Undergraduate Object-Oriented Programming

Robert Grimm
New York University
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)
Why Should We Care?

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

- Learn how to build and evolve large-scale programs using object-oriented programming
Ingredients (1)

- Object-oriented programming
  - Design: How to think about a program in terms of objects?
    - CRC cards, UML, unit testing
  - Primitives: How to express object orientation?
    - Classes, interfaces, inheritance, generics, ...
  - Implementation: How to realize primitives?
    - Virtual method dispatch, automatic memory management
Ingredients (2)

* Large-scale program
  * A translator from Java to C++ written in Java
  * Two versions, with second version adding select few features
  * Teams of 4-5 students
    * Incl. elected speaker who is rotated at middle of term
* Tools
  * Compilers, automatic build management, version management
  * Helper code
Ingredients (3)

* Class
  * Lecture component to introduce topics and techniques
  * Based on actual code: representations of points
  * Q & A component to deepen topics, explore alternatives, reasoning behind techniques
  * You drive the discussion
Object-oriented programming

- Design & primitives
  - Essence of object-oriented programming

- Implementation
  - Important for real, not cursory understanding
    - Also helps with (performance) debugging programs
Large-scale program

- Domain with biggest promised impact of OOP

Java-to-C++ translator

- Is real, topic of (my own and others') active research
- Is complex enough to make OOP worthwhile
  - But not too complex: focus on subsets of Java/C++
Two versions of Java-to-C++ translator

- Educational best practice
  - “Students can try, fail, receive feedback, and try again without impact on grade” (Ken Bains)

- Systems building best practice
  - “Plan to Throw One Away” (Frederick Brooks Jr.)
Justification (4)

- Teams of students
  - Emphasis on collaborative learning
  - Also reality in academia and industry
  - Manageability of feedback process
Class

- Lecture component provides gentle introduction
- Guided discussion component provides playground for exploring alternatives
- Both taken together provide skeleton for course
  - You know what you should focus on at that point
Problems with Old Version of Course

- Artificial divide between two programming languages
  - Both languages are now used at same time
  - Java during first half, C++ during second half

- Little interactivity

Course is team-based, focused on interaction
- Just lecture us

- Resistance to understanding implementation using tools
  - Project is the implementation

- Neither is truly required for managing toy examples
  - Team working on large program requires use of tools
Challenges for New Version of Course

- Complexity of technical content, project
  - xtc: Toolkit for building source-to-source translators
  - Q & A sessions, office hours
- Group-based work
  - Open communication is key: between yourselves, with me
- Be proactive: play with concepts, code, fellow students
Feedback Process (incl. Evaluation)

- Per group weekly progress report
  - What did you accomplish?
  - What did you learn, find surprising, struggle with?
  - What are your plans for next week?
- In the middle, between program versions
  - Per group project presentations
- At the end
  - Per group project presentations, individual final exam
Contract

- I provide the overall structure of the course, introducing topics and techniques, sharing my experiences, and facilitating our conversation.
- You actively participate in all aspects of the course, sharing your ideas, questions, and concerns as well as realizing a significant project.
- Together, we explore how to leverage object-oriented programming to build large-scale programs.