Instructor: Dr. Sam Ramanujan
Office Address: WDE2707
Office Hours: T,R 11:00 – 1:00, W 10:00-11:00 (CSC) and by appointment
Phone: 660-441 0496 [office cell]
E-mail: ramanujan@ucmo.edu

REQUIRED TEXTBOOK


Useful References


The Unified Modeling Language User Guide by Grady Booch, James Rumbaugh, and Ivar Jacobson, Addison-Wesley, 1999

Design Patterns: Elements of Reusable Object-Oriented Software by Gamma, Helm, Johnson, and Vlissides, Addison-Wesley, 1995

Object-Oriented Software Construction by Bertrand Meyer, Prentice Hall, 1997


Meta-site - http://www.cetus-links.org/

COURSE DESCRIPTION

The demand for people who understand Object Orientation (OO) has escalated immeasurably. The perceived benefits of OO, such as gains in productivity, improved quality, and faster development, and the ease of extending systems, make OO an attractive alternative to the structured approach. However, the key to realizing these benefits lies in understanding how to analyze and design systems. OO does not guarantee reusability of classes and flexibility of systems. These can be achieved only by adhering to good analysis and design principles.

This course focuses on OO analysis and design using the Unified Modeling Language (UML). Static and dynamic diagramming notations will be covered. The course also discusses distributed OO architectures (e.g., CORBA) and design patterns.
**LEARNING OBJECTIVES**

Upon successful completion of this course, the student will:
1. Be familiar with the fundamental aspects of OO analysis and design using the **UML**
2. Have an understanding of advanced concepts such as design patterns and frameworks
3. Have the necessary knowledge to transform an analysis model into an OO design that can eventually be implemented in a language such as Java

**COURSE OUTCOMES**

1. