Introduction

Christopher Simpkins
chris.simpkins@gatech.edu

Course Overview

- Workload
- Course Content
- Syllabus

Expected Time Allotment

> *One semester credit is expected to require at least three hours of scholarly activity per week.*
>
> – `http://www.registrar.gatech.edu/faculty/fs_sch.php`

3 credit class = 9 hours a week (12 in summer)

2.5 hours of lecture (3 x 50min, or 2 x 1:45 = 3.5 hours in summer)
1.5 hour of recitation (unless you finish early)

At least 5 more hours (7 in summer) for reading, studying, and homeworks.

Your Semester Schedule

> *One semester credit is expected to require at least three hours of scholarly activity per week.*
>
> – `http://www.registrar.gatech.edu/faculty/fs_sch.php`

12 credit hours = 36 hours a week (49 hours in summer)

Full Time $\geq$ 12 credit hours (including summer)[1]

[1] `http://www.registrar.gatech.edu/students/semestersystem.php`

CS 2340

Amalgam of two courses:

- Software engineering practicum
  - Introduction to software engineering
  - Practical software engineering skills (tools, technologies, practices)
  - Preparation for design capstone (and real jobs)
- Objects and Design
  - Software design principles
  - Object-oriented design
  - Design patterns

CS 2340 bridges from academia to industry.

"Pro" Java

- The classpath
- Project directory layout
- Packages
- Jar files
- Build automation
- Using an IDE

Web Applications

- The HTTP protocol
- Clients and Servers
- Java Servlets and JSPs
- Java web application servers

Software Engineering

- Software development life cycle
- Waterfall process models
- Iterative process models
- Methods for software design, implementation, and testing

Agile Development

Agile Practices

- Pair programming
- Clean code
- Unit testing
- Simple design
- Refactoring

Agile project management (Scrum)

- Team roles
- User stories
- Small releases
- Estimation

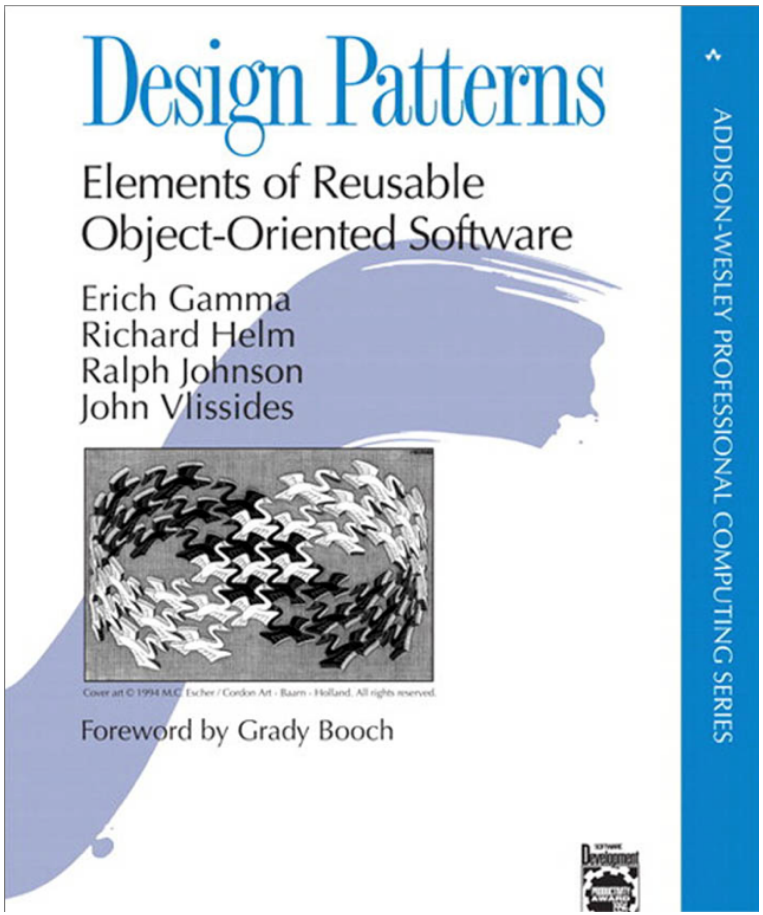
Software Design

- Design principles
- Design techniques
- System architectures
- Design documentation

Object-Oriented Design

- **S**ingle Responsibility Principle
- **O**pen Closed Principle
- **L**iskov Substitution Principle
- **I**nterface Segregation Principle
- **D**ependency Inversion Principle

Design Patterns



A recurring object-oriented design.

- Make proven techniques more accessible to developers of new systems – don't have to study other systems.
- Helps in choosing designs that make the system more reusable.
- Facilitate documenentation and communication with other developers.

Design pattern catalog: descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context.