Công Cụ & Phương Pháp Thiết Kế - Quản Lý (Phần Mềm)

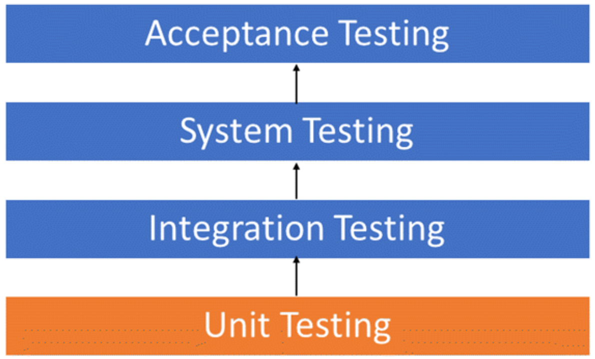
TRAN KIM SANH
Instructor of DTU

Email: trankimsanh@dtu.edu.vn
Tel: 0987 409 464

Using Your Configuration Management Processes Part 2

Contents

- Overview of Java Development Tools Used in this Course
  - eclipse
  - svn
  - Ant
  - CruiseControl (not used in this course)
  - JUnit
    - ✓ What is testing?
    - ✓ What is Junit?
    - ✓ How to write testcase?
    - ✓ What are Junit methods?
    - ✓ How to use Junit in Eclipse?



Question

- **ANT is used to …. ?**

  A. Build Project

  B. Manage the revision of the project

  C. Share document and code

  D. Review code

Question

- **In order to use Eclipse, you must .... ?**

  A. install the current version of Java Development Kit on your computer

  B. install the current version of Tomcat on your computer

  C. install the current version of SVN on your computer

  D. install the current version of ANT on your computer

Software Testing

- Software testing is the process of evaluation a software item to detect differences between given input and expected output

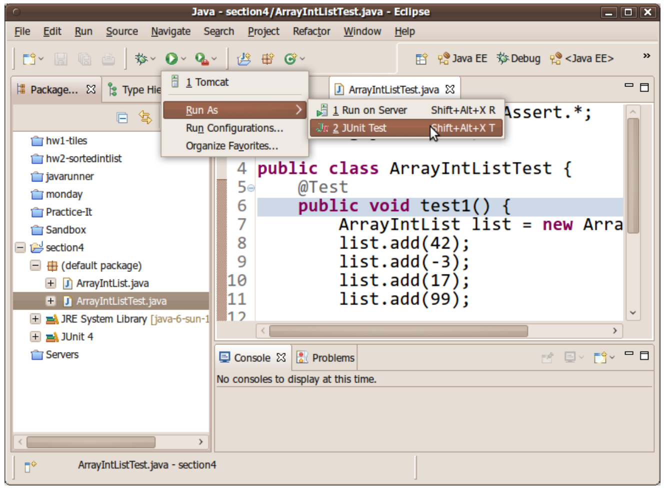
- Types
  - Unit testing
  - Integration testing
  - Regression Testing
  - System testing
  - Acceptance testing

Testing with JUnit

- Junit is a **unit test environment** for Java programs developed by *Erich Gamma* and *Kent Beck*.
  - ✓ Writing test cases
  - ✓ Executing test cases
  - ✓ Pass/fail? (expected result = obtained result?)

Testing with JUnit

- Consists in a **framework** providing all the tools for testing.
  - framework: set of classes and conventions to use them.
- It is **integrated into Eclipse** through a graphical plug-in.

