we will assume throughout that we are only concerned with testing in an object-oriented programming context.

Why Testing?

http://en.wikipedia.org/wiki/Mars_Polar_Lander

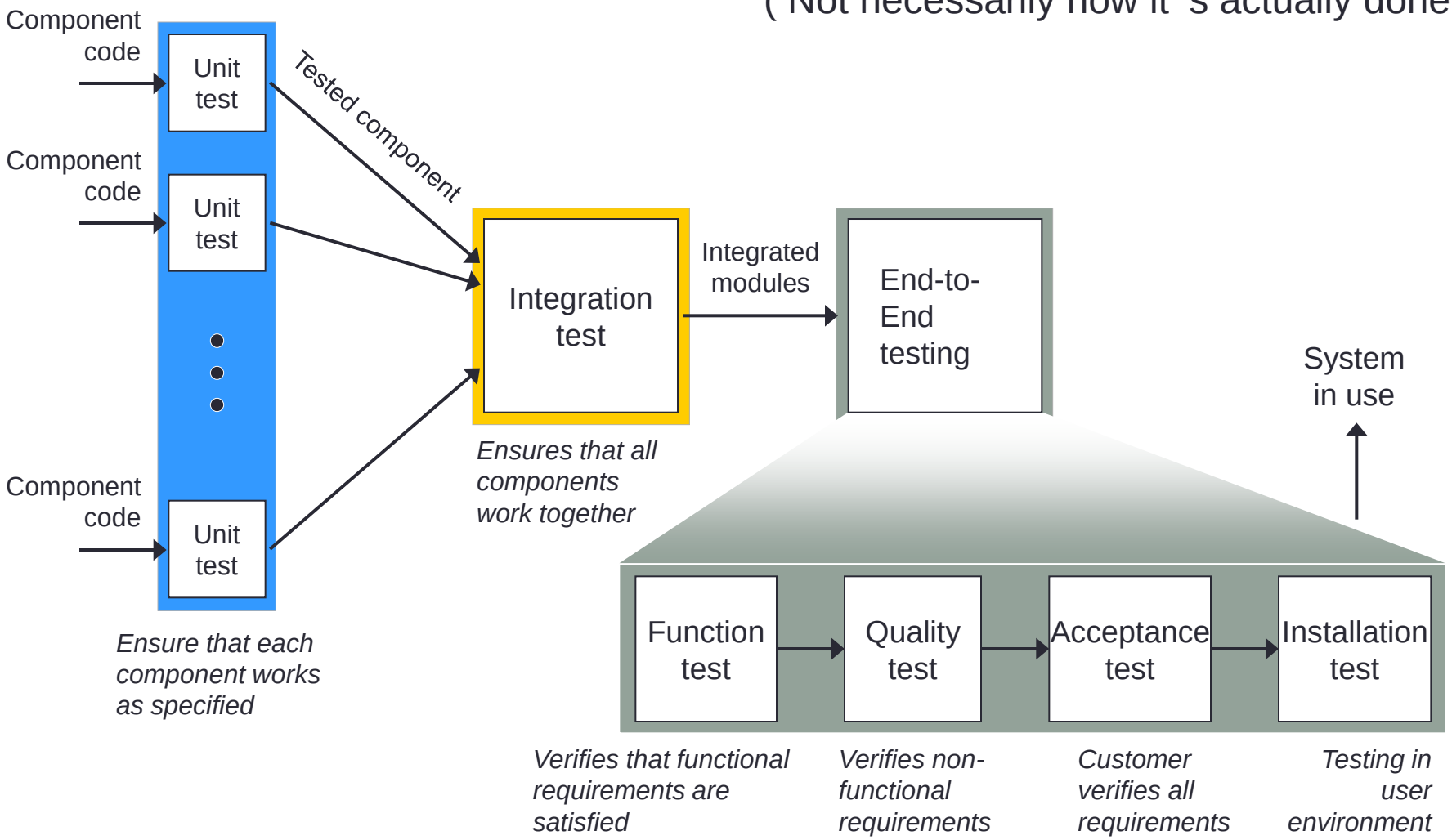
- The **Mars Polar Lander**, was robotic spacecraft lander launched by NASA in 1999 to study the soil and climate on planet Mars.
- After the descent phase was expected to be complete, the lander failed to reestablish communication with Earth.
- The error was traced to a single bad line of software code.

Goals of Testing

- Discover situations in which
  - the behavior of the software is incorrect,
  - Undesirable or
  - does not conform to its specification
- Demonstrate the software meets its requirements.
- Establish confidence.
- Improve the quality of the product developed.

Levels of Testing

(Not necessarily how it's actually done!)



Unit Testing

- Ensuring that a single class/method works correctly.
- The class should be tested in isolation
- Carried out by developers
- Confirm that subsystems is correctly coded and carries out the intended functionality
- **Black box testing**
  - Choose test data *without* looking at implementation
  - Test the input/output behavior
- **Glass box (white box) testing**
  - Choose test data *with* knowledge of implementation
  - Test the internal logic of the subsystem or object

Integration Testing

- Ensuring that different logical parts or modules of code work correctly together.
- Carried out by developers
- Test the *interface* among the subsystem
- Usually slower and more computationally intensive than unit tests.

End-to-End Testing

- Ensure that the whole system from the point of view of the end user works correctly.
- Determine if the system meets the requirements
- End-to-End testing requires no access to or understanding of the code base

Who Performs the End to End testing?

developer

independent tester

Who Performs the End to End testing?

developer

Understands the system

but, will test "gently"

and, is driven by "*delivery*"

independent tester

Must learn about the system,

but, will attempt to break it

and, is driven by *quality*

SUT and DOC

- **System Under Test (SUT)**
- When testing, the subject of the test, called the System under Test, or SUT, is the specific item being tested.
- **Depended On Component (DOC)**
- Part of the system that we are not testing in this particular test but which the SUT depends on.

Basic Steps of a Test

1) Choose input data / configuration

2) Define the expected outcome

3) Run program / method against the input and record the results

4) Examine results against the expected outcome

Automated Testing

- Manual testing is
  - error prone and boring
  - It won’t be done properly
- Testing should be automated.
  – A *test framework* runs the tests and checks the results.

Quiz !!

- **The objective of testing is**

a) Debugging
b) To uncover errors
c) To gain modularity
d) To analyze system

- **Acceptance testing is part of the**

a) Unit testing
b) End to End testing
c) Black box testing
d) Integration Testing

TestNG

- Open source automated testing framework
- **NG** of Test**NG** means **N**ext **G**eneration.
- Testing framework inspired from JUnit but introducing some new functionalities that make it more powerful, flexible and easier to use.
- JUnit are Initially designed to enable unit testing only and it cannot do dependency testing.
- TestNG is designed to cover all categories of tests: Unit, functional, end-to-end, integration, etc.

http://www.tutorialspoint.com/testng/testng_quick_guide.htm

Installing the TestNG Plugin

- Before we can download and start using TestNG, make sure you have Java JDK5 or above is installed on your system.
- Instructions for installing the TestNG Plugin in Eclipse can be found at
  - http://testng.org/doc/download. html.
- Basic use of this plugin is described at
  - http://testng.org/doc/eclipse.html#eclipse-installation.

