Contents

[Naming a class - what is the big deal? 1](#_Toc494017896)

[On references and memory 2](#_Toc494017897)

[equals() and hashCode() methods 3](#_Toc494017898)

[Choosing the right Java Data Type 3](#_Toc494017899)

[Creating Subfolders in Eclipse 5](#_Toc494017900)

[Junit FAQ 8](#_Toc494017901)

[Junit tests not found by eclipse 8](#_Toc494017902)

[Using JUnit with Eclipse 9](#_Toc494017903)

[Java Language and Virtual Machine Specification 11](#_Toc494017904)

# Naming a class - what is the big deal?

Naming of domain classes seems fairly straightforward for some of us, and for few of us, yet it seems elusive. For example, a domain class called **Customer** appears very clear and applicable to any business. But in my experience, in one particular industry various applications within the organizations called this very entity with a different name, in this case it was called a **Subscriber**.

This simple issue led to some painful confusion as in times it was required that the various applications interact with each other with **Customer** and **Subscriber** objects flowing between them. After duration of time it was realized that Customer and Subscriber were synonymous. However, the manner in which they were used and associated with other domain entities made the application integration a task by itself.

With Service Oriented Architecture (SOA, http://en.wikipedia.org/wiki/Service-oriented\_architecture) being a buzz in the last few years, SOA prescribes to among other things a taxonomy definition of ones business model. It’s here one goes about making sure that everyone in the organization speaks the same lingo so chaos from ‘what is a Customer and Subscriber’ are be avoided.

If such fundamental issues are not resolved earlier on, it is bound to take a lot of effort and time to fix it. Case and point, British Telecom (considered one of the largest Telco’s in the world) took two years to define their taxonomy and clean up their vocabulary! One can surmise that chaos ruled till this exercise was done! Now, how many CIOs/CEOs are willing let a core team go reinvent themselves for 2 years, let alone 6 months!? What is the moral of this story? Get the taxonomy down the first time with both the technical and business counterparts and socialize this across the organization so it becomes a part the company’s DNA.

Hopefully I shared with you the importance of understanding the business requirements and building the correct taxonomy via domain classes. You might not notice this being an issue in your academic exercises, but something to keep in mind as you build enterprise solutions.

For our course standpoint, you can still make an effort to build a meaningful domain classes that each have a defined responsibility and effectively collaborate with other classes.

For example, in a typical e\*commerce domain - Customer class holds customer semantics while it collaborates with the Orders class. Orders class holds orders associated with the customer, while it collaborates with an OrderDetails class which contains details of the specific order.

While the focus of this class is not strictly learning OOAD (Object-Oriented Analysis and Design), this is a good place to start as any.

Please let me know if you have any questions.

# On references and memory

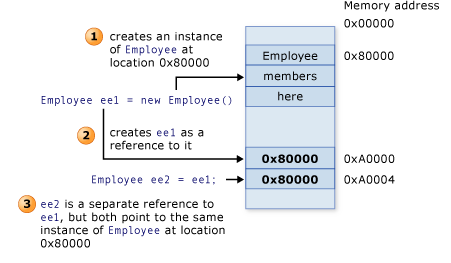
Consider the code:

Employee ee1 = new Employee();

Employee ee2 = ee1;

How to read the code:

1. ee1 is known as a reference to Employee
2. The first line creates an instance of Employee in memory
   1. And sets ee1 to reference it.
3. In line2, when ee2 is set to equal ee1, it now contains a duplicate of the reference to the class in memory.
   1. If you now change properties on ee2, properties on ee1 reflect these changes. This because both point to the same object in memory as shown below:



Source: https://msdn.microsoft.com/en-us/library/ms228360(v=vs.90).aspx

# equals() and hashCode() methods

equals() and hashCode() are very important methods in a class (primarily in Domain layer classes). Please see following links for details on them.