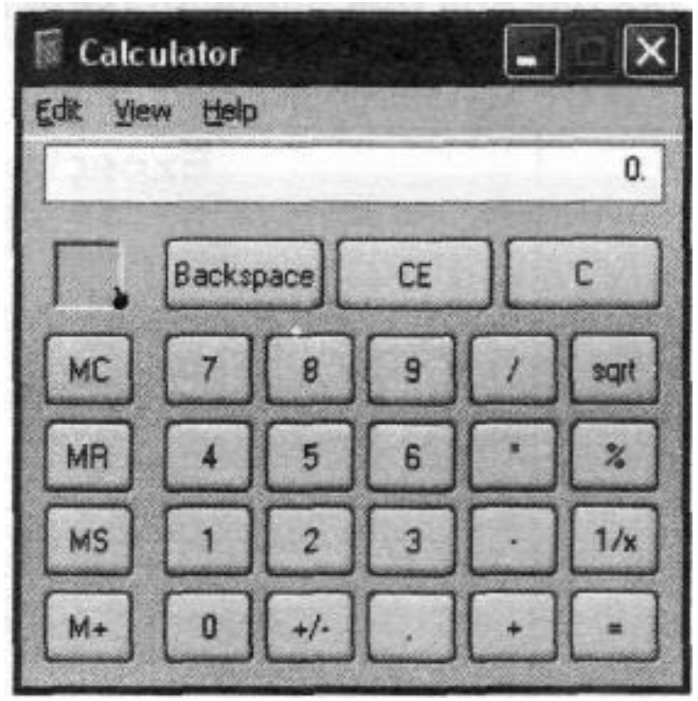


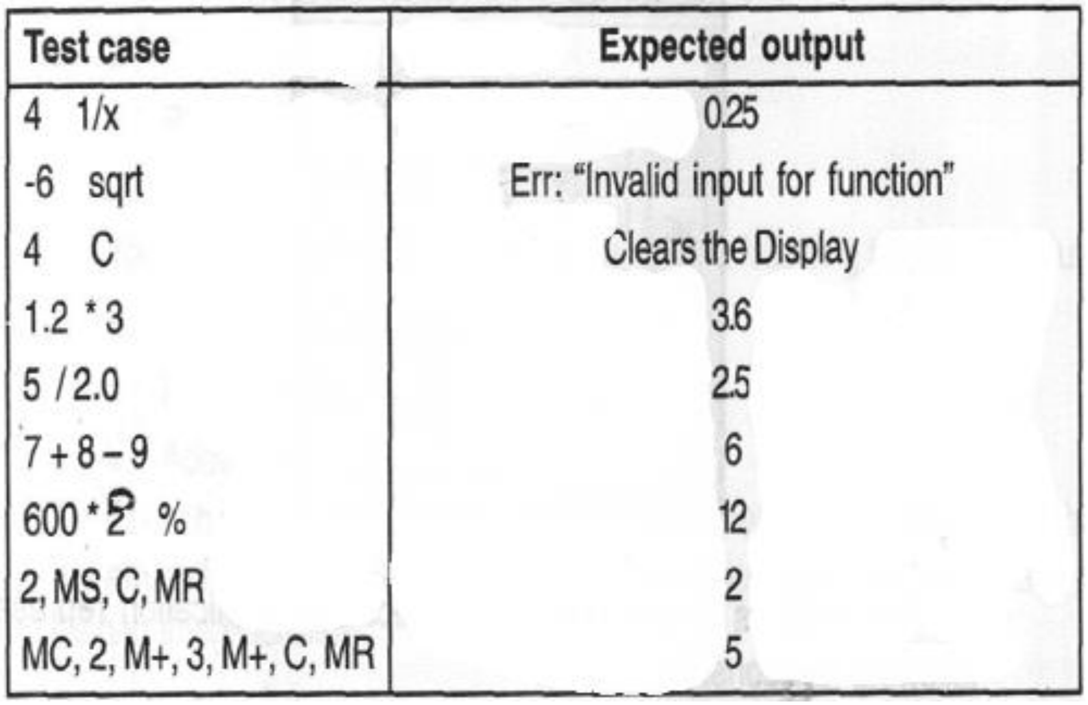
```
Assert.*;

public class ArrayIntListTest {
    @Test
    public void test1() {
        ArrayIntList list = new Arra
        list.add(42);
        list.add(-3);
        list.add(17);
        list.add(99);
```

No consoles to display at this time.

Design Test Cases

- A test case in software engineering is a set of conditions or variables under which a tester will determine whether an application or software system is working correctly or not.





Test case	Expected output
4 1/x	0.25
-6 sqrt	Err: "Invalid input for function"
4 C	Clears the Display
1.2 * 3	3.6
5 / 2.0	2.5
7 + 8 − 9	6
600 * 2 %	12
2, MS, C, MR	2
MC, 2, M+, 3, M+, C, MR	5

Assert methods

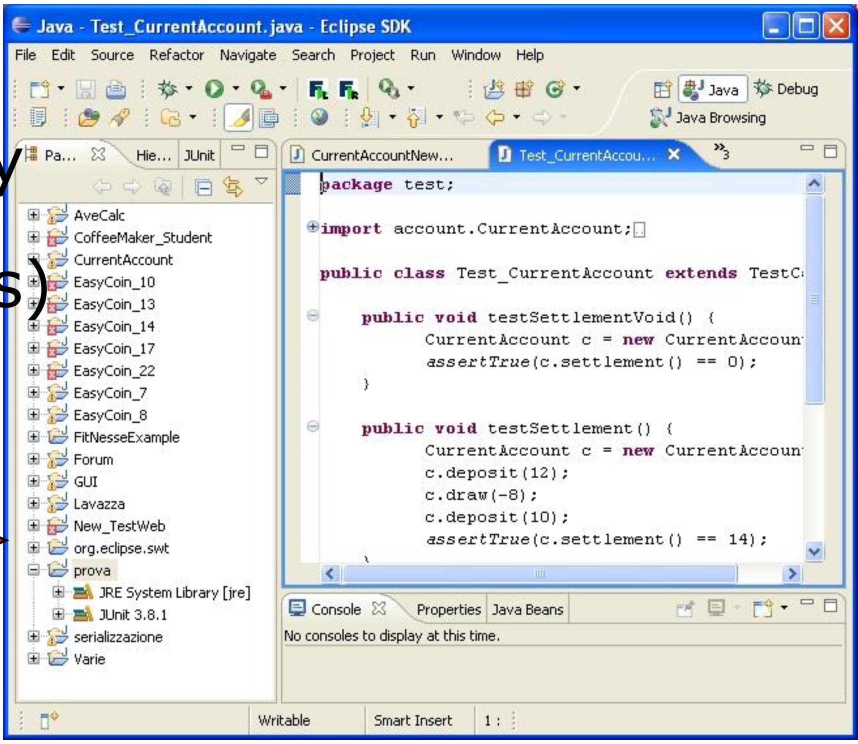
- assertEquals(Object *expected*, Object *actual*)
  - Test that float or double values match. The tolerance is the number of decimals which must be the same.
- assertNull([message], object)
  - Checks that the object is null.

Assert methods

- assertTrue(Boolean *test*)
  - Will always be true / false. Can be used to predefine a test result, if the test is not yet implemented.
- fail([String message])
  - Let the method fail. Might be used to check that a certain part of the code is not reached. Or to have failing test before the test code is implemented.

Junit in Eclipse - Setup

- In Eclipse
  - Create a new project
  - Open project’s property window (File -> Properties)
  - Select: Java build path
  - Select: libraries
  - Add Library
  - Select Junit
    - ✓ Select the type 3.x or 4.x...



```
package test;

import account.CurrentAccount;

public class Test_CurrentAccount extends TestC

    public void testSettlementVoid() {
        CurrentAccount c = new CurrentAccoun
        assertTrue(c.settlement() == 0);
    }

    public void testSettlement() {
        CurrentAccount c = new CurrentAccoun
        c.deposit(12);
        c.draw(-8);
        c.deposit(10);
        assertTrue(c.settlement() == 14);
```

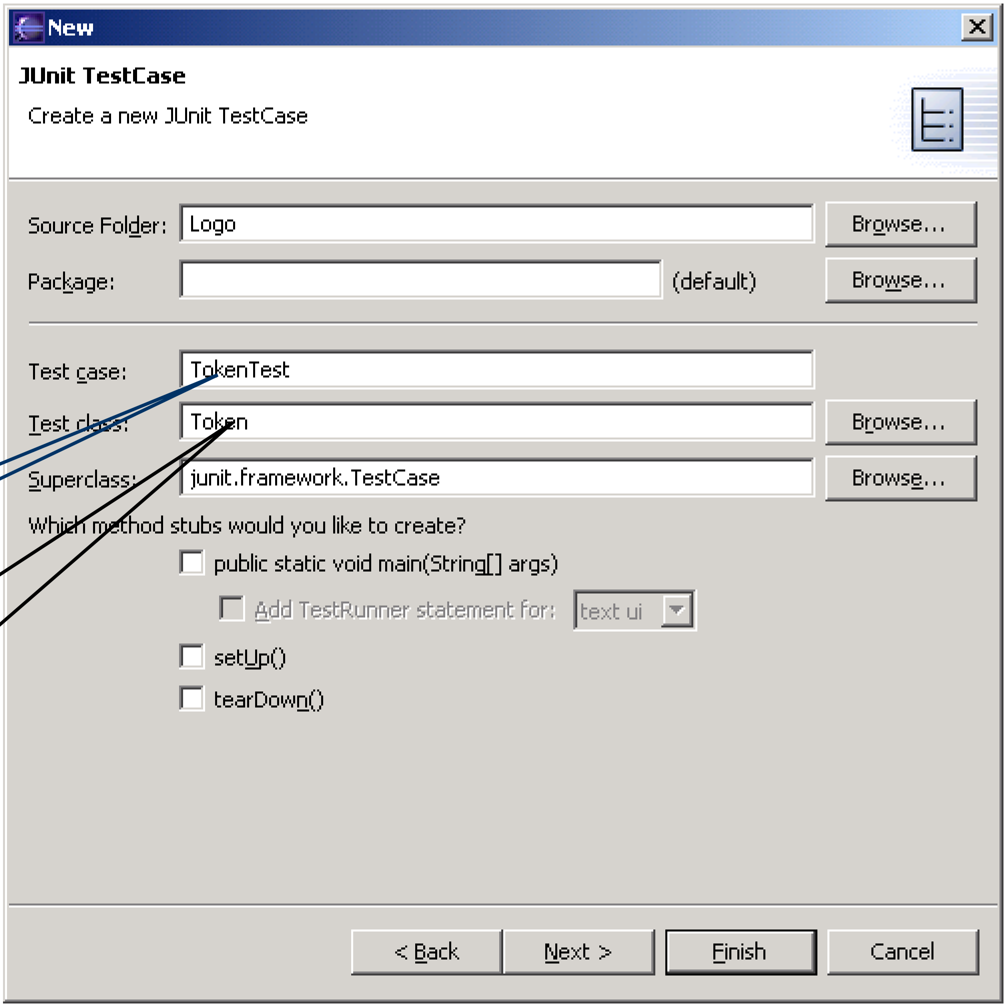
No consoles to display at this time.

