Course Outline

COMP4001
Object Oriented Software Development
Teaching Period 2 2010

Contents

- Course Staff
- Course Details
- Course Summary
- Course Aims
- Student Learning Outcomes
- Assumed Knowledge
- Teaching Rationale
- Teaching Strategies
- Assessment
- Academic Honesty and Plagiarism
- Course Schedule
- Resources for Students
- Course Evaluation and Development

Course Staff

Contact Details

Staff Name	Role	Email	Phone
John Potter	Course Convenor	potter at cse.unsw.edu.au	54334

Consultations

Staff Name	Day/Time	Location
John Potter	Wednesday 1500-1600	CLB-1 after lecture
John Potter	Thursday 1000-1200	K17-217E
Trevor McDonell	Thursday 1500-1600	K17-204 consult room
Trevor McDonell	Friday 1200-1400	K17-204 consult room

Course Details

Course Code: COMP4001
Course Title: Object Oriented Software Development
Units of Credit: 6
Course WebSite: http://www.cse.unsw.edu.au/~cs4001/
Handbook Entry: http://www.handbook.unsw.edu.au/undergraduate/courses/2010/COMP4001.html

Course Summary

This course considers a selection of advanced topics in object oriented software development, with a particular focus on modern developments in OO programming languages and approaches to design. In 2010 the course comprises two major topics:

- concurrent object oriented programming (withJava)
- object oriented / functional programming (with Scala)

Course Aims

This course aims to

- achieve deeper understanding of OO concepts
- develop familiarity with concurrent programming in Java
- gain experience with higher order functional programming in an OO context, using the Scala programming language
- use a number of different frameworks for concurrent programming

- design and implement small domain specific languages
- understand issues in designing for reusability and in library design
- develop understanding of advanced OO type systems, with type parameters and abstract types

Student Learning Outcomes

After completing this course, students will:

- know concurrent programming concepts in an OO context
- understand concepts of thread safety and liveness
- be able to use the concurrency c
- be able to write safe and reliable multithreaded code in Java
- be able to write parallel programming applications with a thread-join framework
- understand higher order functional programming with functions and the notion of closure for abstracting functions from code
- understand higher order techniques for generic programming with functions like map, filter, fold etc, and the concept of a combinator
- be able to write simple parsers using a parser combinator library
- be able implement small domain specific languages
- be able to write concurrent applications using a concurrent actors framework

This course contributes to the development of the following graduate attributes:

Graduate Attribute	Where Acquired
the skills involved in scholarly enquiry	concurrency and high level programming language continue to be hot research topics. This course exposes students to the leading edge of some of this research.
an in-depth engagement with relevant disciplinary knowledge in its interdisciplinary context	this course builds on fundamentals of program design achieved in earlier computing courses. The Domain Specific Language component gives the opportunity to draw on other disciplines, but will not be a focus of this course.
the capacity for analytical and critical thinking and for creative problem solving	programming assignments will require detailed problem analysis, and careful program design involves creative problem solving
the ability to engage in independent and reflective learning	as an advanced course, students are expected to learn most aspects of this course independently. Lectures serve as a guide to the content, but do not cover all required knowledge.
the skills to locate, evaluate and use relevant information (Information Literacy)	self study via online resources should be a major part of a student's learning process in this course. This will be assessed by their ability to complete the programming assignments.
the capacity for enterprise, initiative and creativity	through challenges set in homework exercises and programming assignments
an appreciation of and respect for, diversity	by exposure to external Wikis and other online forums providing international discussions and comments on advanced programming techniques and language features, students have the opportunity to observe and join a diverse programming community
a capacity to contribute to, and work within, the international community	as above
the skills required for collaborative and multidisciplinary work	n/a
an appreciation of, and a responsiveness to, change	by studying leading edge programming techniques and languages, students will gain skills in adapting to future changes in software technology
a respect for ethical practice and social responsibility	by mature participation in course message boards, and discriminating between appropriate shared discussion of assignment problems and avoiding plagiarism of other students solutions
the skills of effective communication	through participation in lecture discussions, message boards and consultation sessions. There is no emphasis on written communication in this course.

Assumed Knowledge

Before commencing this course, students should:

- know basic object oriented programming concepts, including classes, interfaces, abstract classes and methods, method implementation and overriding, dynamic binding
- know basics of class diagrams for OO design
- understand design by contract in an OO context (i.e. pre- and postconditions for methods, and invariants for classes)
- be able to implement object oriented designs in Java (or similar language)
- be able to implement unit tests, preferably with a testing tool such as JUnit
- be able to use a debugger for Java (such as in the eclipse IDE)

These are assumed to have been acquired in COMP2911 for undergraduate students, or its equivalent for postgraduate students

Teaching Rationale

As an advanced level course, students are expected to adopt a mature and self-motivated approach to their studies. The lectures cover core course material, but student will need to read online tutorial material to get a full understanding of the course content.

