Effective Object-Oriented Design

Outline	Learn how to design high quality OO solutions using Test Driven Development (TDD), refactoring and patterns. The course is lab based, with every new concept backed up by a comprehensive set of exercises that allow you to validate your understanding of the material. Participants will work in pairs.
Pre-requisites	Previous development experience with Java and the Unified Modelling Language (UML). Alternatively, attendance at our "Effective Object-Oriented Programming Using Java" and "UML Notation" courses would satisfy this pre-requisite.
Duration	4 days
Daily Schedule	9.30am - 5.30pm
Maximum attendees	10

#	Module	Objectives	Minutes	Type
1	"Hello World" Unit Test	• Gain familiarity with the JUnit testing tool through writing your first unit test. • Learn how to run several tests together. • Learn about the development tools that you will be using namely JUnit, Ant, and the IDE.	30	Lab
2	Simple Test Driven Development Example	• Gain insights into Test Driven Development though a simple example that illustrates the essential elements of the approach.	60	Presentation
3	Test Driven Development Exercise	• Use the TDD approach for an exercise. (This will include time for discussing the different solutions and issues that were raised for you during the exercise.)	120	Lab
4	Refactoring Introduction	• Gain familiarity with the refactoring technique. • Undertake some initial code refactorings assisted by a refactoring tool.	30	Lab
5	Why Refactor? – 'Bad Code Smells'	• Learn bad code smells and the types of refactorings they lead to. (The trick with refactoring is to know when to do it and which refactoring is appropriate.)	60	Presentation
6	Refactoring Exercise	• Learn how to remove complexity from existing code using refactoring. This exercise will allow you to refactor poor code with the safety net of a bank of pre-existing unit tests.	90	Lab
7	Introduction to Mock Objects	• Learn how to unit test your code when it needs to collaborate with complex resources such as databases and application servers. • Write a simple unit test that tests against a mock version of a real	30	Lab

Centre for SOFTWARE Engineering

		resource.		
8	Mock Objects	• Learn about the why and when of Mock Objects. • Learn about Mock Object related testing patterns such as self-shunting and control flow testing. • Understand how Aspect Oriented Programming can be used to replace real objects with their mock cousins for unit testing purposes.	60	Presentation
9	Mock Objects Exercise	• Refactor code to use mock JDBC classes instead of the real thing.	120	Lab
10	Design Exercise	• Use the techniques acquired in the previous modules to tackle a non-trivial design problem. • Review the techniques learnt and utilised in the previous modules.	360	Lab
11	Anatomy of a Design	• Learn how design patterns provide a language that allows designers to communicate and reason with each other about their designs. This is achieved by studying a pattern rich application.	60	Presentation
12	Design Patterns and TDD	• Learn how patterns 'emerge' in a design, when a Test Driven Development (TDD) approach is taken.	60	Presentation
13	Acceptance Testing Introduction	• Learn how to write user acceptance tests using the FIT open-source testing tool.	30	Lab
14	Acceptance Testing	• Learn how to get users to express their requirements as Acceptance tests. • Learn how to drive your development with these tests. • Learn how FIT complements commercial GUI testing tools.	60	Presentation
15	Acceptance Testing Exercise	• Taking the role of a customer, learn how to specify requirements in the form of user acceptance tests.	150	Lab
16	Designing the User Interface	• Learn how to deal with the evolution of the user interface when using a Test Driven Development (TDD) approach.	60	Presentation
17	Designing for Concurrency	• Learn the fundamentals of dealing with concurrency in a TDD context.	120	Presentation
18	Summary	• Ask questions and raise issues on what has been learnt during the course.	60	Discussion