JUnit in Eclipse

- To create a test class, select File→ New→ Other... → Java, JUnit, TestCase and enter the name of the *class* you will test

Fill this in

This will be filled in *automatically*



New

JUnit TestCase

Create a new JUnit TestCase

Source Folder: Logo

Package: (default)

Test case: TokenTest

Test class: Token

Superclass: junit.framework.TestCase

Which method stubs would you like to create?

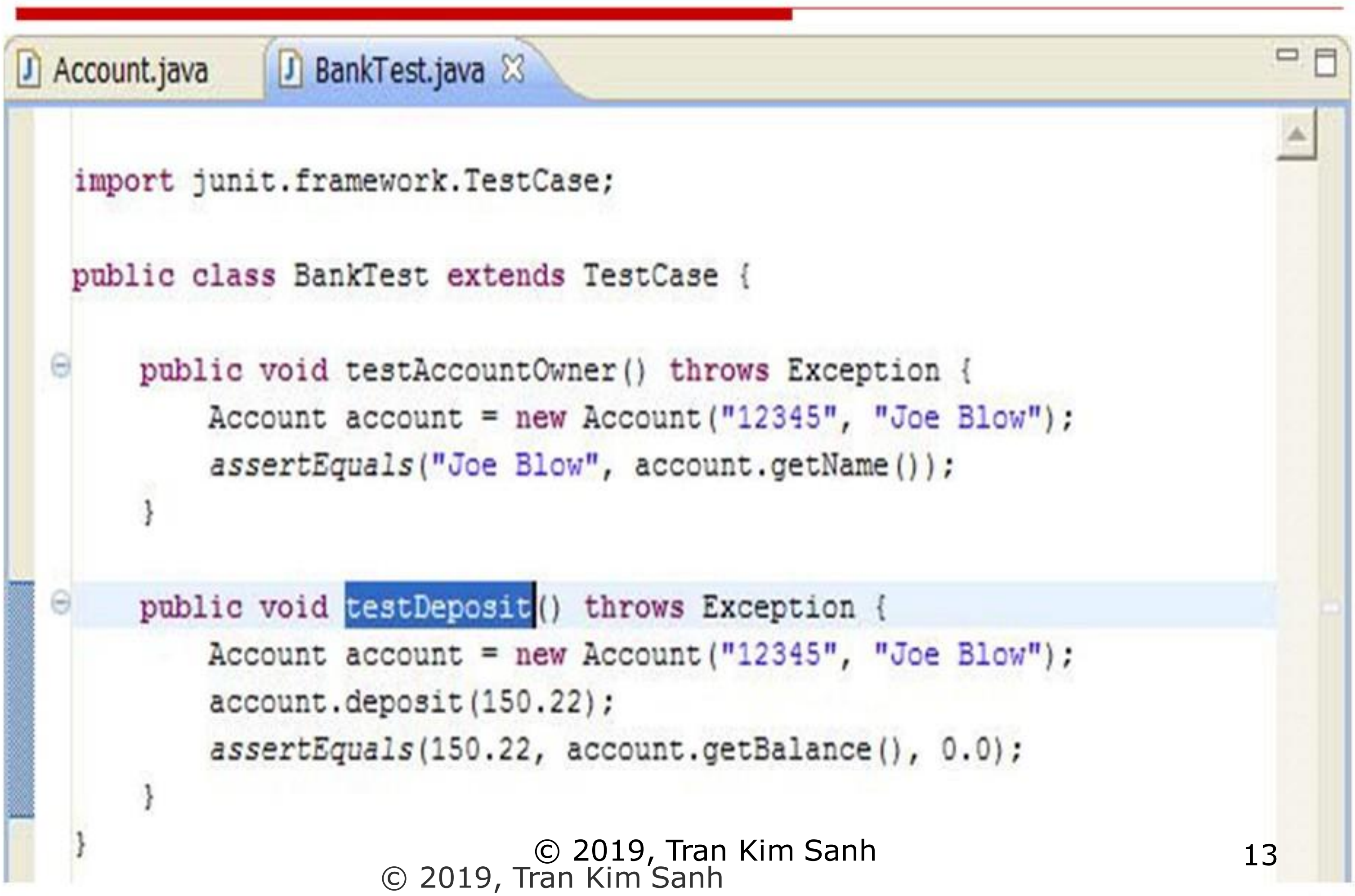
☐ public static void main(String[] args)

☐ Add TestRunner statement for: text ui

☐ setUp()

☐ tearDown()