1. [**http://tinyurl.com/lh9wwgf**](http://tinyurl.com/lh9wwgf) **- Joshua Bloch on equals/hashCode**
2. Study the domain class 'Car' in attachment for critical information on 'equals' and 'hashCode'.
3. Supplemental Read: Read <http://www.artima.com/lejava/articles/equality.html>

# Choosing the right Java Data Type

<http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html>

If you review the content in the above link, you’ll notice that much like in other languages, primitive data types have a certain size.

A punctilious developer would use this to decide why a certain variable should be say a short vs. an int.

For example, let’s take ‘age’, it’s not uncommon to see it being declared as

**int** age;

But, in Java, this data type is a 32-bit signed two's complement integer. With a minimum value of **-2,147,483,648** and a maximum value of **2,147,483,647** (inclusive)!

**Question:**

In a business app, what would ‘age’ range be (if we were dealing with humanoid’s, then it’s a different matter! ☺ )?

Would the one below be better?

**short** age;

This data type is a 16-bit signed two's complement integer. It has a minimum value of -**32,768** and a maximum value of **32,767** (inclusive)!

How about this?

**byte** age;

This data type is an 8-bit signed two's complement integer. It has a minimum value of

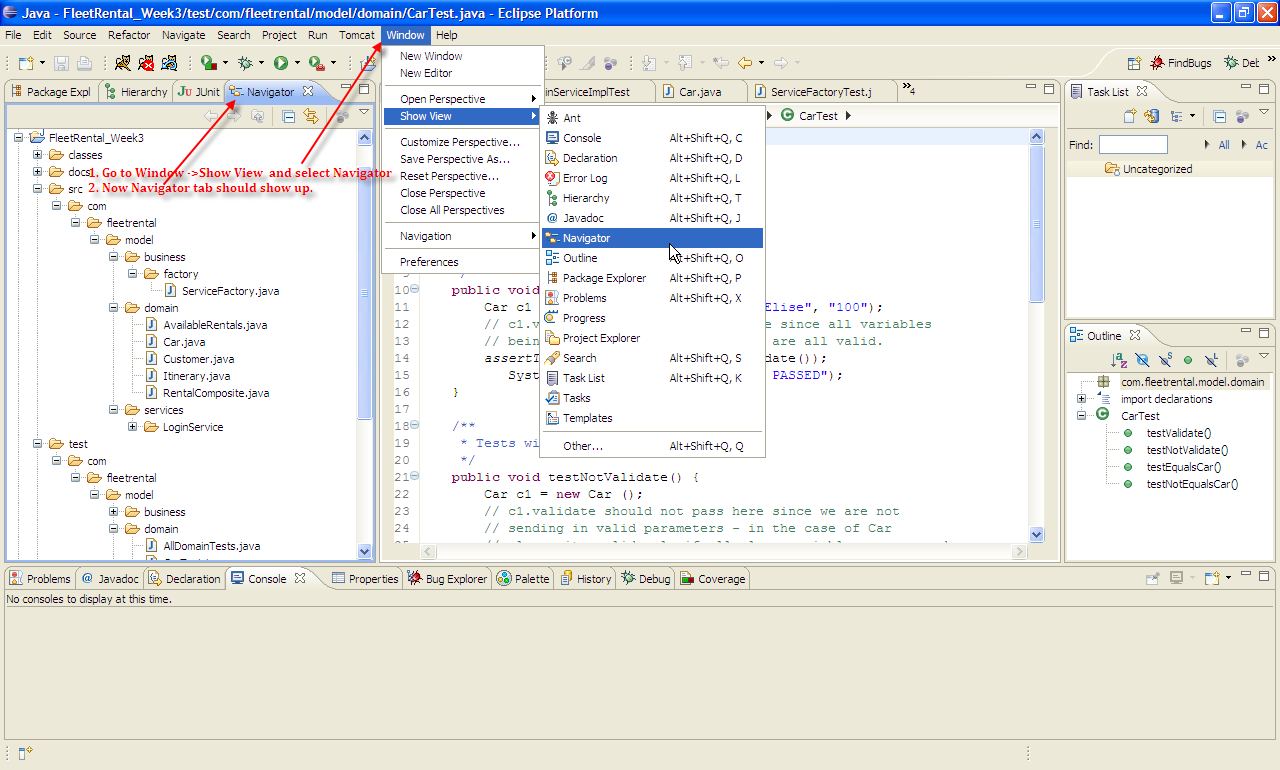
**-128** and a maximum value of **127** (inclusive)!

