Object-Oriented Programming
CSCI-UA 0470-001
Instructor: Thomas Wies

Fall 2017

Class 1 - Introduction
Object-oriented programming is an exceptionally bad idea which could only have originated in California.

Edsger Dijkstra
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

http://xkcd.com/292/
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?
    • 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 with vtables, static overloading resolution, and automatic memory management
How Do We Achieve This Goal?

• In-class lectures and discussions
  – lectures to introduce topics and techniques
  – in-class exercises to deepen understanding
• Individual homework assignments that give a structured introduction to tools and concepts.
• Course project: A translator from Java to C++
  – Written in Java, using the XTC toolkit for source-to-source transformers
  – Two versions, with second version improving on first version
  – Teams of 4-6 students
From Java to C++

• **Input:** Java with inheritance and virtual methods
  – But without interfaces, nested classes, enums, generics, ...

• **Output:** C++ without inheritance and virtual methods
  – 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 and automatic memory management
  – Due end-of-term, again with presentation and written report
Don't Panic

• I will try and structure your approach to the project such that you are not overwhelmed

• We will have regular meetings

• XTC provides a lot of functionality
  – Though you need to learn how to use it
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
- Allows for ‘Pair Programming’
Pair Programming

• Programming is sometimes thought of as a solitary act. It doesn’t have to be!

• Programming in pairs
  – yields more readable code
  – fewer bugs
  – is more productive (!!)
  – shares knowledge
  – is more fun
Test-driven Development

• This course is, in part, emulating real software engineering.

• Write test for small parts of your application, end-to-end tests on every additional feature is inefficient and a difficult way to debug.

• Test-driven approach using JUnit and sbt
Operational Details
Important Dates

• Class: M & W 2:00 - 3:15pm in Silv 206

• Office hour: W 4:00 - 5:00pm in 60FA 403

• Midterm Presentations: Wednesday, Nov 1

• Final Exam: Monday, Dec 13 (no midterm exam)

• Final Presentations: Monday, Dec 18
Textbooks (not strictly required)

• Rather than making you buy more books I will rely on free online resources where I can

• For Java, “Object-Oriented Design & Patterns”
  – 2nd edition by Cay Horstmann

• For C++, “C++ for Java Programmers”
  – 1st edition by Mark Weiss

• In the long term, you may want a good reference for C++
  – “The C++ Programming Language.”, by Bjarne Stroustrup
Online Resources

• **Piazza** - Online discussion and announcements
• **NYU Classes** - Grade posting
• **Github** – Homework assignments, project, and class notes and source code
• **Website**
  – Shows 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
- 20% for individual assignments
- 30% for final exam
Homework Policies

- Grading criteria for project and homework assignments will be published.

- Homework must be submitted before the announced date and time deadline for full credit.

- For every 24 hours late you lose 10%.

- Late homework will not be accepted after the late deadline. (usually a week)

- If you turn in a homework that does not compile, it will not be accepted. You can resubmit according to the above rules.
Expectations

• Course is a lot of work, but will be fun and rewarding

• Attendance is important. Not everything discussed will be captured online.

• You drive your project's development! No handholding.
Rules & Resources

• You must do all assignments on your own, without any collaboration!
• You must do the projects as a group, but not with other groups and without consulting previous years' students, code, etc.
• You should help other students and groups on specific technical issues, but you must acknowledge such interactions in code comments.
• If you need help, first stop is Piazza. If you have the question, then almost certainly someone else does.
  – If a student does not give a satisfactory answer, I will chime in.
  – If that does not solve your issue, visit me or a grader in office hours.
• Teams can make appointments with me any time.
  – We will schedule some required meetings throughout the semester.
Three Languages

• **Source Language** – Java 1.6
  – No nested classes, anonymous classes, interfaces, enums, annotations, generics, the enhanced for loop varargs, automatic boxing and unboxing, synchronization, strictfp, transient and volatile fields and no new Java 8 features
  – Assume good input

• **Target Language** – C++
  – No virtual methods, inheritance, templates (mostly) and no new C++11 features
  – Support for basic classes, exceptions, and name spaces

• **Translator language** – Java 1.8
  – The kitchen sink
Toolchain

- Linux or OS X.
  - Windows is not advised. I will give instructions and support for Ubuntu and OS X.
  - I will provide instructions on installing a VM for Ubuntu on Windows.
- IntelliJ & CLion.
  - In a project this complex, you really need good tools.
  - These IDEs are very good. While it’s not strictly mandatory, I recommend to use these as much of the project will utilize their capabilities.
  - Full versions are available for free under a student license.
- Sbt, XTC, Git, JUnit, Astyle...
  - Real software engineering tools!
  - Your first homework will be a detailed guide on installing most of these tools.
  - You will need them!!
- Homework 1 will deal with setting up the toolchain.
Challenges

• How to translate Java class hierarchies into C++ without inheritance
• How to implement Java's virtual method dispatch in C++ without virtual method dispatch
• How to select the right overloaded method (using a symbol table)
• How to automatically manage memory without an existing garbage collector (using smart pointers)
Team make-up

• 4-6 students
• one speaker
  – main contact point with me
  – ceremonial role
• key to success is to divide and conquer.
Team Selection

• At the end of class, we will take a few minutes to go around and introduce ourselves to each and chat a bit.

• You may want to look for students with complementary expertise. Java? C++? Git? etc..

• Use Piazza to "advertise" yourself to potential teammates.

• **Important**: fill out the survey that I sent out.

• I will select the teams.