CST8284 Part 2 JUnit Technical Brief

JUnit testing with JUnit4

What is JUnit?

- A framework used to automate unit testing in the Java programming language
- Implemented as a Java file with test methods
- See the recommended readings and additional resources for more details (end of this handout).

General Workflow within each test case (@Test method) inside a Test-Class:

1. Prepare objects and variables **(use meaningful variable names)**
2. Perform one task (typically changing the objects state, getting a return value etc.)
3. Check results for that one task using an appropriate assert method
4. Perform tear-down of resources to prep for next test (e.g. close resources, set reference variable to null etc)

What are some commonly used assert methods used with JUnit?

(Taken from http://junit.sourceforge.net/javadoc/org/junit/Assert.html)

assertEquals(java.lang.String message, double expected, double actual, double delta)
Asserts that two doubles or floats are equal to within a positive delta.

assertEquals(java.lang.String message, long expected, long actual)
Asserts that two longs are equal.

assertEquals(java.lang.String message, java.lang.Object expected, java.lang.Object actual)
Asserts that two objects are equal.

assertFalse(java.lang.String message, boolean condition)
Asserts that a condition is false.

assertTrue(java.lang.String message, boolean condition)
Asserts that a condition is true.

assertNotNull(java.lang.String message, java.lang.Object object)
Asserts that an object isn't null.

What is a tolerance level (delta or epsilon) when comparing double values?

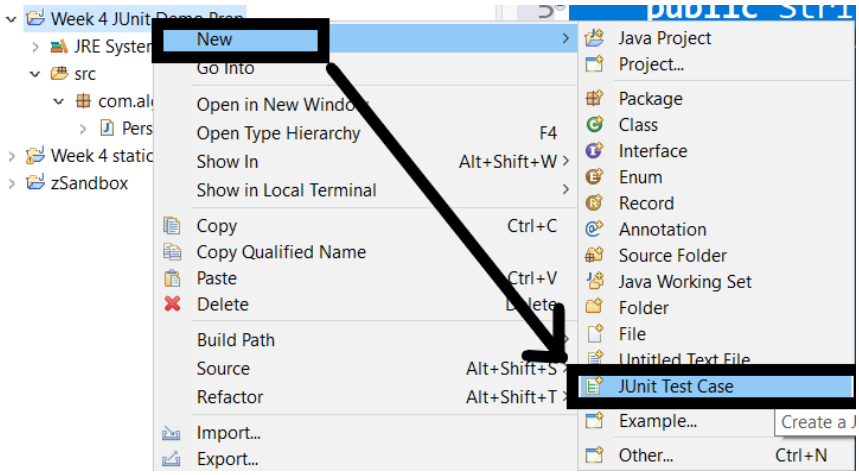
- The smallest difference between the two values at which point they are considered equal (i.e. close enough)
- A rough illustration is given two floats A & B: Math.abs(A - B) <= epsilon as true means A and B are the "Same"
- See this website for much more information on comparing floating point numbers using epsilon values
  Random ASCII. (2012). Comparing Floating Point Numbers, 2012 Edition. Retrieved from https://randomascii.wordpress.com/2012/02/25/comparing-floating-point-numbers-2012-edition/