Point here is, using the right data type, instead of larger then needed (hence wasting memory), demonstrates a discipline (which comes with experience) and attention for details.

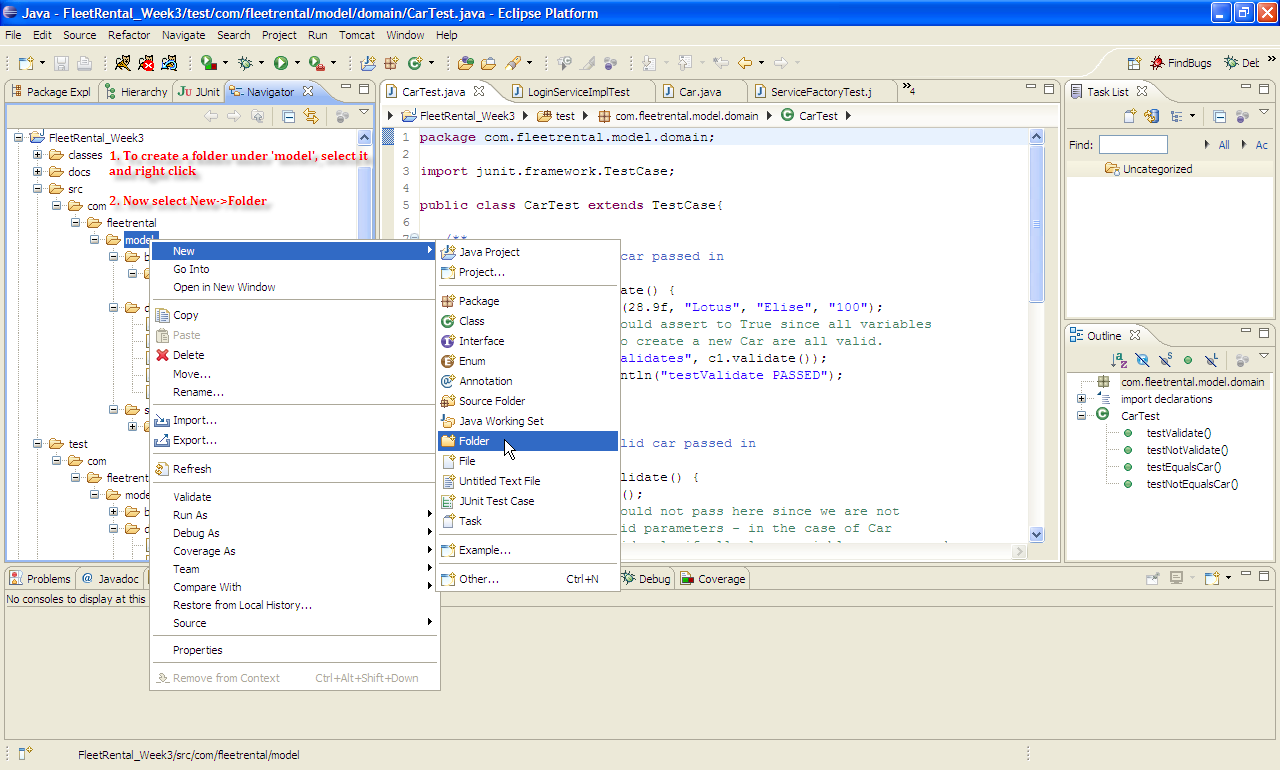
This is a common error we all make initially so nothing to feel bad about! However, having read the above, can you please revisit your discussions and see where you could tweak it. ☺

