San José State University
Department of Computer Engineering

CMPE/SE 135
Object-Oriented Analysis and Design

Section 1
Spring 2019

Course and Contact Information

Instructor:	Ron Mak
Office Location:	ENG 250
Email:	ron.mak@sjsu.edu
Website:	http://www.cs.sjsu.edu/~mak/
Office Hours:	TuTh 3:00 - 4:00 PM
Class Days/Time:	TuTh 1:30 - 2:45 PM
Classroom:	ENG 337
Prerequisites:	For SE Majors: CS 046B For others: CMPE 126

Course Format

This course will be taught primarily via classroom presentations.

Faculty Web Page and Canvas

Course materials, syllabus, assignments, grading criteria, exams, and other information will be posted at my faculty website at http://www.cs.sjsu.edu/~mak and on the Canvas Learning Management System course login website at http://sjsu.instructure.com. You are responsible for regularly checking these websites to learn of any updates. You can find Canvas video tutorials and documentations at http://ges.sjsu.edu/canvas-students

Course Catalog Description

"Feasibility analysis and system requirements determination, object-oriented design methodology, and information systems design using object-oriented modeling techniques. Emphasis on both theoretical and practical aspects of object-oriented systems analysis and design. Team-based design project."

Course Goals

Become familiar with object-oriented analysis and program design. Employ industry-standard practices of an object-oriented approach to software development. Avoid the pitfalls of object-oriented design.

The primary goal of this course is to become a much better programmer.

The instructor will share decades of experience as a successful software developer in industry, government, and scientific research institutions. The programming examples will be in C++, but the material will apply well to other object-oriented languages such as Java.

Course Learning Outcomes (CLO)

Upon successful completion of this course, students will be able to:

CLO 1: **Requirements gathering:** Gather the requirements for a software application, distinguish between functional and nonfunctional requirements, and express the requirements in the form of use cases.

CLO 2: **Object-oriented analysis:** Derive the appropriate classes from the requirements and define their responsibilities, behaviors, interrelationships, and internal structures. Draw UML use case, class, and sequence diagrams to document and communicate the analysis results.

CLO 3: **Object-oriented design:** Apply the results of analysis to implement the classes and interfaces. Incorporate concepts such as inheritance and polymorphism, programming by contract, coding to the interface, the open-closed principle, the Liskov substitution principle, and the Law of Demeter. Write code that is easily tested and use proven testing techniques.

CLO 4: **Design patterns:** Learn the major "Gang of Four" design patterns and recognize when it is appropriate to apply them.

CLO 5: **The C++ object model:** Understand how C++ implements the object model, including the Standard Template Library (STL). Become aware of the hazards of C++.

CLO 6 **GUI programming:** Develop interactive programs that have a graphical user interface (GUI). Use callback routines with a software framework and comprehend inversion of control.

CLO 7: **Multi-threaded programming:** Learn the basics of programming multiple threads of control using semaphores, mutexes, and critical regions.

You will follow industry-standard best practices and use software development tools that are common in today's software industry.

You will develop the *critical job skill* of working in a small project team to successfully develop a software application that uses shared interfaces and data formats. Your application can then interact with applications from other teams to exchange data and results.

Academic Integrity

"Major exams in this class may be video recorded to ensure academic integrity. The recordings will only be viewed if there is an issue to be addressed. Under no circumstances will the recordings be publicly released."

Recommended Texts

This book provides optional background material. The examples are in Java.

Title:	**Object-Oriented Analysis, Design and Implementation: An Integrated Approach,** *2nd edition*
Author:	Brahma Dathan and Sarnath Ramnath
Publisher:	Springer, 2015
ISBN:	978-3319242781

Software to Install

You should install and use an interactive development environment (IDE) such as Eclipse. To write interactive programs that have a graphical user interface (GUI), you will need to download and install the wxWidgets package. This is relatively straightforward on the Mac and Linux platforms. However, the Windows platform often has significant compatibility challenges. Therefore, if you're on Windows, we highly recommend that you download and install the VirtualBox virtual machine manager, and then install and run Ubuntu (a variant of Linux) in a virtual machine.

Some useful tutorials:

- "Install and Configure VirtualBox on Windows" http://www.cs.sjsu.edu/~mak/tutorials/InstallVirtualBox.pdf
- "Install and Configure Ubuntu on a VirtualBox Virtual Machine" http://www.cs.sjsu.edu/~mak/tutorials/InstallUbuntu.pdf
- "Install and Configure Eclipse on Ubuntu for Java and C++ Development" http://www.cs.sjsu.edu/~mak/tutorials/InstallEclipse.pdf
- "Install and Configure wxWidgets on Ubuntu" http://www.cs.sjsu.edu/~mak/tutorials/InstallwxWidgets.pdf

Course Requirements and Assignments

You should have good C++ programming skills and be familiar with software development tools such as Eclipse.

You will work during the semester in small teams. Programming assignments will provide practice with OOAD techniques and will include developing a game program that uses simple machine learning. Each assignment will include rubrics for its grading criteria.

Each team will also have a semester design project to develop an application that it can demonstrate to the class. Each team will write a short report (10-15 pp.) that describes the design patterns and other OOAD techniques that it used, including a high-level architecture description with UML diagrams.

