Object-Oriented Programming
CSCI-UA 0470-001

Instructor: Thomas Wies

Fall 2013

Lecture 1 - Introduction

Acknowledgments

This course is based on Robert Grimm's course on Object-Oriented Programming.

Object-Oriented Programming (OOP)

- “Computer programming that emphasizes the structure of data and their encapsulation with the procedures that operate upon it.” (Britannica Concise)

- “An object is a software bundle of related variables and methods. Software objects are often used to model real-world objects you find in everyday life.” (Sun’s Java Tutorial)

- “The idea behind object-oriented programming is [...] opposed to a traditional view in which a program may be seen as a collection of [...] procedures.” (Wikipedia)

Object-Oriented Programming (OOP)

- “Object-oriented programming is claimed to promote greater flexibility and maintainability in programming, and is widely popular in large-scale software engineering.” (Wikipedia)

The Goal of this Course

- Learn how to build and evolve large-scale programs using object-oriented programming
  - Design:
    How do we think in objects?
    - CRC cards, UML, and design patterns
  - Language Primitives:
    How do we express object orientation?
    - classes, interfaces, inheritance, method dispatch, generics, operator overloading, and reflection
  - Language Implementation:
    How do we realize OO primitives?
    - virtual method dispatch and automatic memory management

How Do We Achieve This Goal?

- In-class lectures and discussions
  - Lectures to introduce topics and techniques
  - Q&A sessions to deepen understanding
- Course project: A translator from Java to C++
  - Written in Java, using xtc toolkit for source-to-source transformers
  - Two versions, with second version improving on first version
  - Teams of 4-5 students

From Java to C++

- Input: Java with inheritance and virtual methods
  - But without interfaces, nested classes, enums, generics, ...
- Output: C++ without inheritance, virtual methods, templates
  - I.e., a better C with namespaces, classes, operator overloading

Two Versions

- Version 1
  - Challenge: Implement inheritance and virtual methods in translator
  - Due mid-term, with in-class presentation and written report
- Version 2
  - Challenge: Implement method overloading in translator
    - Also, integrate automatic memory management
  - Due end-of-term, again with presentation and written report

Don’t Panic

- I draw on translator for most lectures
  - We develop basic translation scheme in class, together
- We have plenty of Q&A sessions and out-of-class meetings with groups
  - You drive the discussion
- xtc provides a lot of functionality
  - Though you need to learn how to use it

Some Highlights of xtc

- Facilities for representing and processing ASTs
  - (Abstract Syntax Tree = internal representation of a program)
- Parsers, type checkers, and pretty printers for Java and C
  - Convert from source, determine types, convert to source again
- Generic tool support
  - Command line flags, file search paths, error reporting,...

But Why?

Translator from Java to C++?

- Is a real, large-scale program (and not just a toy)
  - Domain with biggest promised impact of OOP
- Exposes you to implementation of OOP primitives
  - While also integrating Java and C++
- Requires you to learn and build on existing tools
  - Common scenario in practice

Two Versions of Translator?

- Educational best practice
  - "Students can try, fail, receive feedback, and try again without impact on grade." (Ken Bains)
- Software engineering best practice
  - "Plan to throw one away; you will, anyhow." (Frederick Brooks Jr.)

Teams of Students?

- Places emphasis on collaborative learning
- Prepares you for reality in industry and academia
- Helps me keep the feedback process manageable

More Details on Course

Textbooks

- For Java, “Object-Oriented Design & Patterns”
  - 2nd edition by Cay Horstmann
- For C++, “C++ for Java Programmers”
  - 1st edition by Mark Weiss

- If you have a different book on C++, you may use that

- In the long term, you will need a good reference for C++
  - “The C++ Programming Language.”, by Bjarne Stroustrup

Tools

- Personally, I use Emacs and Unix tools
  - Powerful, flexible, and easy to automate
- Linux: you are ready to go
- Mac OS: install Apple's XCode
  - http://developer.apple.com/xcode/
- Windows: not recommended
  - Dual boot into Linux
  - Install virtual machine monitor (e.g., "VirtualBox") and run Linux

Tools (cont.)

- If you insist on an IDE, I recommend Eclipse
- Java Development Tools (JDT)
  - Visual debugger, more extensive errors/warnings than JDK
  - Known to build xtc
- C Development Tools (CDT)
  - You still need developer tools on Mac OS
- XCode on the Mac works pretty well too
- I have no experience using them, so you are pretty much on your own

Expectations

- Class is an integral part of this course
  - You really should attend
- The course home page is an important part of this course
  - Shows exact requirements for project
  - Lists reading assignments, class notes
  - Provides links to useful material

Grading

- 50% for group projects
  - Typically, same grade assigned to all members of group
  - Every group will grade all other groups; peer grades are advisory
- 25% for individual assignments
  - I will hand out a few assignments, due within a week
- 25% for final exam

A Cautionary Tale

A Cautionary Tale (cont.)

- Karl Theodor zu Guttenberg
  - Used to be secretary of defense in Germany, extremely popular
  - Forced to resign because most of his PhD thesis was plagiarized
    - 94.4% of all pages, 63.8% of all text lines
  - Some choice quotes
    - “The allegation that my thesis is plagiarized is absurd”
    - “I did not consciously or deliberately cheat”
    - “I personally wrote this dissertation”

Rules

- You must do all assignments on your own
  - Without any collaboration!
- You must do the projects as a group
  - But not with other groups
  - Without consulting previous years' students, code, etc.
- You should help other students and groups on specific technical issues
  - But you must acknowledge such interactions

How to Get Started

- Introduce yourself in a few minutes
- Subscribe to the class mailing list
  - By tonight
- Form groups and elect a speaker
  - By Friday, September 6
- Get xtc running on your laptop
  - You can verify that everything works as expected by running:

```
> make check-rats check-c check-java
```