Eclipse
integrated development environment (IDE)

eclipse.org

Pearson - Obje... | research.micro... | SE9 Presentations | software testin... | www.inf.ed.ac.... | www.google.co... | www.google.co... | https://www.go... | Getting Started...

Create account Log in

eclipse

Search eclipse.org

GETTING STARTED MEMBERS PROJECTS MORE

DOWNLOAD

HOME / GETTING STARTED

1 2 3 4

Getting Started

You can download the standard version of Eclipse that contains the basic bits for any Java developer to start coding in Java.

DOWNLOAD

Eclipse also has pre-defined packages

Extend Eclipse

Eclipse Marketplace is a great source of plug-ins and product that you can add to Eclipse. Browse the online catalog or use the **Eclipse Marketplace** Client from within Eclipse. Look under the Eclipse Help Menu.

Popular Plugins:

- **Subversive - SVN Team Provider**

Documentation

These are a few of the popular getting-started documents for someone new to Eclipse:

- Getting Started with the **Eclipse Workbench**
- Getting Started with **Java development**

Getting Help

- There are many online sources of help in the Eclipse community. First thing to do is **create an account** so you can use them.
- Our **forums** are great places to ask questions, especially the **newcomer forum**.

Eclipse

Java EE - jspx2/src/eg/java/net/web/jspx/ui/pages/Page.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access

Java EE | SVN Repository Exploring | CVS Repository Exploring | Team Synchronizing

Page.java

jspx2 ▸ src ▸ eg.java.net.web.jspx.ui.pages ▸ Page ▸

```
73 /**
74  * Main controller for All JSPX files. <br />
75  * used as default for pages without controller. <br />
76  * the content and master pages inherits from.
77  *
78  * @author amr.eladawy
79  *
80  */
81 public class Page
82 {
83     private static final Logger logger = LoggerFactory.getLogger(Page.class);
84
85     public static final String DispatchedPage = "DispatchedPage";
86     public static final String RedirectPage = "RedirectPage";
87
88     protected ServletContext context;
89     protected String charSet;
90
91     private String pageURL;
92
93     protected String filePath;
94
```

Problems | Call Hierarchy | Console | Servers | Progress | Search | History | Call Hierarchy | Synchronize | Search

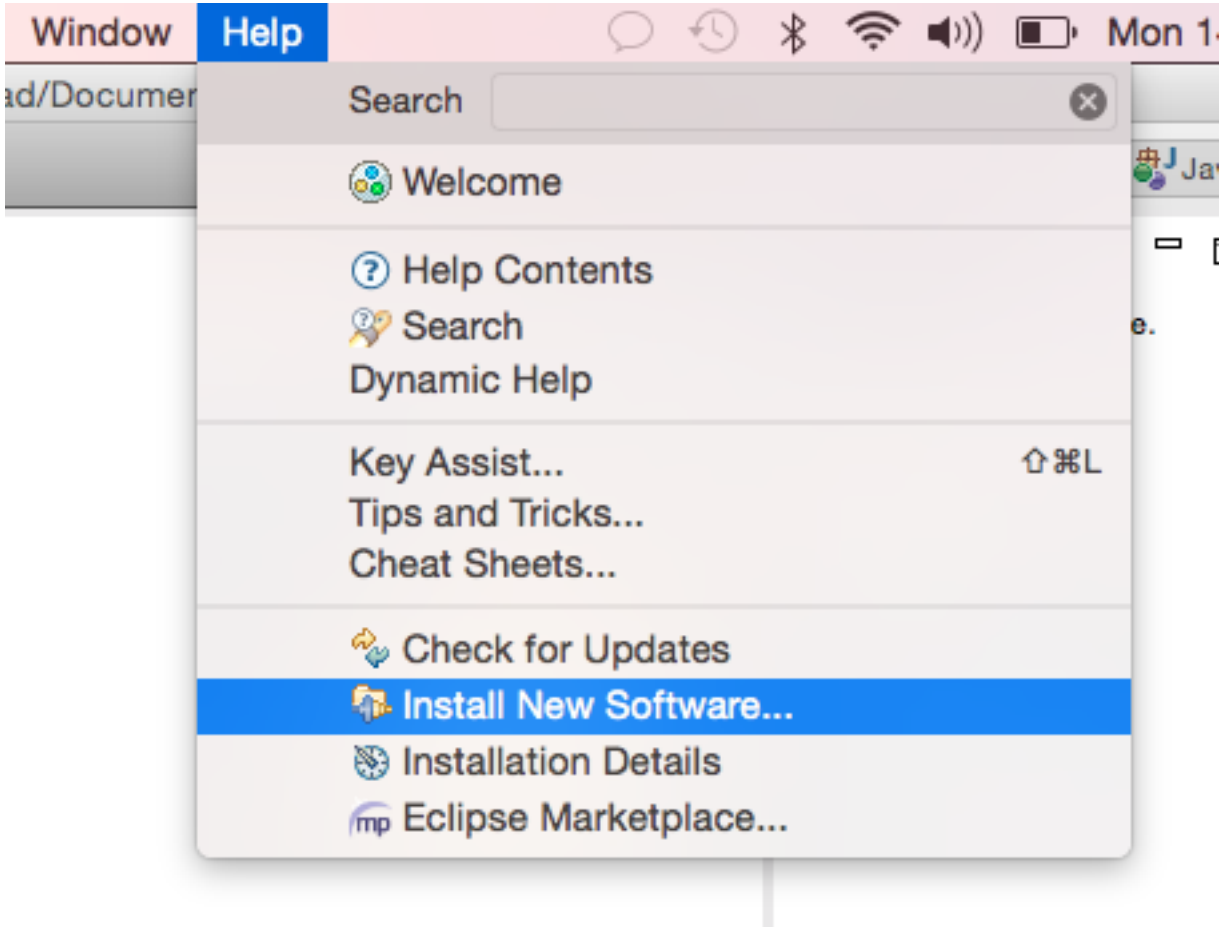
JBoss 7.1 Runtime Server
JBoss v5.0 at localhost

jspx2
- src
  - eg.java.net.web.jspx
    - engine
    - ui
      - controls
      - pages
        - ContentPage.java 1.1
        - MasterPage.java 1.1
        - Page.java 1.4
        - PortletPage.java 1.1
    - org.kxml
    - resources
- commons-fileupload-1.2.1.jar 1.1
- commons-io-1.3.2.jar 1.1
- poi-3.9-20121203.jar 1.1
- servlet-api.jar 1.1
- slf4j-api-1.7.2.jar 1.1
- commons-jexl-2.1.1.jar 1.1
- JRE System Library [jdk1.5.0_22]
- lib
- Proposed Schema
- ant.properties 1.2
- build.properties 1.4
- build.xml 1.1

Writable | Smart Insert | 81 : 18

Installing TestNG in Eclipse

- Open your Eclipse application.
- Go to Help | Install New Software.



http://testng.org/doc/download.html

Downloading TestNG

Current Release Version

The latest version of TestNG can be downloaded from Maven Central or here for ant users.

