Object-Oriented Programming

CSCI-UA 0470-001
Class 1
Instructor: Randy Shepherd

Purpose of OOP?

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

Course Goals

- Learn how to build and evolve large-scale programs using object-oriented programming and real software development tools.

- What are the constructs of OOP? ex. classes, interfaces, inheritance…

- How to implement software using OOP design principles.

- Understand something about language implementation. Moreover, how do we realize OO primitives? ex How is virtual method dispatch implemented?

How Do We Achieve This Goal?

- In-class lectures and discussions.
- Individual homeworks that give a structured introduction to tools and concepts.
- Course project: A translator from Java to C++

From Java to C++

- *Input*: Java with inheritance and virtual methods
- *Output*: C++ without inheritance –
  i.e., a better C with namespaces, classes, operator overloading

Two Versions

- Version 1 – Challenge: Implement inheritance and virtual methods in translator
    - Due at midterm, with in-class presentation and written report
- Version 2 – Challenge: Implement method overloading in translator and integrate 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 develop basic translation scheme in class, together.
- We will have regular meetings.
- XTC provides a lot of functionality (Though you need to learn how to use it)

Why?

- 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 you 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

Acknowledgments

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

Important Dates

- Class: M & W 2:00PM — 3:15PM in CIWW 312
- Office hours: T & R 4:45–6:15 @ WWH 425
- Final Exam: Set by department, TBA (No midterm)

Prerequisites

- Computer Systems Organization (CSCI-UA 201)

Textbooks

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

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

- Rather than making you buy more books I will rely on Blogs, YouTube and Wikipedia where I can.

Online resources

- **Mailing list** Announcements only (rare!)
- **Piazza** Online discussion
- **NYU Classes** Grade posting
- **Github** Homework, project and in-class 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 team member will submit peer reviews of their teammates.
- 20% for individual assignments
- 30% for final exam
- 5% extra credit for participation (in class and on Piazza) and ownership (showing initiative and proactivitity)
- If you have a concern about a grade on an homework or an exam and want to discuss it with me, do it in person during my office hours.

Homework Policies

- Grading criteria for project and homeworks will be published.
- Homework must be submitted before the announced date and time deadline for full credit.
- For every 24 hours late you lose 5% (so it is better to submit correct homework late than incorrect homework on time), with a max lateness of 15%.
- Late homework will not be accepted after the late deadline. (Probably 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.
    - I will try to post notes from class as quickly and completely as possible. There will be a mixture of slides and notes.
- You drive your projects development! No hand-holding.

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 in office hours.
- Teams can make appointments with me any time. We may schedule regular time.

DON’T CHEAT

(I **will** figure it out)

More Details on the Project

Three Languages

- *Source Language* - Java
    - 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 in not advised. I will give instructions and support for Ubuntu and OS X. I will provide instructions on installing a VM for Ubuntu.

- IntelliJ & CLion In a project this complex, you really need good tools. These IDEs are very good. While its not strictly mandatory, you really should use these as much of the project will utilize their capabilities.

- 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!!

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 a smart pointer)

Team make-up

- 4-5 students.
- *Speaker* - Main contact point with me
- *Architect* - “Filters” design decisions.
- 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.

- Fill out the survey I will send out.

- I will select teams.