Eclipse - Test Case

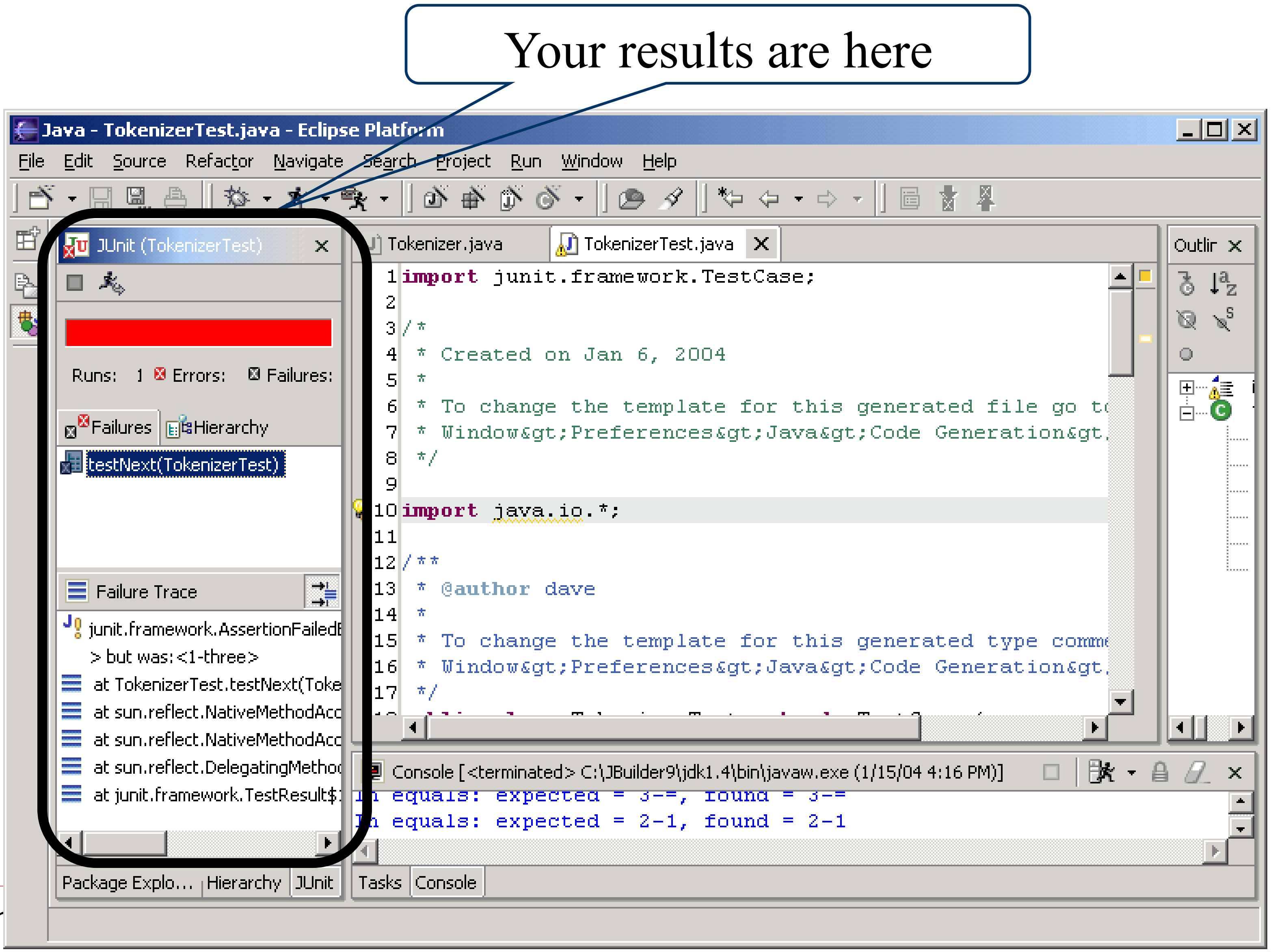


Account.java BankTest.java

```
import junit.framework.TestCase;

public class BankTest extends TestCase {

    public void testAccountOwner() throws Exception {
        Account account = new Account("12345", "Joe Blow");
        assertEquals("Joe Blow", account.getName());
    }

    public void testDeposit() throws Exception {
        Account account = new Account("12345", "Joe Blow");
        account.deposit(150.22);
        assertEquals(150.22, account.getBalance(), 0.0);
    }
}
```

© 2019, Tran Kim Sanh 4

Results



Java - TokenizerTest.java - Eclipse Platform

JUnit (TokenizerTest)

Runs: 1 Errors: Failures:

testNext(TokenizerTest)

Failure Trace

```
junit.framework.AssertionFailedE
> but was:<1-three>
at TokenizerTest.testNext(Toke
at sun.reflect.NativeMethodAcc
at sun.reflect.NativeMethodAcc
at sun.reflect.DelegatingMetho
at junit.framework.TestResult$
```

```java
import junit.framework.TestCase;

/*
 * Created on Jan 6, 2004
 *
 * To change the template for this generated file go to
 * Window&gt;Preferences&gt;Java&gt;Code Generation&gt;
 */

import java.io.*;

/**
 * @author dave
 *
 * To change the template for this generated type comme
 * Window&gt;Preferences&gt;Java&gt;Code Generation&gt;
 */
```

Console [<terminated> C:\JBuilder9\jdk1.4\bin\javaw.exe (1/15/04 4:16 PM)]

```
In equals: expected = 3-=, found = 3-=
In equals: expected = 2-1, found = 2-1
```

Group discussion?