How to create a simple JUnit test in Eclipse - Hands On

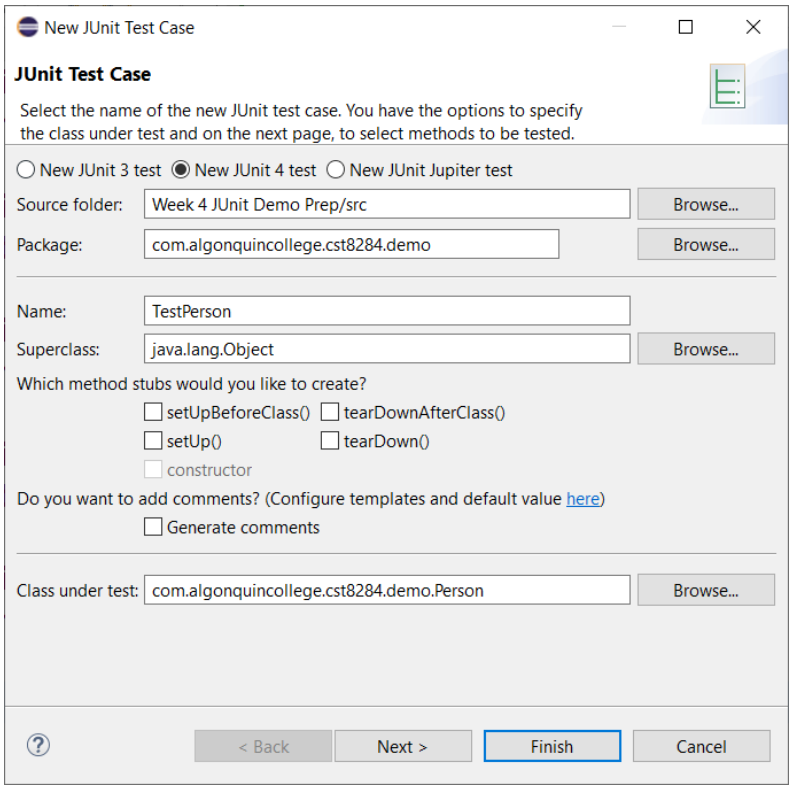
- Create a new Project E.g. “Week 4 JUnit Demo”
- Create a new package “com.algonquincollege.cst8284.unittest.demo”
- Create a new class inside the package, name it Person.
- Use the following code sample (copy and paste if you like)
  - Note: getFullName() has a mistake in it, no space between names, this is intended.

```
package com.algonquincollege.cst8284.unittest.demo;
public class Person {
	private String firstName;
	private String lastName;
	public String getFirstName() {
		return firstName;
	}
	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}
	public String getLastName() {
		return lastName;
	}
	public void setLastName(String lastName) {
		this.lastName = lastName;
	}
	public String getFullName() {
		return firstName + lastName;
	}
}
```

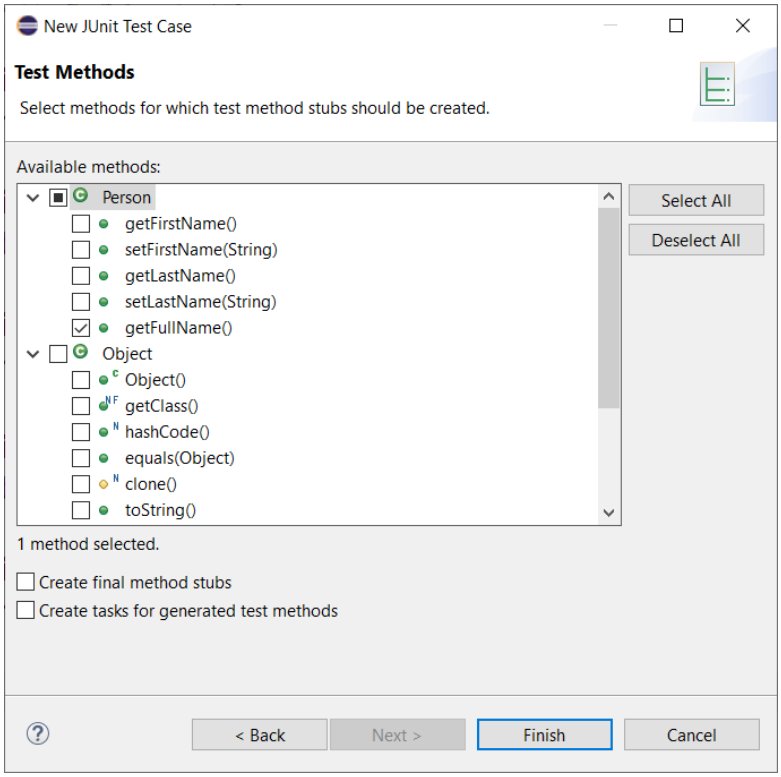
- Right-Click on the Project.
- Use New > JUnit Test Case



- Select “New JUnit 4 test”

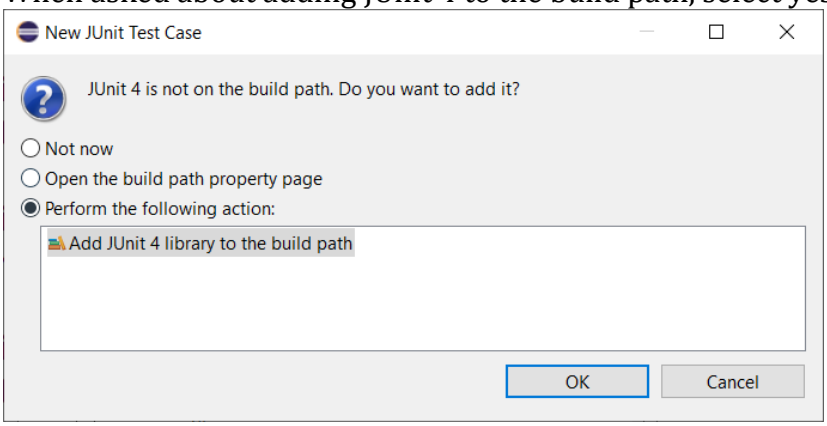


- Name the test-case “TestPerson”
- For this introduction, we will place the TestPerson class in the same package as the class-under-test.
- Alternatively, a different package name can be used to separate test classes from the production code.
- Use the Browse button and type Person in the upper box to locate class Person as the class-under-test.
- Use the Next > button



- We can select what methods we want test-stubs to be created. Select method getFullName() for this demo
- Click Finish

When asked about adding JUnit 4 to the build path, select yes.



