ECE 449 – Object-Oriented Programming and Computer Simulation
Fall 2014

Instructor: Professor Jia Wang
Office: 317 Siegel Hall
Phone: 312-567-3696
E-Mail: jwang@ece.iit.edu (Please start your email subject line with [ECE449].)

Prerequisites: CS 116 and (CS 350 or ECE 242).
You are REQUIRED to have previous experiences on the following topics:
• Introductory object-oriented programming: branch and loop, function, class.
• Basic data structure and algorithm: array, linked list, searching, sorting, recursion.
• Digital logic and assembly language programming.
Experience with Verilog is a plus for course projects.

Reasonable accommodations will be made for students with documented disabilities. In order to receive accommodations, students must obtain a letter of accommodation from the Center for Disability Resources and make an appointment to speak with me as soon as possible. The Center for Disability Resources is located in the Life Sciences Building, room 218, 312-567-5744 or `disabilities@iit.edu`.

Class Time and Location: Wed.: 6:25 PM – 9:05 PM, Wishnick Hall 113

Class Home Page: `http://www.ece.iit.edu/~jwang/ece449-2014f/`

Required Textbook:
• "Accelerated C++: Practical Programming by Example"
A. Koenig and B.E. Moo, Addison-Wesley, 2000. ISBN: 978-0201703535
• Plus additional notes

Recommended Textbooks:
• "The C++ Programming Language: Special Edition"
B. Stroustrup, Addison-Wesley, 2000. ISBN: 978-0201700732
• "Design Patterns: Elements of Reusable Object-Oriented Software"
E. Gamma et al., Addison-Wesley, 1994. ISBN: 978-0201633610
• "The C++ Programming Language: 4th Edition"
B. Stroustrup, Addison-Wesley, 2013. ISBN: 978-0321563842

Course Summary: This course gives students a clear understanding of the fundamental concepts of object-oriented design/programming (OOD/OOP) and how they are supported by the standard C++ language. Students will design a complex computer simulation program using these concepts and modern software engineering practices.

Topics Covered:
• C++: core language and standard library.
• OOD/OOP: object semantics and class design, inheritance and polymorphism, design patterns.
• Computer simulation algorithms: logic simulation, event-driven simulation.
• Software engineering: Agile development.

Grading: Homeworks 5% / Final Exam: 25% / Projects: 95% (25% extra).
A: $\geq$ 90% / B: $\geq$ 80% / C: $\geq$ 60% / D (undergraduate only): $\geq$ 55%.

Homework and Project Policy: Late homeworks and projects will not be graded. Deadlines will NOT be extended, except for extraordinary reasons. Homeworks will be graded based on general approach and completion, and solutions will be released shortly after due date. Discussions on homeworks/projects are encouraged, but copying will call for disciplinary action.

Final Exam Policy: Close book, close note, cheat sheet allowed. Makeup exams will NOT be given, except for extraordinary reasons.

Lecture Schedule (tentative):

No.	Date	Topic	Chapters	HW Out	Project Due
1	8/27	Introduction	0		
2	9/3	Files and Strings	1, 2, 3		
3	9/10	Organizing Programs and Data	4	#1	1.0 (Initial)
4	9/17	Containers and Algorithms I	5, 6		1.1 (Final)
5	9/24	Containers and Algorithms II	7	#2	
6	10/1	Class Design I	9		2.0 (Initial)
7	10/8	Class Design II		#3	2.1 (Final)
8	10/15	Logic Simulation			
9	10/22	Inheritance and Polymorphism	13		3.0 (Initial)
10	10/29	Design Patterns		#4	
11	11/5	Resource Management I	10, 11		3.1 (Final)
12	11/12	Resource Management II	8, 14	#5	
13	11/19	Event-Driven Simulation			4.0 (Initial)
14	~~11/26~~	**Thanksgiving Break**			
15	12/3	Discussions and Review			4.1 (Final)
16	12/8 – 12/12	**Final Exam**			5 (Final)

Course Objectives (ABET)
After completing this course, the student should be able to do the following:

1. Identify objects and their interactions for computer simulation.
2. Utilize object lifetime for resource management considering object composition, inheritance, and exception handling.
3. Understand typical computer simulation algorithms.
4. Reuse existing class libraries to improve code quality and productivity.
5. Utilize class invariants to design class types. Document and validate pre-conditions and post-conditions via assertions.
6. Construct reusable class libraries using polymorphism.
7. Utilize design patterns when designing and reusing class libraries.
8. Design and implement a computer simulator following test-driven and iterative/incremental software engineering practices.