- How to write unit test for the codes below - 4 students – 10 minutes

```
public class Sort {
 int number1;
 int number2;
 public void sortDesc()
 {
  if(number1< number2)
  {
   int temp = number1;
   number1 = number2;
   number2 = temp;
  }
 }
}
```

Can You Afford a Build Machine?

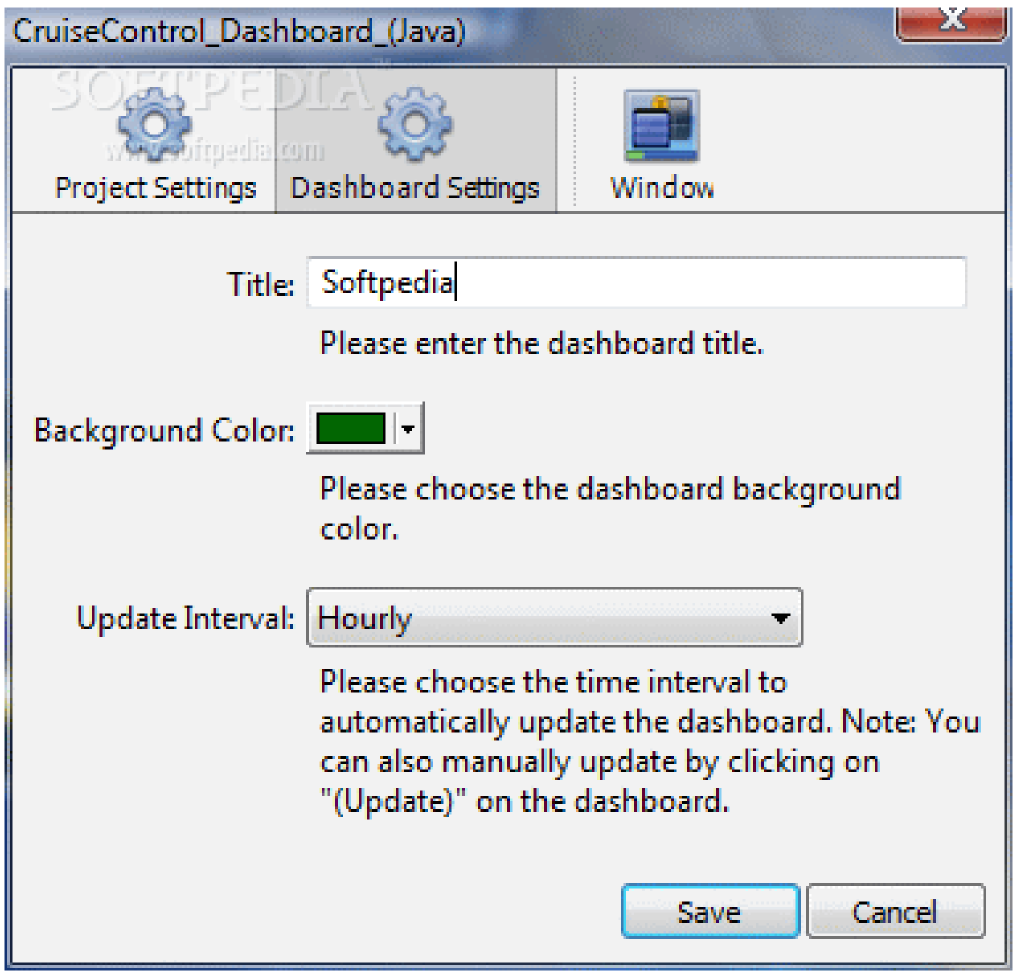
- Can your team afford a machine dedicated to automatically building and testing your software on a regular interval?
  - Assume that a ten-person development team costs your company $500/ hour
  - If that team spends two less hours debugging integration problems over the life of the project, you’ve paid for a build machine

Can You Afford a Build Machine?

- Every day your team is fighting integration and quality problems at the end of your project is another day your product is late to market
  - You can’t afford not to have a dedicated build machine
  - Is your competition using automated building & testing?

CruiseControl

- Requires a separate build machine



Title: Softpedia

Please enter the dashboard title.