This course is intended to develop advanced programming skills, for which students need to have the skills to discover solutions to problems by accessing a wide range of online resources.

The programming assignments are intended to force students outside of their comfort zones, and present them with problems which assist in developing both their programming and learning of programming language skills.

Teaching Strategies

- Lectures ... introduce concepts, discuss examples of code, live demonstration of code development, execution and debugging
- Exercises ... provide self-assessed programming exercises for reinforcement of lecture content
- Self Study ... students should read suggested programming tutorials and online resources to expand the coverage of the lecture material.
- Assignments .. allow students to solve significant problems
- Consultations ... face to face contact with staff for assistance with problems and assignments
- Message Board ... for class-wide discussion of concepts and examples covered in lectures, and for assistance with programming problems and assignments

Assessment

50% Four Individual Programming Assignments

1. 10% (Due end of Week 4) Multi-threaded programming in Java
2. 15% (Due end of Week 7) DSLs in Scala
3. 15% (Due end of Week 10) Advanced synchronization in Java
4. 10% (Due end of Week 13) Actor-based application in Scala

50% Final Examination

Academic Honesty and Plagiarism

Plagiarism is defined as "using the words or ideas of others and presenting them as your own". UNSW and CSE treat plagiarism as academic misconduct, which means that it carries penalties as severe as being excluded from further study at UNSW. There are several on-line sources to help you understand what plagiarism is and how it is dealt with at UNSW:

- Learning Centre: Plagiarism and Academic Integrity
- MyUNSW: Plagiarism and Academic Misconduct
- CSE: Addendum to UNSW Plagiarism Guidelines
- CSE: Yellow Form (whose terms you have agreed to)

Make sure that you read and understand these. Ignorance is not accepted as an excuse for plagiarism.

Course Schedule

The course has been designed to be covered in two parallel streams. The breakdown for each module is as follows:

Concurrent Object Oriented Programming

- Weeks 1-3
  Introduction to concurrency in Java, Threads, Thread Safety, Concurrent Collections in Java
- Weeks 4-5
  Java Concurrency Framework, Tasks, Thread and Task Control, Fork-Join Framework, Data Flow Models
- Weeks 6-7
  Advanced Synchronisation, Concurrent Data Structures
- Weeks 8-9
  Liveness, Deadlock, Performance
- Weeks 10-12
  Advanced Topics: Custom Synchronisers, Java Memory Model, Non-blocking Synchronisation

Object Oriented / Functional Programming

- Weeks 1-3

Introduction to Scala. Classes, Traits, Objects. Multiple inheritance and mixins. Pattern matching.
- Weeks 4-6
  Functions as objects. Higher order functions. Parser combinators. Domain specific languages.
- Weeks 7-9
  Scala Libraries: Collections, Actors and concurrency, GUI programming.
- Weeks 10-12
  Advanced Topics: the Expression Problem, Parameterised Types, Module Systems

Resources for Students

There is no set text book for this course. Recommended references and online material include ...

Concurrency in Java

- Online (quick overview): Lars Vogel: Java Concurrency / Multithreading - Tutorial
- Online: Java Concurrency tutorial
- Book (highly recommended): Goetz et al: Java Concurrency in Practice
- Book: Doug Lea: Concurrent Programming in Java: Design Principles and Patterns 2nd ed

OO/Functional Programming (with Scala)

- Online list of resources Learning Scala This references most, if not all of the following ...
- A Brief Scala Tutorial (pdf)
- Scala By Example (pdf)
- An Overview of the Scala Programming Language (pdf)
- A Tour of Scala.
- Perhaps the best book: Odersky, Spoon and Venners: Programming in Scala
- Free Online Book: Wampler and Payne: Programming Scala
- Book: Pollak: Beginning Scala

Course Evaluation and Development

This course is evaluated each session using the CATEI system.

In previous offerings of this course, there was a greater focus on OO design techniques and design patterns, with a group design and implementation project, as well as individual programming assignments for concurrency in Java. Scala and other advanced OO language topics were dealt with sketchily. Some students were not happy with group work; other students felt that they did not get a chance to experience the more advanced topics (i.e. no practical experience with Scala).

Based on their comments, we have reduced the emphasis on the design component (which is now covered in COMP2911) and removed the group work project. Extra content and assignments on the advanced topics in Scala have replaced them.