This is the generated code:

```
package com.algonquincollege.cst8284.demo;

import static org.junit.Assert.*;
import org.junit.Test;

public class TestPerson {

	@Test
	public void testGetFullName() {
		fail("Not yet implemented");
	}

}
```

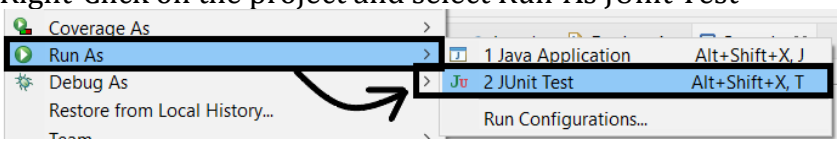
To conduct the test:

1. Set up for the test
    a. Instantiate Person
    b. Set values
    c. Set expected result
    d. Get actual result
2. Perform the test
    a. Use assertEquals(String message, Object expected, Object actual)
3. Tear down after the test
    a. Set reference variable to null

Code Sample next page.

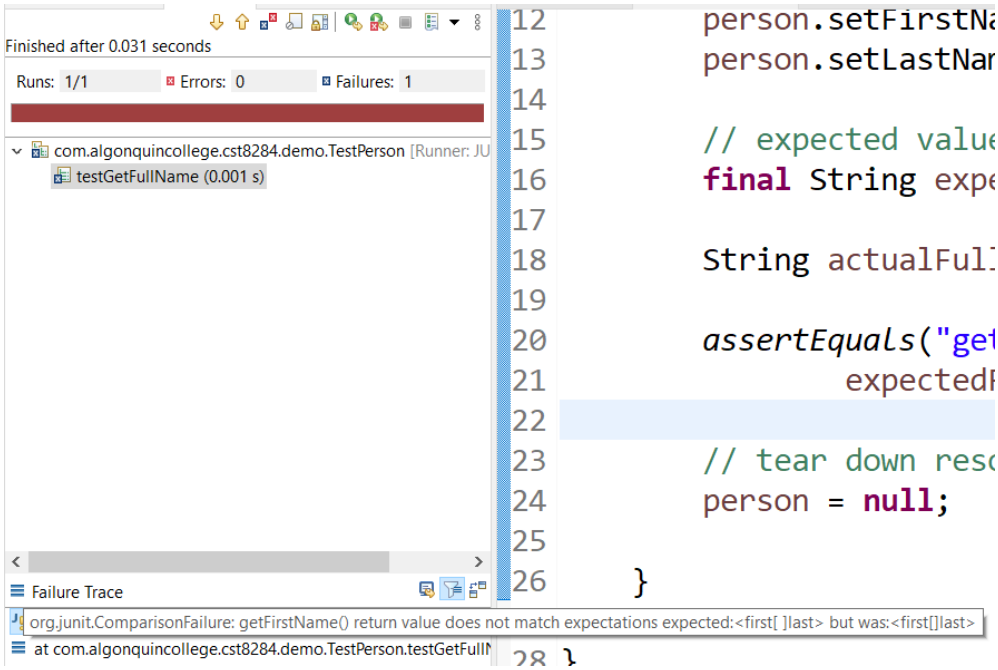
```
@Test
public void testGetFullName() {
	// set up the test
	Person person = new Person();
	person.setFirstName("first");
	person.setLastName("last");

	// expected value or result
	final String expectedFullName = "first last"; // note the space inside

	String actualFullName = person.getFullName();

	assertEquals("getFirstName() return value does not match expectations",
			expectedFullName, actualFullName);

	// tear down resources i.e. make avail for Garbage Collector
	person = null;
}
```