For the Eclipse plug-in, we suggest using the update site:

- Select *Help / Software updates / Find and Install.*
- *Search for new features to install.*
- *New remote site.*
- For Eclipse 3.4 and above, enter http://beust.com/eclipse.
- For Eclipse 3.3 and below, enter http://beust.com/eclipse1.
- Make sure the check box next to URL is checked and click *Next.*
- Eclipse will then guide you through the process.

Copy URL

You can also install older versions of the plug-ins here. Note that the URL's on this page are update sites as well, not direct download links.

TestNG is also hosted on GitHub, where you can download the source and build the distribution yourself:

```
$ git clone git://github.com/cbeust/testng.git
$ cd testng
$ cp ivy-2.1.0.jar ~/.ant/lib
$ ant
```

You will then find the jar file in the `target` directory

Install

Available Software

Check the items that you wish to install.

Paste URL here in Eclipse

Work with: http://beust.com/eclipse Add...

Find more software by working with the "Available Software Sites" preferences.

Name	Version
TestNG	

Select All Deselect All

Details

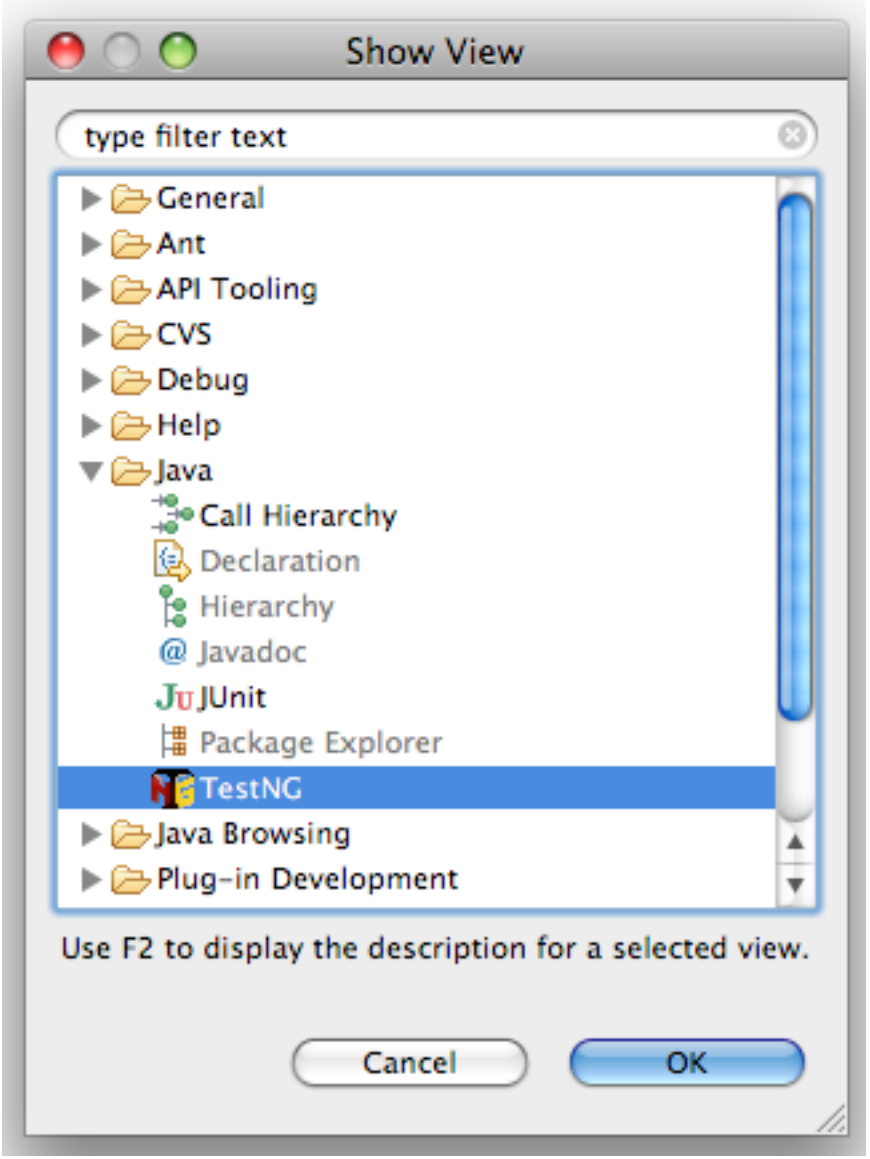
- [x] Show only the latest versions of available software
- [x] Group items by category
- [ ] Show only software applicable to target environment
- [x] Contact all update sites during install to find required software
- [x] Hide items that are already installed

What is already installed?

< Back Next > Cancel Finish

Note*: Copied from TestNG Site

- restart Eclipse and select the menu Window / Show View / Other... and you should see the TestNG view listed in the Java category.



Tests in TestNG

- A test in TestNG usually involves:

1. Setup
   - Create and initialise fixture object(s) necessary for the test.
2. Act
   - Carry out the operation that is being tested.
3. Assert
   - Check that the results of the action are as expected.
4. Teardown
   - Reset or destroy any state that was changed by the test to avoid this test having side-effects on any following tests.

Test Code

- The following imports are typical for test code in TestNG:

```
1.  import org.testng.annotations.Test;
2.
3.  import static org.testng.Assert.assertEquals;
4.  import static org.testng.Assert.assertTrue;
```

- Tests are written as part of test classes.
- You can use Eclipse to create a new TestNG class for you and it will then take care of generating the appropriate imports.

Fixture

- A fixture is a common base state that all associated tests should start in.
- For unit tests
  - creating an initial set of SUT objects to be used in the tests.
- For integration tests
  - clearing out a database and creating an initial set of records in the database.
- For End-to-End tests
  - populating a database, creating a set of users with different authorisation levels and creating a number of data files for use by the system.

Annotations

- TestNG uses annotations to mark classes and methods as tests
- To mark a class as a test class, put the annotation @Test before the class.

```
1.    @Test
2.    public class myTest
3.    {
4.          ...
5.    }
```

- If you do not mark the class with the annotation @Test, then you can still mark individual methods in the class as test methods by adding that annotation to each of the individual methods.