Background Color: 🟩

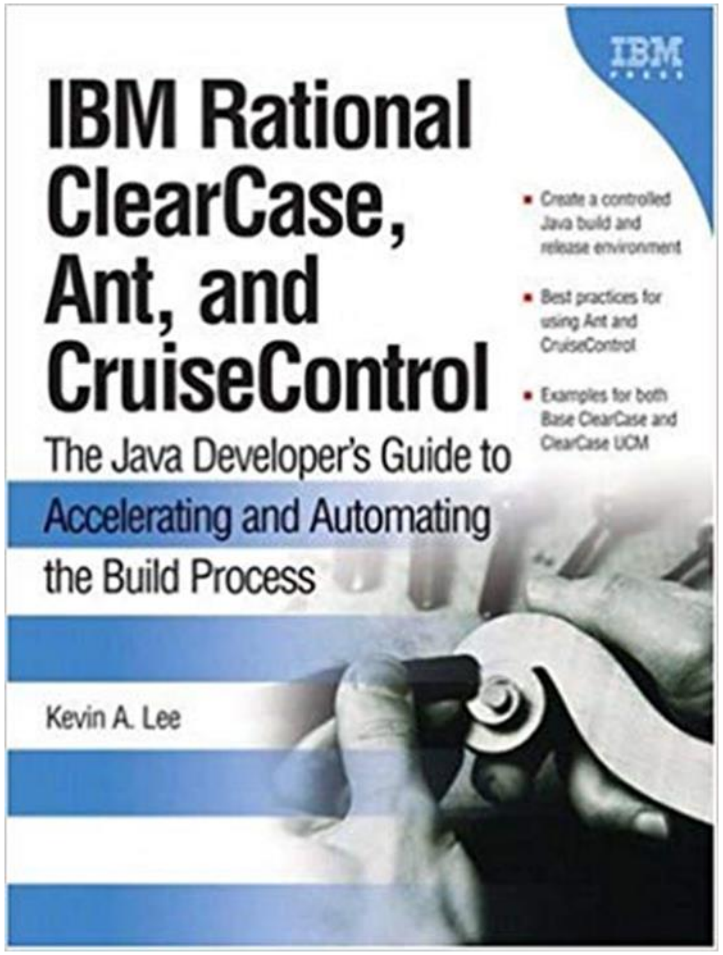
Please choose the dashboard background color.

Update Interval: Hourly

Please choose the time interval to automatically update the dashboard. Note: You can also manually update by clicking on "(Update)" on the dashboard.

CruiseControl

- Automated Builds
  - Push a button
- Scheduled Builds
  - Don’t need to push any buttons


IBM Press

IBM Rational ClearCase, Ant, and CruiseControl

The Java Developer's Guide to Accelerating and Automating the Build Process

- Create a controlled Java build and release environment
- Best practices for using Ant and CruiseControl
- Examples for both Base ClearCase and ClearCase UCM

Kevin A. Lee

For Continuous Integration ...

- SCM system
  - To ensure latest code changes are included
- Automated or (better) scheduled builds
  - To detect and resolve build issues early and often
- Automated unit tests
  - To detect and resolve unit and integration issues early
- (Not really part of Continuous Integration, but some teams also like to automate some acceptance tests)

Developer’s Goals

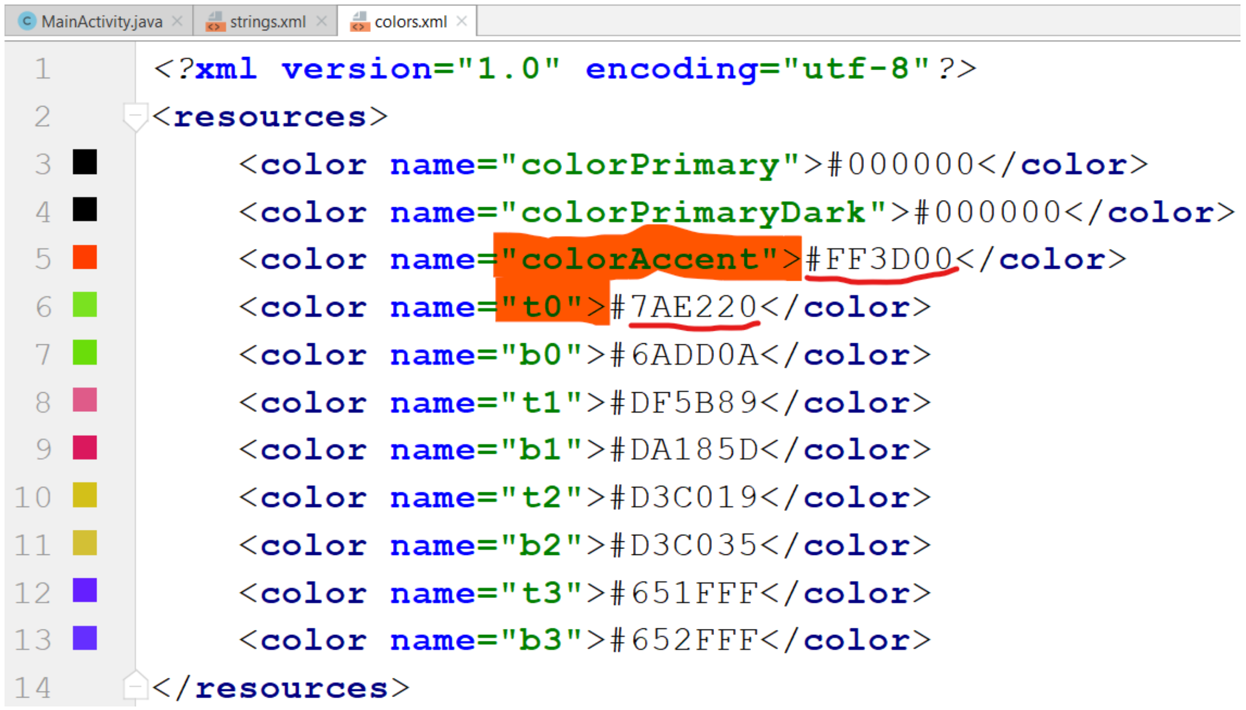
- Deliver working production code
  - Measurable quality
  - Resulting in happy customers
- Be productive
  - Intelligent use of time
  - Address issues associated with software changing over time
- Work smarter, not harder
  - Processes
  - Tools

Three Types of Code Changes

- Fix defects
- New requirements
- Refactoring
  - Refactoring is NOT the same as fixing defects
  - You end up with exactly the same functionality as before, just (hopefully) simpler code

Making Code Changes

- If you make even a simple change, how do you know that you haven’t broken anything?


