Introduction

Object-Oriented Software Development COMP4001
CSE UNSW Sydney

Lecturer: John Potter
Course Aims

• Gain knowledge of modern OO software development
  – The course will emphasise programming practice.

• Focus on 2 key areas
  – Concurrency
    • Thread-based concurrency for applications software
    • Not operating systems, not database, not data parallel (high performance computing for scientific applications)
    • Language and library support for concurrency
  – OO / functional programming
    • Immutable objects as data values
    • Functions as objects
Course Aims

• Improve OO programming skills
  – More advanced Java programming (for multithreaded programming)
  – Use of high level program abstractions in Scala (for OO / functional programming)
    • Parser combinators, Domain Specific Languages, Actors

• Develop awareness of OO research directions
  – OO type systems
  – Concurrent object models
  – Techniques for extensibility. Code reuse. Multiple inheritance and mixins
  – The expression problem
Course Outline

• All course material will be accessible via WebCMS
  – Lecture slides, assignments, links to reading material and resources

• Web-site http://www.cse.unsw.edu.au/~cs4001

• For technical help with lecture material and programming assignments
  – try the Message Board
    • do NOT post your own code
  – then face to face consultations

• For admin help, email potter@cse.unsw.edu.au
Course Delivery and Assessment

• 12 x 3 hour lectures
  – Thu 3 – 6
  – 1 x 10 minute break each lecture
  – Slides + program demos

• 4 programming assignments. 60% total
  – 2 in Java, 2 in Scala
  – Due at end of weeks 4, 7, 10, 13
    • thread safety in Java 10%
    • parsers in Scala 15%
    • concurrent tasks in Java 15%
    • actors in Scala 10%
  – Assignment description will be released at least 2 weeks before due date
    • Refinement of assignment, with sample data and/or test code, will be released at least 1 week before the due date

• Final exam (2 hour) 40%
  – based on lecture and assignment material
  – students who score less than 35% for the final exam will not be permitted to pass the course
    • their scores will be capped at 45%
Course Assistance

• There are no formal tutorials for this course
  – Be prepared to ask, and answer, questions during lectures
  – The programming assignments are intended to focus your learning experience

• The lecturer will be available for face-to-face consultation for 3 hours per week
  – Times will be posted on course web-site
  – Extra help sessions may be arranged in a lab
    • Depending on demand
Lecture Outline
Concurrent OO Programming

• Focus on Concurrency in Java
• Weeks 1-4: Threads and Thread Safety
  Synchronisation and Wait Conditions
• Weeks 5-7: Java’s Concurrency Utilities
  Concurrent Collections
  Tasks, Execution Control
  Liveness, Deadlocks
  Performance and Testing
• Weeks 8-10: Concurrency Models and Concepts
  Message-based, Event-based
  Software Transactions
  Actors
• Week 11-12: Advanced Topics
Lecture Outline

**OO / Functional Programming**

- **Weeks 1-3**  
  Introduction to Scala

- **Weeks 4-6**  
  Functional Objects  
  Parser Combinators  
  Domain Specific Languages  
  Scala Libraries  
  Collections

- **Weeks 7-9**  
  Implicits  
  Actors and Concurrency  
  Swing GUI

- **Weeks 10-12**  
  Advanced Topics  
  Expression Problem, Visitor  
  Generics, Parameterized Types  
  Module Systems
Reference Books

• these are not essential, but are recommended
  – they have not been ordered for UNSW bookshop
  – you should be able to find/order online versions

• Java Concurrency in Practice. by Brian Goetz, Tim Peierls, Joshua Bloch, Joseph Bowbeer, David Holmes, and Doug Lea. Addison-Wesley.
  – See http://jcip.net/

• Concurrent Programming in Java: Design Principles and Patterns, second edition, by Doug Lea. Addison-Wesley

• Programming in Scala, second edition by Odersky, Spoon, Venners
  – for online version of first edition see http://www.artima.com/pins1ed/
  – second edition is updated for new libraries in Scala 2.8
Software

• you may use whatever development tools you like
  – BUT I cannot guarantee help with general tools
• you need to use:
  – Java and Scala development tools
    • JDK 1.6 (at least) and Scala 2.9
  – preferably with an IDE
    • I recommend you use eclipse with at least Java 1.6 and Scala 2.9
      – which version?
        » the Scala 2.9 plug-in works for eclipse 3.5 (Helios)
        » but may have problems with eclipse 3.6
      – see http://www.scala-lang.org
Course Prerequisites

• you MUST be a competent Java programmer
• you must understand basic concepts of OO
  – classes, methods and field definitions
  – objects are run-time instances of classes
  – interfaces/abstract classes
    • inheritance with method implementation/method overriding
    • polymorphism and dynamic binding
      – runtime selection of methods
        » based on the type of object the call is made on
  – object initialisation rules
    • constructors and initialisation of super objects
Course Prerequisites

• you should have understanding of basic OO design techniques
  – at least to level of COMP2911
  – modelling systems as collections of objects
  – class diagrams (UML)

• you should understand OO abstraction in Java
  – programming to an interface
  – hiding implementation details of objects
  – use of access modifiers to control visibility
    • public, (package), protected, private
Course Desirables

• ideally, you should
  – be familiar with Java collections and iterators
  – understand generic types in Java: e.g. List<E>
  – know how to write unit tests
    • and have used JUnit
  – understand design by contract
    • class invariants (constraining valid states of objects)
    • method contracts
      – preconditions (specifying when a method call is valid)
      – postconditions (specifying effect and result of a method call)
  – know some basic design patterns:
    • template method, iterator, decorator, adapter, composite
Homework for Week 1

• set up your software environment for working with Java and Scala
  – recommended:
    • eclipse3.5
    • java1.6 (or higher)
    • scala2.9 see: http://www.scala-lang.org

• be sure that you can run the week 1 exercises for Java and Scala