Right-Click on the project and select Run-As JUnit Test



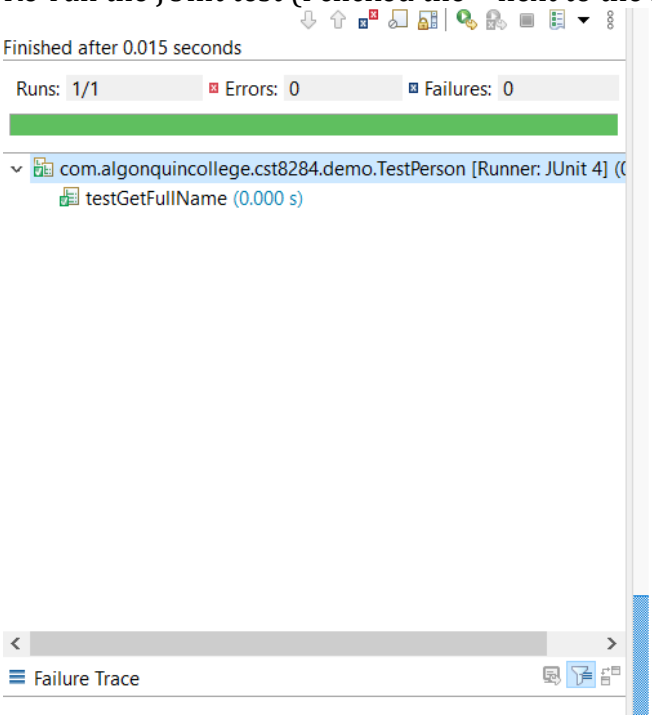
View the test results

- The original class Person had a bug (intentionally) so that no space is placed between the first and last names.
- The assertEquals detects this, and the test fails i.e. Failures: 1
- The tool-tip for the failed test has the message we set within it, and some additional information.
- Org.junit.ComparisonFailure: getFirstName() return value does not match expectations expected:<first[ ]last> but was <first[]last>
- The testing framework is using square brackets to indicate we are missing a space, i.e. [ ] vs []



Return to class Person and add a space between firstName + lastName i.e. firstName + " " + lastName

Re-run the JUnit test (I clicked the > next to the first line to see the test method names)



The recommended learning resources will take you beyond the basics.

Recommended Resources (School Library Videos + Books)

Steps: Visit: http://www.algonquincollege.com/library/
Use the Digital Resources button at the top
Use the letter S button to access Safari by O'Reilly

Safari Books: JUnit Testing

JUnit in Action, Second Edition
By: Petar Tahchiev; Felipe Leme; Vincent Massol; Gary Gregory
Publisher: Manning Publications
Pub. Date: July 28, 2010
Print ISBN-10: 1-935182-02-1
Print ISBN-13: 978-1-935182-02-3

Java Programming Interviews Exposed
By: Noel Markham
Publisher: Wrox
Pub. Date: February 17, 2014
Print ISBN-13: 978-1-118-72286-2
Web ISBN-10: 1-118722-86-8
eISBN-13: 978-1-118-72288-6
(- This book may help you after graduation (read it before then))
(- See Chapter 9: Testing with JUnit)

Additional Recommended Resources on JUnit (YouTube / Web)

David Whitlock. (2013). Unit testing with JUnit. [Video] Retrieved from
https://www.youtube.com/watch?v=k1DE9H8EGNA
(We won't be using HamCrest, or IntelliJ but the code samples and lecture are quite good)
(43 minutes)

McProgramming. (2014). Java - JUnit testing in Eclipse. [Video] Retrieved from
https://www.youtube.com/watch?v=I8XXfgF9GSc
(11 minutes)

JUnit tutorial in Eclipse part 1 : javavids. (2012). [Video] Retrieved from
https://www.youtube.com/watch?v=QEyxgtCEWMw&list=PL0951947FC3CB5BB3
(There are actually 9 videos here, parts 1 through 9)
(Parts 1 through 4 I highly recommend)
(Parts 5 through 9 cover more advanced topics... it looks like 7 and 8 are the same video)

Other reference:

junit.sourceforge.net. (n.d.). JUnit API. [Webpage] Retrieved from
http://junit.sourceforge.net/javadoc/

JUnit. (2014). JUnit. Retrieved from
https://junit.org/junit4/ (many learning resources)

Lars Vogel. (2015). Unit Testing with JUnit – Tutorial. Retrieved from
https://www.vogella.com/tutorials/JUnit4/article.html

While we are using JUnit 4 in our course, a new JUnit is available. See these resources for future learning:

JUnit. (2021). JUnit. Retrieved from
https://junit.org/junit5/ (many learning resources)

Lars Vogel. (2021). Unit Testing with JUnit – Tutorial. Retrieved from
https://www.vogella.com/tutorials/JUnit/article.html