# Creating Subfolders in Eclipse

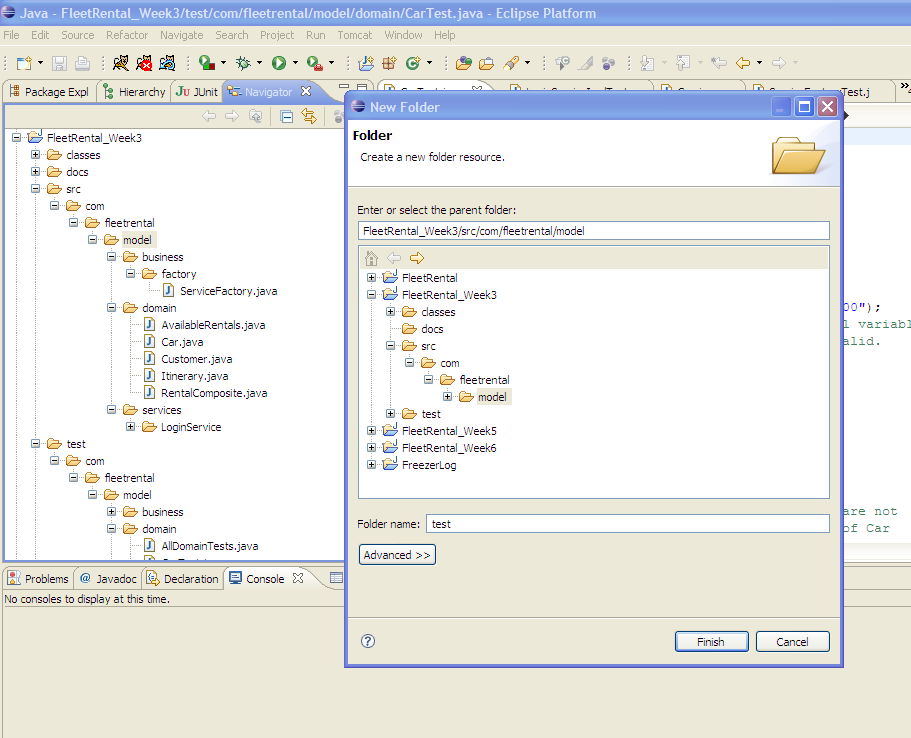
1. Go to Navigator tab as shown below.



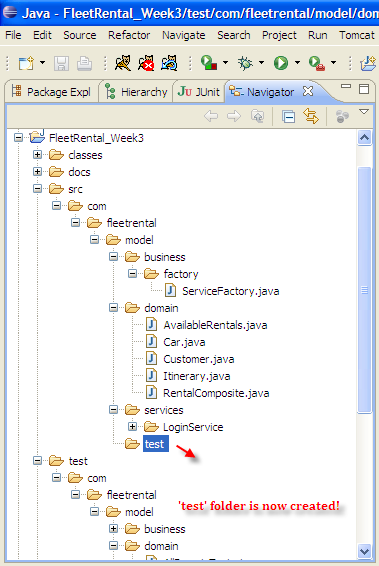
2. Now, say we want to create a folder called “test” under “model” folder. To do this follow instructions in image below.



3. In the ensuing dialog, enter “test” as the new folder to create as shown below:



4. Now you should see the “test” folder as shown below. To create subfolders under “test”, keep repeating above process.



# Junit FAQ

<http://junit.sourceforge.net/doc/faq/faq.htm#tests_2>

# Junit tests not found by eclipse

If you encounter this scenario:

<https://stackoverflow.com/questions/20057771/no-junit-tests-found-in-eclipse>

My other reason to provide this link is to get you acquainted (if you are not already) with stackoverflow.com. This today is by far the most prevalent place for all technical questions and answers professionals go to. It’s evident in google search rankings to its popularity!

