Advanced Issues in Object-Orientation (IOOR/ID2017)

Lecture 1
Introduction to the course

Beatrice Åkerblom  Tobias Wrigstad
beatrice@dsv.su.se  tobias@dsv.su.se

torsdag den 5 november 2009

Our Goals

- Having successfully completed the course, you should be able to
- Explain (basic and) advanced OO terms and ideas, reason about different languages' realisation of OO and what consequences these differences may have on actual programming
- Reason about and use less-main stream OO concepts like prototype-based programming,
- Learn new OOPLs and understand their underlying logic (wrt the OO realisation) in a few days

Our Goals (cont’d)

- Having successfully completed the course, you should be able to
- Read and understand code in several different OOPLs (e.g. Ruby, Python, Eiffel, Smalltalk, Scala, C++, Java)
- Write non-trivial programs in Smalltalk and use the language’s specific features to solve programming problems
- Reason about the semantics of OO/ how OO works under the hood and use this knowledge to solve programming problems and explain different implementation techniques for OOPLs

Our Goals (cont’d)

- Know at least one open problems in the field of OO research, e.g. shared-memory concurrency problems and solutions, gradual typing and pluggable type systems
- Give an account of relevant and current trends within PL research, particularly for OOPLs
- Explain the rationale for, and the foundational concepts for, aspect-oriented programming languages, plus being able to identify features suitable to model as aspects
How to Study

• Research shows that these are the worst studying techniques:
  – Postponing reading literature to just before exam
  – Postponing doing assignment work to just before exam
• What we mean by worst
  – No real understanding (harder in real-life and subsequent courses that build on this knowledge)
  – Easy to forget (harder at the re-exam)
  – You won’t pass the course

Our Pedagogic Approach

• Dividing the work up into several, small and manageable bits
• Working together with other people, discussing and reasoning
• What this means for you in practice
  – Several small assignments
  – Reading exercises to keep lectures and reading at the same pace
  – Working both in group and individually

Tutoring and Help

• There will be only one scheduled tutoring session (on Smalltalk) — not in your schedule yet
• Otherwise, request help through email
• Visit us in our offices (but email first)
• Generally, you get excellent aid — but only when you ask for it

How to Pass

• Individual tasks
  – Complete at least two well done assignments (1.5 HP)
  – Pass the take-home exam (1.5 HP)
• Group tasks
  – Do and correct reading exercises (1.5 HP)
  – Design and implement a graphical adventure game (3 HP)
Individual Assignments

- Goal: deepen your understanding of interesting topics
- There are three possible grades: 2 points, 1 point or fail. Note that there will be no possibilities to complement an individual assignment later.
- Choose from a growing selection of assignments or suggest your own
- Assignments of both practical and theoretical nature
  - Implement this using this...
  - Read this and criticise this.

Reading Exercises

- Goal: keeping your literature studies in synch with the lectures
- You will divide yourselves into reading groups of about four people
- You will read and discuss the book
- Each week, you will submit written answers to questions about the chapters you’ve read, and correct another groups’ answers
- Most students like or love this approach

Take-Home Exam

- Goal: test your understanding of the theoretical aspects—no OOA/D or programming
- Your answers should be in essay form
- Exam is handed out 2009-12-10 and is due back before 2009-12-11, 23:59
- Use the literature, follow the guidelines on the web and answer everything yourselves—no co-operation is allowed
- Sample questions can be found on the course web
- Must pass all questions to pass exam
- Let the formal requirements guide your answers

Group Assignment

- Goal: improve understanding of the theoretical concepts by practical use & have fun
- Same groups as the reading groups
- Design an implement a graphical adventure game and discuss design issues and language issues
- Languages allowed:
  - Smalltalk, (if not used before)
- Poster seminar 2009-11-11
- Possibly final seminar in the pub
Course Council

- Four persons
- Council meetings Wednesdays after the lecture
- Meetings cancelled if not needed
- The course can be changed while it runs

The Book

Article Compendium

- Handed out in six batches
- All articles listed in the course information in Daisy

Now

- Join a reading group
- Three (3) in each group
- Please mix with new people
- Names and name of contact person handed in
- Sign up for the course council
- Stretch your legs, go grab coffee
- After this, the real lecture starts
- When you return, seat yourself in your reading group!
Lecture 1
Advanced Object-Orientation
Tobias & Beatrice

What is OO?
- You should already be seated in your reading groups!
- Talk for 5 minutes about what is OO and what a programming language must have in order to be an object-oriented programming language
- Take notes, in English
- Be ready to present them

OO ≠ Java

OO ≠ C++
What is OO?

- A structuring principle for programs
- A way of viewing the world
- Does not require programming language support, but certainly benefits from it
The world is populated by objects that communicate by sending messages to each other.

An object decides how/if to react to a message.

Objects may be grouped, this grouping can be conceptual or language-supported.

### Terminology
- Stack and heap
- Pointer and reference
- Aliasing
- Diamond of death
- Interpreter
- Critical section
- Parametric polymorphism
- Overriding
- Shadowing
- Overloading
- Subclass
- Subtype
- Type
- Static typing
- Primitive data type
- State
- Aggregation

### Common Concepts
- Class
- Fields and methods
- Inheritance
- Encapsulation
- Subtype polymorphism
- Dynamic dispatch
- Recursive types
- This
- Abstraction
- State and Behaviour
- Grouping and sharing
- Autonomicity
- Flexibility
- Substitution
- Expressiveness
- Identity

### OOPLs are Commonly
- Imperative
- Garbage-Collected
- Statically typed
- Impure
- Class-based
- Compiled
- Multiply inherited
- O’Caml
- C++
- Smalltalk
- Ruby
- JavaScript
- Python
- Smalltalk

Exceptions
Some Java Examples

What is wrong with this code?

```java
void example(T foo) {
    foo.bar = foo.bar + 1;
}
```

What is wrong with this code?

```java
private T foo;
T getFoo() {
    return foo;
}
```

Resolving a Method Call

```java
void example(T foo) {
    foo.bar(27);
}
```

What can affect the way the method call is compiled/performed?
Does subclassing imply subtyping?

Research Track

- Concurrency
- Memory models
- Shared memory and thread-locality
- Gradual typing
- Pluggable type systems

Is OO important?