Annotations

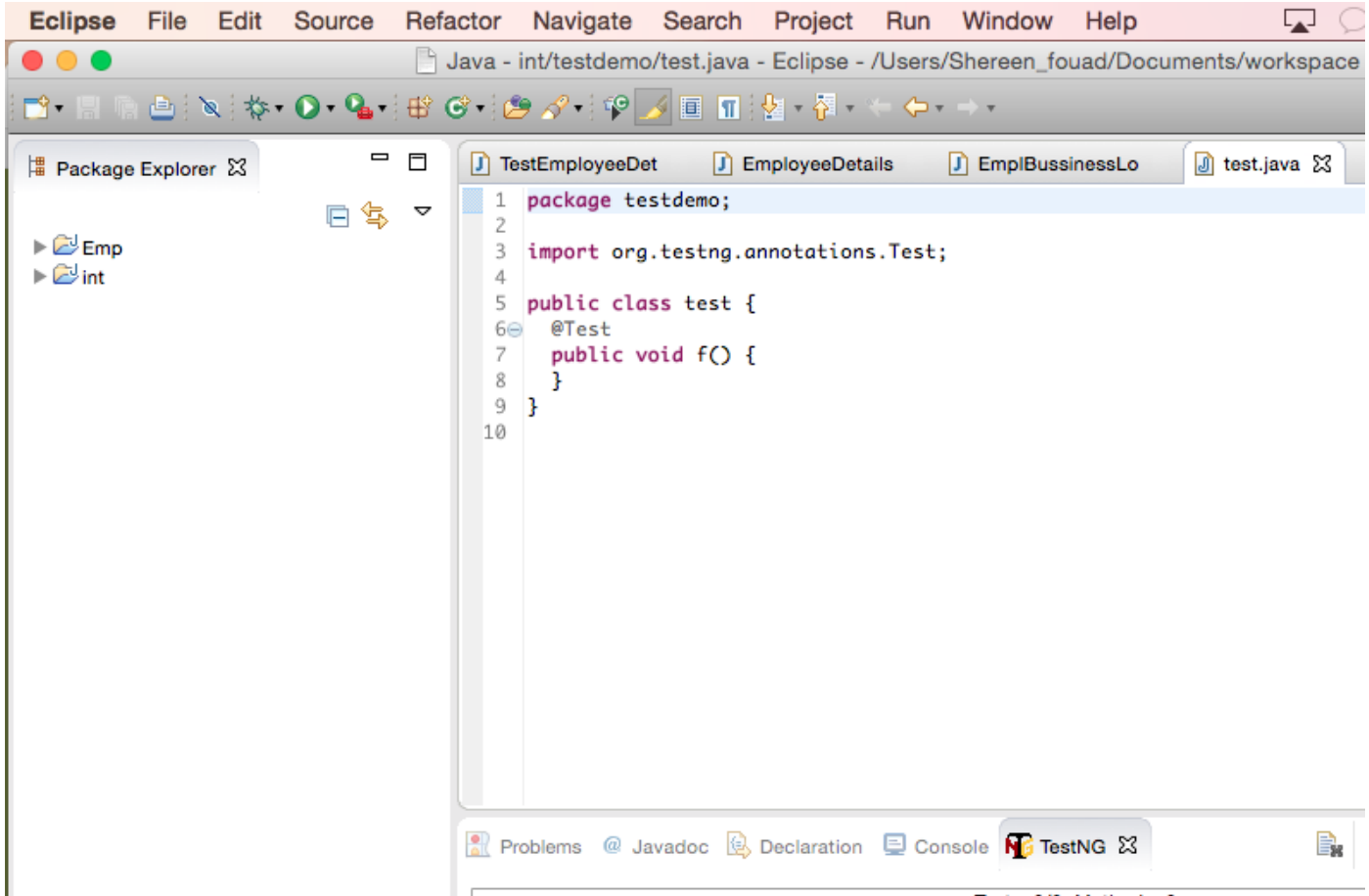
@Test	Marks a class or a method as part of the test.

Annotations allows you to perform some Java logic before and after the test

@BeforeClass	The annotated method will be run only once before the first test method in the current class is invoked.
@AfterClass	The annotated method will be run only once after all the test methods in the current class have been run.
@BeforeTest	The annotated method will be run before any test method belonging to the classes inside the <test> tag is run.
@AfterTest	The annotated method will be run after all the test methods belonging to the classes inside the <test> tag have run.

Test methods

- Test methods must have return type void.
- Each test method is executed in isolation



Assertions

- The key mechanism for specifying the expected behaviour in a test is an assertion.
- A method that evaluates some condition and throws an AssertionErrorException if the condition is not satisfied.
- A test asserts something is true:
  – assertEquals(”Bham”,obj.getName())
  – assertNull(aRef)
- TestNG provides a range of assertion methods, each with an optional String argument to print if the assertion fails.

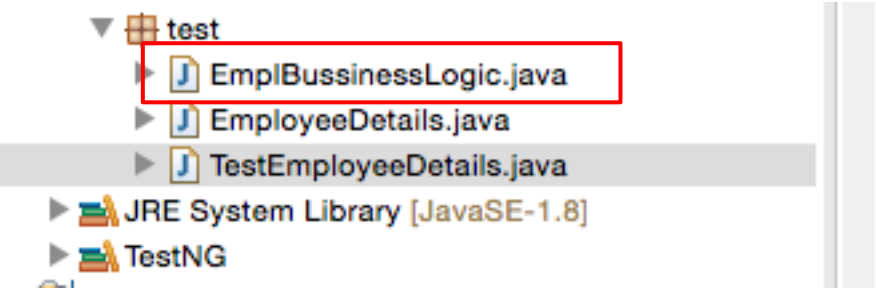
Example

EmployeeDetails *class is used to:*

- *get/set the value of employee's name.*
- *get/set the value of employee's monthly salary.*
- *get/set the value of employee's age.*

```java
package test;
public class EmployeeDetails {
   private String name;
   private double monthlySalary;
   private int age;
   /**
   * @return the name
   */
   public String getName() {
      return name;
   }
   /**
   * @param name the name to set
   */
   public void setName(String name) {
      this.name = name;
   }
   /**
   * @return the monthlySalary
   */
   public double getMonthlySalary() {
      return monthlySalary;
   }
   /**
   * @param monthlySalary the monthlySalary to set
   */
   public void setMonthlySalary(double monthlySalary) {
      this.monthlySalary = monthlySalary;
   }
   /**
   * @return the age
   */
   public int getAge() {
      return age;
   }
   /**
   * @param age the age to set
   */
   public void setAge(int age) {
   this.age = age;
   }
}
```

http://www.tutorialspoint.com/testng/testng_quick_guide.htm



EmpBusinessLogic *class is used for calculating.*

- *the yearly salary of employee.*
- *the appraisal amount of employee.*

```java
package test;


public class EmplBussinessLogic {
    // Calculate the yearly salary of employee
    public double calculateYearlySalary(EmployeeDetails employeeDetails){
        double yearlySalary=0;
        yearlySalary = employeeDetails.getMonthlySalary() * 12;
        return yearlySalary;
    }

    // Calculate the appraisal amount of employee
    public double calculateAppraisal(EmployeeDetails employeeDetails){
        double appraisal=0;
        if(employeeDetails.getMonthlySalary() < 10000){
            appraisal = 500;
        }else{
            appraisal = 1000;
        }
        return appraisal;
    }
}
```

http://www.tutorialspoint.com/testng/testng_quick_guide.htm

Adding the TestNG libraries in Eclipse

- Select the project from the Package Explorer panel,
- Then select menu item Project|Properties.
- In the dialog box that opens, select Java Build Path in the left panel and then the Libraries tab.
- You should have the JRE System Library there by default.
- Click the Add Library... button and select the TestNG library, click Next> and then Finish to add the library.