Each team will submit its assignments and project into Canvas, which will display the scoring rubrics. At the end of the semester, each team will give a presentation and demo of its design project, and students will help to score each presentation.

Each assignment and project will be worth up to 100 points, and each will include rubrics for its grading criteria. Late assignments will lose 20 points and an additional 20 points for each 24 hours after the due date.

The university's syllabus policies:

- University Syllabus Policy S16-9 at http://www.sjsu.edu/senate/docs/S16-9.pdf.
- Office of Graduate and Undergraduate Program's Syllabus Information web page at http://www.sjsu.edu/gup/syllabusinfo/

"Success in this course is based on the expectation that students will spend, for each unit of credit, a minimum of 45 hours over the length of the course (normally 3 hours per unit per week with 1 of the hours used for lecture) for instruction or preparation/studying or course related activities including but not limited to internships, labs, clinical practica. Other course structures will have equivalent workload expectations as described in the syllabus."

Exams

The midterm and final examinations will be closed book. The exams will test understanding (not memorization) of the material taught during the semester and now well each of you participated in your team assignments and project. Instant messaging, e-mails, texting, tweeting, file sharing, or any other forms of communication with anyone else during the exams will be strictly forbidden.

There can be no make-up quizzes and midterm examination unless there is a documented medical emergency. Make-up final examinations are available only under conditions dictated by University regulations.

Grading Information

Individual total scores will be computed with these weights:

35%	Assignments*
30%	Design project*
15%	Midterm exam**
20%	Final exam**

* *team scores*
** *individual scores*

Each assignment and exam will be scored (given points) but not assigned a letter grade. The average score of each assignment and exam will be available in Canvas after it has been graded.

Final course grades will be based on a curve. Per CMPE Department policy, the median total score will earn a B–. Approximately one third of the class will earn higher grades, and another one third will earn lower grades.

Postmortem Report

At the end of the semester, each student must also turn in a short (under 1 page) individual postmortem report that includes:

- A brief description of what you learned in the course.
- An assessment of your accomplishments for your team assignments and design project.
- An assessment of each of your other project team members.

Only the instructor will see these reports. How your teammates evaluate you may affect your course grade.

Classroom Protocol

It is very important for each student to attend classes and to participate. Mobile devices in silent mode, please.

University Policies

Per University Policy S16-9, university-wide policy information relevant to all courses, such as academic integrity, accommodations, etc. will be available on Office of Graduate and Undergraduate Program's Syllabus Information web page at http://www.sjsu.edu/gup/syllabusinfo/.

CMPE/SE 135
Object-Oriented Design and Analysis

Section 1
Spring 2019

Course Schedule (subject to change with fair notice)

Week	Dates	Topics
1	Jan 24	Introduction Manage change and complexity An example of iterative development *Form programming teams*
2	Jan 29 Jan 31	Encapsulation Gather functional and non-functional requirements Create use cases Identify objects, behaviors, and dependencies *The Functional Specification*
3	Feb 5 Feb 7	Key points for good design Abstract classes Designs that scale well The Boost library
4	Feb 12 Feb 14	Analysis precedes design Where do classes come from? UML class, sequence, and state chart diagrams The Principle of Coding to the Interface *The Design Specification*
5	Feb 19 Feb 21	Class design example Accessors and mutators Immutable classes The Law of Demeter and the Principle of Least Knowledge Cohesion and consistency
6	Feb 26 Feb 28	Programming by contract Preconditions, postconditions, invariants, and assertions Pre- and postconditions and inheritance The Liskov Substitution Principle Simple machine learning for the Rock-Paper-Scissors game
7	Mar 5 Mar 7	Code reuse Abstract superclasses The Principle of Favoring Delegation over Inheritance “Has a” vs. “is a” Polymorphism Virtual destructors

Week	Dates	Topics
8	Mar 12 Mar 14	***Midterm exam Tuesday, March 12*** The Model-View-Controller architecture Interactive programming with a graphical user interface (GUI) Introduction to wxWidgets Inversion of control Callback functions Events and event handlers
9	Mar 19 Mar 21	Software frameworks A GUI version of Rock-Paper-Scissors The Open-Closed Principle What are design patterns? Strategy design pattern Observer design pattern
10	Mar 26 Mar 28	Decorator design pattern Factory method design pattern Singleton design pattern Adapter design pattern Facade design pattern
	Apr 1 - 5	***Spring break***
11	Apr 9 Apr 11	Template design pattern Iterator design pattern Composite design pattern State design pattern
12	Apr 16 Apr 18	Lambda expressions Exception handling Constructor and destructor calls How does a Standard Template Library (STL) vector grow? Why did my program crash? Shallow vs. deep copy
13	Apr 23 Apr 25	Overloading vs. overriding STL iterators A "safe" array type The "Big Three" C++ template classes and functions The **`auto`** keyword The **`decltype`** pseudo-function
14	Apr 30 May 2	Pointers vs. references Raw pointers vs. unique and shared smart pointers Move semantics Multi-threaded programming
15	May 7 May 9	Project presentations
Final exam	***Friday, May 17***	Time: 12:15 - 2:30 PM Room: ENG 337