MainActivity.java × strings.xml × colors.xml ×

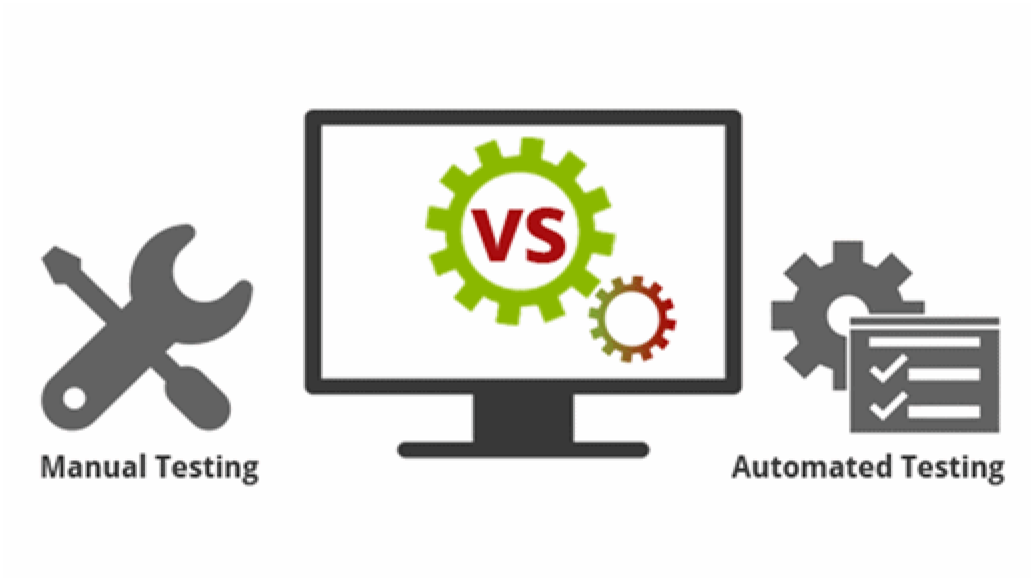
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#000000</color>
    <color name="colorPrimaryDark">#000000</color>
    <color name="colorAccent">#FF3D00</color>
    <color name="t0">#7AE220</color>
    <color name="b0">#6ADD0A</color>
    <color name="t1">#DF5B89</color>
    <color name="b1">#DA185D</color>
    <color name="t2">#D3C019</color>
    <color name="b2">#D3C035</color>
    <color name="t3">#651FFF</color>
    <color name="b3">#652FFF</color>
</resources>
```

Making Code Changes

- You MUST have automated unit tests
- Imagine a large application with many people changing it
  - Adding new features
  - Fixing defects
  - Refactoring code
- Continuous Integration will help you address most of your integration problems early – if you build your automated test cases correctly

Automated Unit Test Cases

- Create your unit test cases first, then change the code until the test cases pass
  - Adding new features
  - Fixing defects



SCM & Continuous Integration Perceived Weaknesses

- Too much overhead (time and money)
  - Overhead is much less than time and effort required to fix integration issues late in the project
- Requires discipline by the project team
  - This is a good thing
  - Soft measures of project maturity:
    - ✓ Who does your builds?
    - ✓ How easy it is for any developer to make a change, test it and commit the change for other developers to use

Summary - 1

- Build, compile and debug software
  - eclipse
- Build, compile and debug software where source code is stored in a shared repository
  - eclipse
  - svn
- Build, compile and debug software where source code is stored in a shared repository and builds are automated
  - eclipse
  - svn
  - Ant

Summary - 2

- [Continuous Integration] Build, compile and debug software where source code is stored in a shared repository and builds and unit testing are automated
  - eclipse
  - svn
  - Ant
  - JUnit

Summary - 3

- [Continuous Integration] Build, compile and debug software where source code is stored in a shared repository and builds and unit testing are scheduled
  - eclipse
  - svn
  - Ant
  - JUnit
  - CruiseControl

Summary - 4

- Different tools for different environments and development languages
  - JUnit (CppUnit, ...)
  - CruiseControl (Anthill, ...)
- Take advantage of the many good open source tools to increase:
  - Your productivity
  - The quality of your code

Video link

- https://www.youtube.com/watch?v=qOhfl9gayB8
- https://www.youtube.com/watch?v=SDwqcFwvwY0

References

- Ian Sommerville (2016). *Software engineering update 10th edition.* Wesley Computer Publishing. PP 735- 756