Package Explorer

Interest.java | EmployeeDetails

```
1 package test;
  import org.testng.Assert;

  public class TestEmployeeDetails
      EmplBussinessLogic empBusin
      EmployeeDetails employee =

      @Test
      public void testCalculateApp
          employee.setName("Rajeev
          employee.setAge(25);
          employee.setMonthlySalar
          double appraisal = empBu
              .calculateAppraisal(
          Assert.assertEquals(500,
      }

      // test to check yearly sala
```

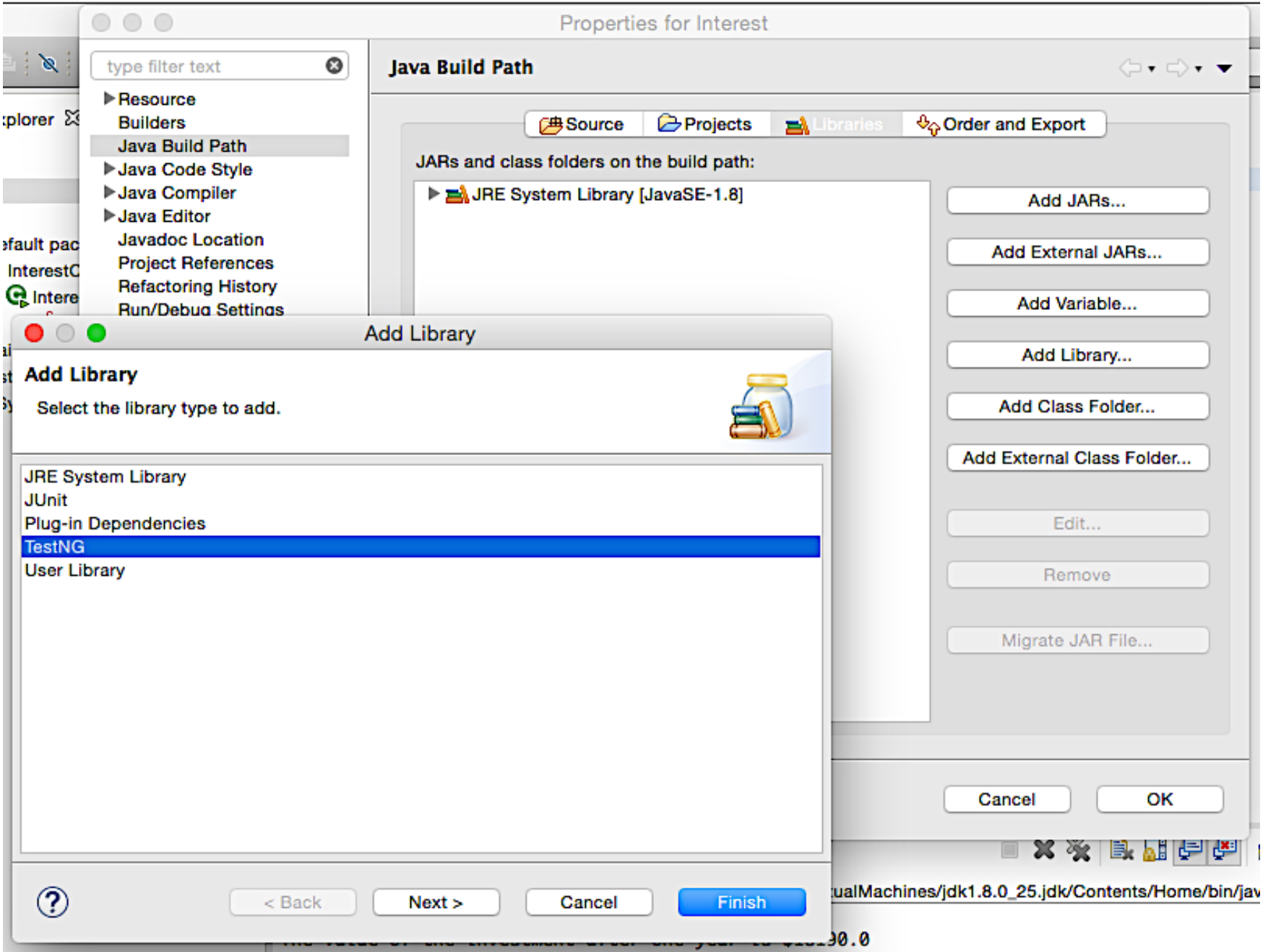
oblems | @ Javadoc | Declaration

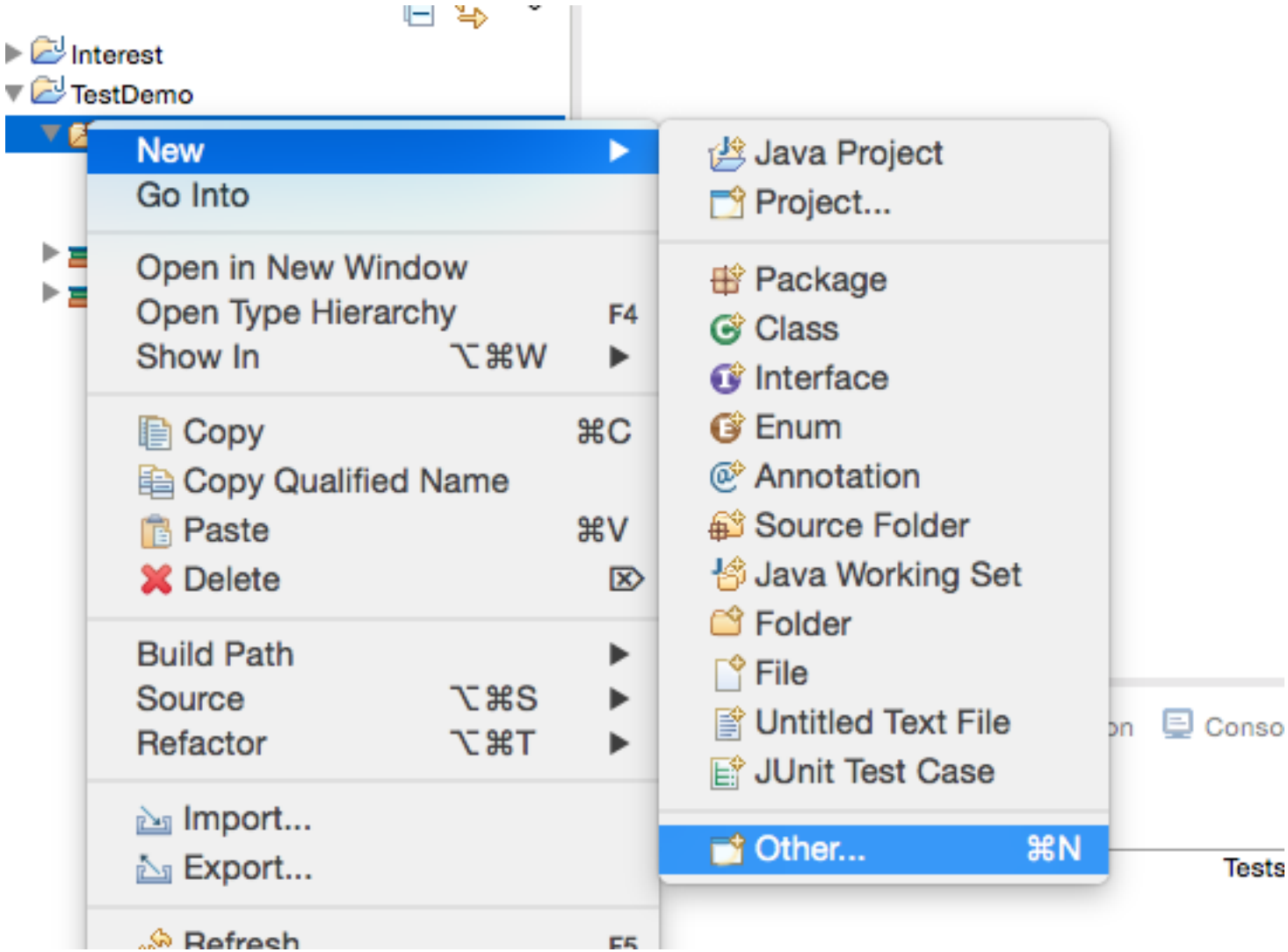
ch:

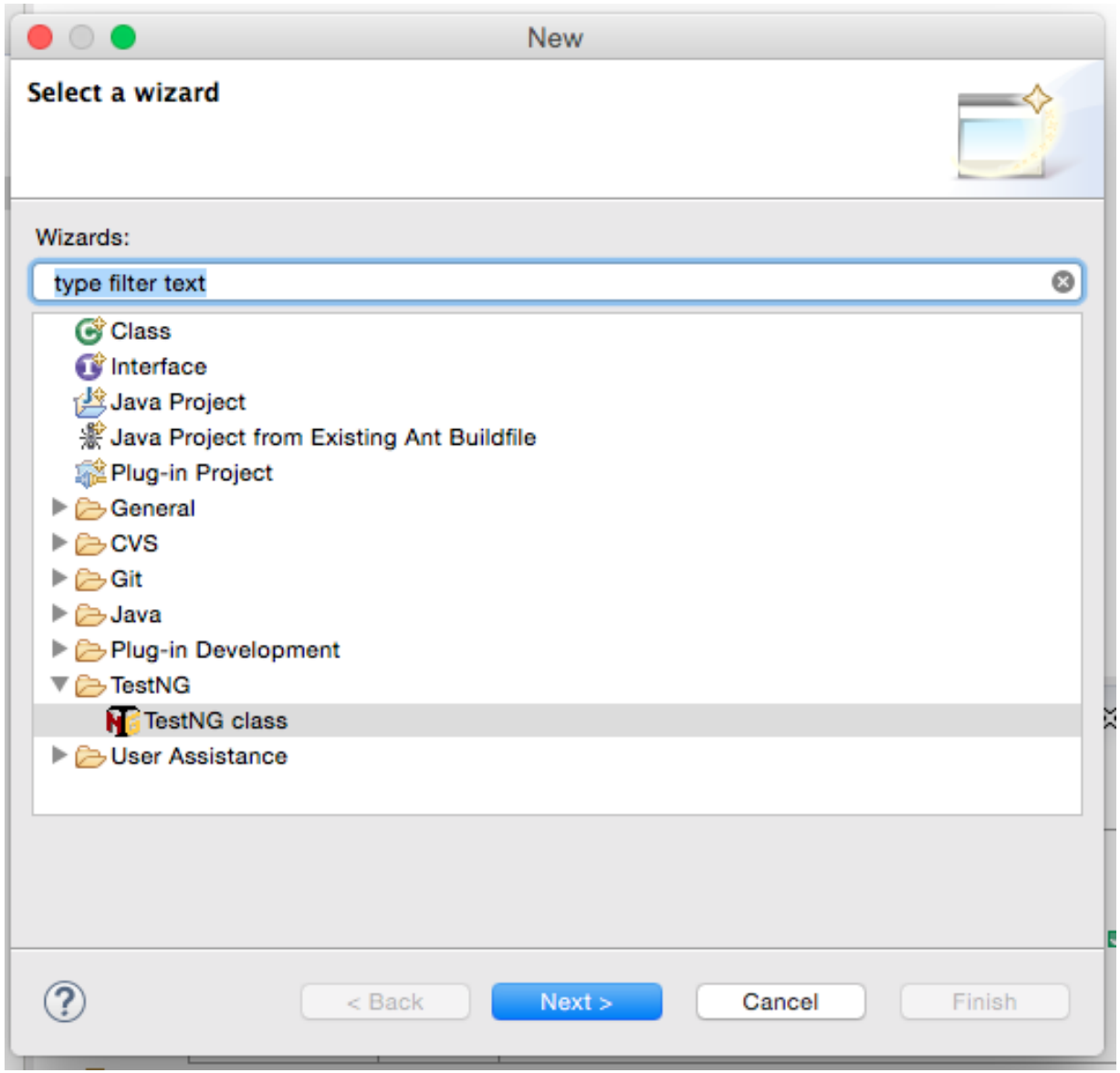
ll Tests | Failed Tests | Summary

Emp

- New ▶
- Go Into
- Open in New Window
- Open Type Hierarchy F4
- Show In ⌥⌘W ▶
- Copy ⌘C
- Copy Qualified Name
- Paste ⌘V
- Delete ⌦
- Build Path ▶
- Source ⌥⌘S ▶
- Refactor ⌥⌘T ▶
- Import...
- Export...
- Refresh F5
- Close Project
- Close Unrelated Projects
- Assign Working Sets...
- Debug As ▶
- Run As ▶
- Team ▶
- Compare With ▶
- Restore from Local History...
- Configure ▶
- TestNG ▶
- Properties ⌘I


Properties for Interest
type filter text
Resource
Builders
Java Build Path
Java Code Style
Java Compiler
Java Editor
Javadoc Location
Project References
Refactoring History
Run/Debug Settings
Java Build Path
Source
Projects
Libraries
Order and Export
JARs and class folders on the build path:
JRE System Library [JavaSE-1.8]
Add JARs...
Add External JARs...
Add Variable...
Add Library...
Add Class Folder...
Add External Class Folder...
Edit...
Remove
Migrate JAR File...
Cancel
OK
Add Library
Add Library
Select the library type to add.
JRE System Library
JUnit
Plug-in Dependencies
TestNG
User Library
< Back
Next >
Cancel
Finish


Interest
TestDemo
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In ⌥⌘W
Copy ⌘C
Copy Qualified Name
Paste ⌘V
Delete
Build Path
Source ⌥⌘S
Refactor ⌥⌘T
Import...
Export...
Refresh F5
Java Project
Project...
Package
Class
Interface
Enum
Annotation
Source Folder
Java Working Set
Folder
File
Untitled Text File
JUnit Test Case
Other... ⌘N
Conso
Tests


New
Select a wizard
Wizards:
type filter text
Class
Interface
Java Project
Java Project from Existing Ant Buildfile
Plug-in Project
General
CVS
Git
Java
Plug-in Development
TestNG
TestNG class
User Assistance
< Back
Next >
Cancel
Finish

New TestNG class

Specify additional information about the test class.

Source folder: /TestDemo/src/test Browse...

Package name: test Browse...

Class name: NewTest

Annotations

- [ ] @BeforeMethod
- [ ] @AfterMethod
- [ ] @DataProvider
- [ ] @BeforeClass
- [ ] @AfterClass
- [ ] @BeforeTest
- [ ] @AfterTest
- [ ] @BeforeSuite
- [ ] @AfterSuite

XML suite file:

< Back | Next > | Cancel | Finish

Eclipse File Edit Source Refactor Navigate Search Project Run Window Help

Java - int/testdemo/test.java - Eclipse - /Users/Shereen_fouad/Documents/workspace

Package Explorer

- Emp
- int

TestEmployeeDet | EmployeeDetails | EmplBussinessLo | test.java

```
1  package testdemo;
2
3  import org.testng.annotations.Test;
4
5  public class test {
6    @Test
7    public void f() {
8    }
9  }
10
```

Problems | Javadoc | Declaration | Console | TestNG

Tests: 0/0 Methods: 0

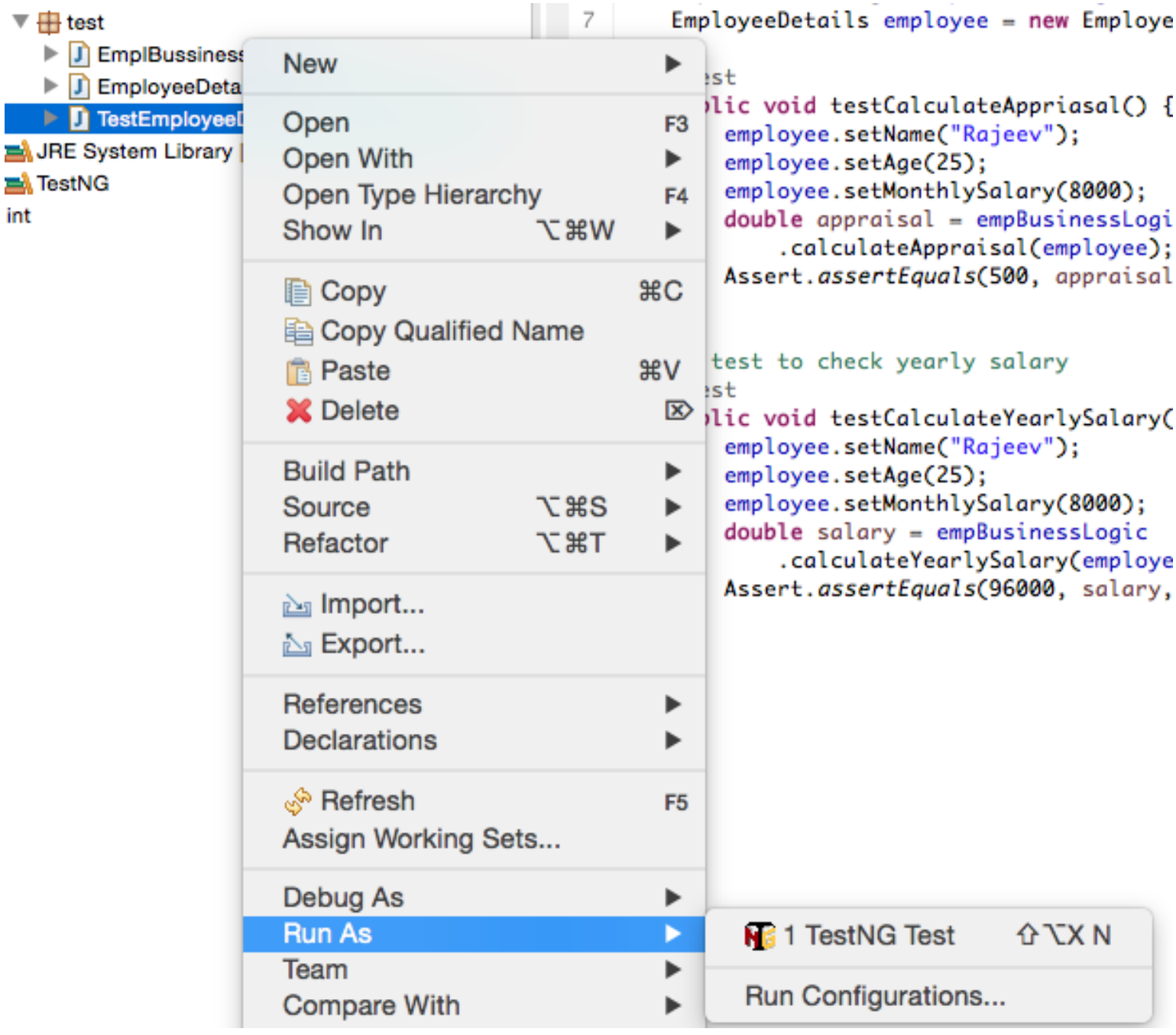
EmployeeDetails | EmplBussinessLo | TestEmployeeDet | *EmployeeDetail | EmplBus

```java
package test;
import org.testng.Assert;
import org.testng.annotations.Test;

public class TestEmployeeDetails {
    EmplBussinessLogic empBusinessLogic = new EmplBussinessLogic();
    EmployeeDetails employee = new EmployeeDetails();

    @Test
    public void testCalculateAppriasal() {
        employee.setName("Rajeev");
        employee.setAge(25);
        employee.setMonthlySalary(8000);
        double appraisal = empBusinessLogic
            .calculateAppraisal(employee);
        Assert.assertEquals(500, appraisal, 0.0, "500");
    }

    // test to check yearly salary
    @Test
    public void testCalculateYearlySalary() {
        employee.setName("Rajeev");
        employee.setAge(25);
        employee.setMonthlySalary(8000);
        double salary = empBusinessLogic
            .calculateYearlySalary(employee);
        Assert.assertEquals(96000, salary, 0.0, "8000");
    }
}
```

tests the yearly salary of the employee

tests the appraisal amount of the employee

http://www.tutorialspoint.com/testng/testng_quick_guide.htm


test
EmplBussiness
EmployeeDeta
TestEmployeeD
JRE System Library
TestNG
int
New
Open F3
Open With
Open Type Hierarchy F4
Show In ⌥⌘W
Copy ⌘C
Copy Qualified Name
Paste ⌘V
Delete
Build Path
Source ⌥⌘S
Refactor ⌥⌘T
Import...
Export...
References
Declarations
Refresh F5
Assign Working Sets...
Debug As
Run As
Team
Compare With
1 TestNG Test ⇧⌥X N
Run Configurations...
7 EmployeeDetails employee = new Employe
est
lic void testCalculateAppriasal() {
employee.setName("Rajeev");
employee.setAge(25);
employee.setMonthlySalary(8000);
double appraisal = empBusinessLogi
.calculateAppraisal(employee);
Assert.assertEquals(500, appraisal
test to check yearly salary
est
lic void testCalculateYearlySalary(
employee.setName("Rajeev");
employee.setAge(25);
employee.setMonthlySalary(8000);
double salary = empBusinessLogic
.calculateYearlySalary(employe
Assert.assertEquals(96000, salary,

Verify the Result

Problems | Javadoc | Declaration | Console | Results of running class TestEmployeeDetails

```
<terminated> TestEmployeeDetails [TestNG] /Library/Java/JavaVirtualMachines/jdk1.8.0_25.jdk/Contents/Home/bin/java (5 Jan 2015 21:04:31)
[TestNG] Running:
  /private/var/folders/f8/nt30n7qx5sd_cwl9krnj5hnr0000gn/T/testng-eclipse--560227673/testng-customsuite.xml

PASSED: testCalculateAppriasal
PASSED: testCalculateYearlySalary

===============================================
    Default test
    Tests run: 2, Failures: 0, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 2, Failures: 0, Skips: 0
===============================================

[TestNG] Time taken by org.testng.reporters.jq.Main@1b701da1: 121 ms
[TestNG] Time taken by org.testng.reporters.XMLReporter@7dc5e7b4: 56 ms
[TestNG] Time taken by org.testng.reporters.EmailableReporter2@515f550a: 8 ms
[TestNG] Time taken by org.testng.reporters.SuiteHTMLReporter@3f8f9dd6: 14 ms
[TestNG] Time taken by [FailedReporter passed=0 failed=0 skipped=0]: 1 ms
[TestNG] Time taken by org.testng.reporters.JUnitReportReporter@379619aa: 3 ms
```

Verify the Result

Problems | Javadoc | Declaration | Console | Results of running class TestEmployeeDetails

Tests: 1/1 Methods: 2 (500 ms)

Search:

All Tests | Failed Tests | Summary

- Default suite (2/0/0/0) (0.029 s)
  - Default test (0.029 s)
    - test.TestEmployeeDetails
      - testCalculateAppriasal (0.028 s)
      - testCalculateYearlySalary (0.001 s)

Failure Exception

Interest.java | EmployeeDetails | EmplBussinessLo | TestEmployeeDet | »3

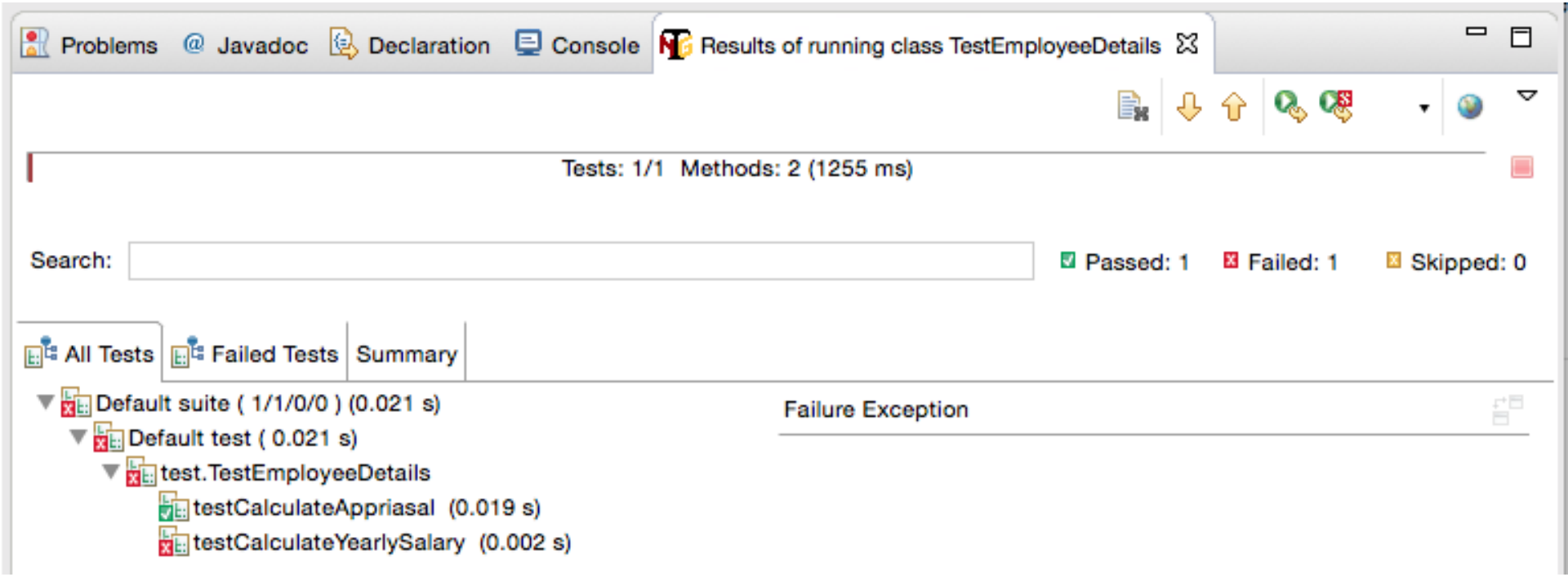
```java
package test;
import org.testng.Assert;

public class TestEmployeeDetails {
    EmplBussinessLogic empBusinessLogic = new EmplBussinessLogic();
    EmployeeDetails employee = new EmployeeDetails();

    @Test
    public void testCalculateAppriasal() {
        employee.setName("Rajeev");
        employee.setAge(25);
        employee.setMonthlySalary(8000);
        double appraisal = empBusinessLogic
            .calculateAppraisal(employee);
        Assert.assertEquals(500, appraisal, 0.0, "500");
    }

    // test to check yearly salary
    @Test
    public void testCalculateYearlySalary() {
        employee.setName("Rajeev");
        employee.setAge(25);
        employee.setMonthlySalary(8000);
        double salary = empBusinessLogic
            .calculateYearlySalary(employee);
        Assert.assertEquals(9000, salary, 0.0, "8000");
    }
}
```

If a test fails...

Problems | Javadoc | Declaration | Console | Results of running class TestEmployeeDetails

```
<terminated> TestEmployeeDetails [TestNG] /Library/Java/JavaVirtualMachines/jdk1.8.0_25.jdk/Contents/Home/bin/java (10 Jan 2015 04:51:03)
[TestNG] Running:
  /private/var/folders/f8/nt30n7qx5sd_cwl9krnj5hnr0000gn/T/testng-eclipse--1740108362/testng-customsuite.xml

PASSED: testCalculateAppriasal
FAILED: testCalculateYearlySalary
java.lang.AssertionError: 8000 expected [96000.0] but found [9000.0]
        at org.testng.Assert.fail(Assert.java:94)
        at org.testng.Assert.failNotEquals(Assert.java:494)
        at org.testng.Assert.assertEquals(Assert.java:207)
        at test.TestEmployeeDetails.testCalculateYearlySalary(TestEmployeeDetails.java:27)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:483)
        at org.testng.internal.MethodInvocationHelper.invokeMethod(MethodInvocationHelper.java:84)
        at org.testng.internal.Invoker.invokeMethod(Invoker.java:714)
        at org.testng.internal.Invoker.invokeTestMethod(Invoker.java:901)
```

If a test fails...



Conclusion

- Testing is a key activity in software development process
- Testing can reveal the presence of errors NOT their absence
- Testing may be done in different levels:
  - Unit Testing
  - Integration Testing
  - End to End Testing
- TestNg is an open source automated testing framework

Next time

- Passing Parameters with *@Dataproviders*
- *Test Driven Development*