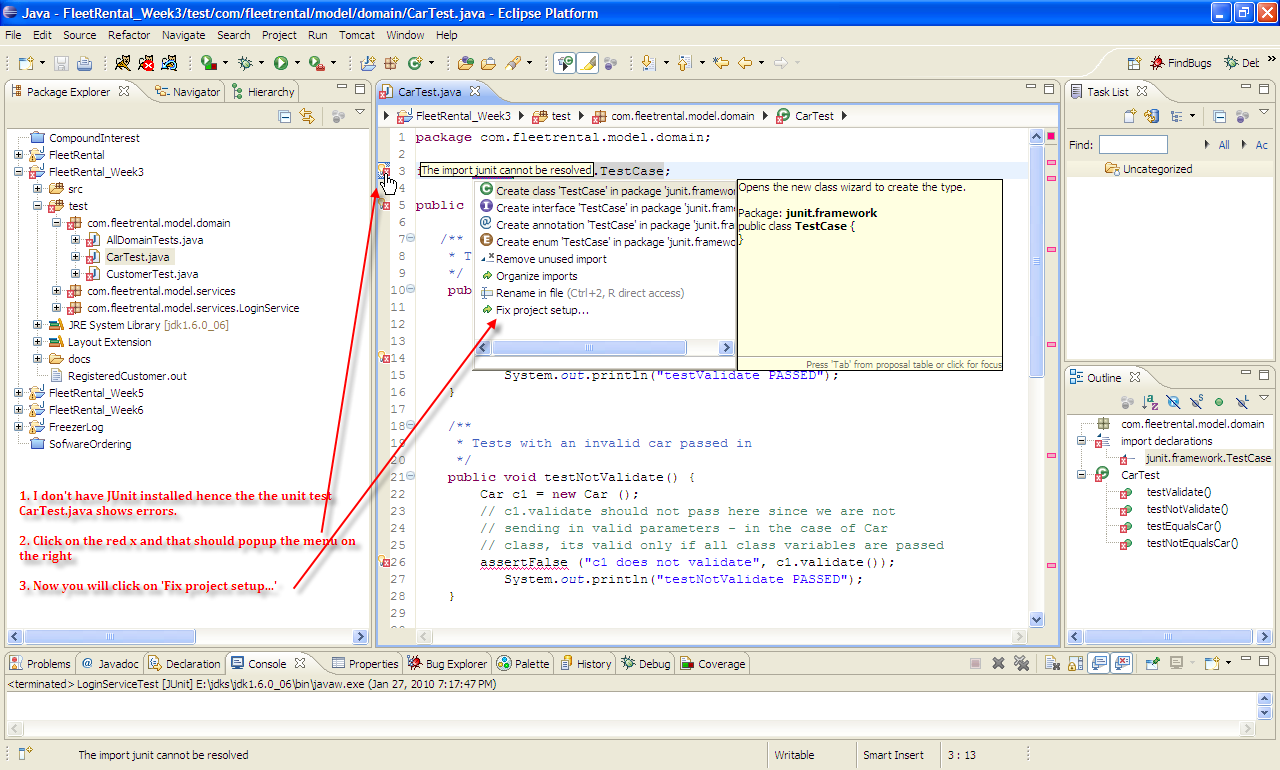
# Using JUnit with Eclipse

**Quick tutorial on using JUnit with Eclipse**

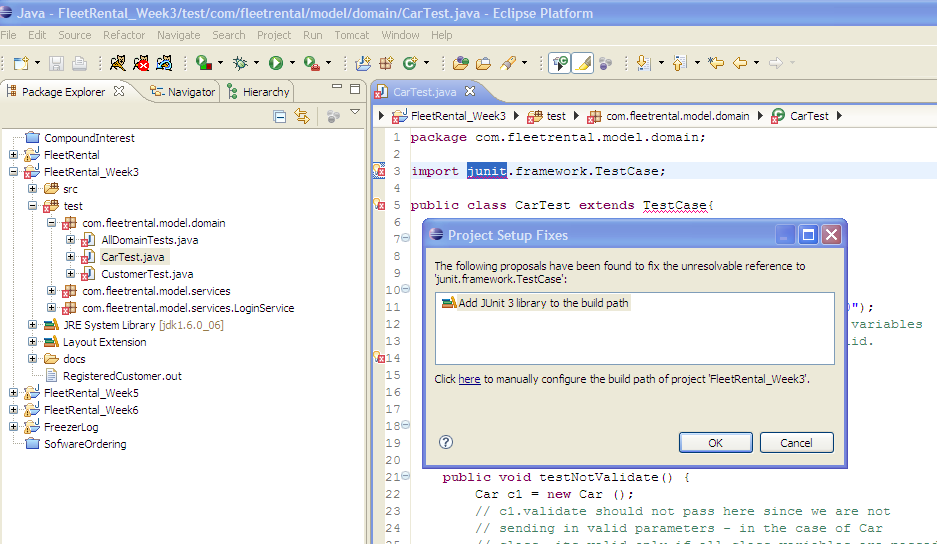
**Note: Though this document refers to JUnit 3, use the latest version that comes with Eclipse.**

This document shows how to install or update JUnit in case it is not already installed in your Eclipse.

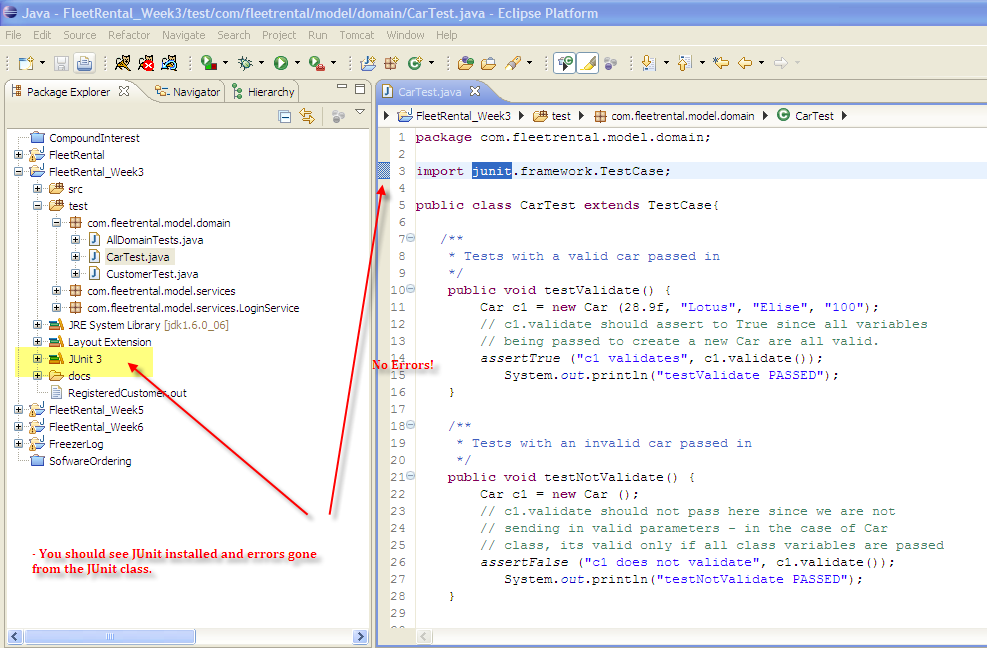
1. Follow instructions on the image below.



2. Clicking on ‘Fix project setup…’ should get you the ‘Project Setup Fixes dialog’ below.



3. Click OK on the ‘Project Setup Fixes’ dialog



**Additional Reference**

1. **Junit Tutorial** : <http://www.tutorialspoint.com/junit/>
   1. This link provides additional material and a good website to bookmark.

# Java Language and Virtual Machine Specification

I always use the specifications to get precise details. For my professional work, text books lag behind key details. Recommending bookmarking this page for future reference.

<http://docs.oracle.com/javase/specs/#7.7>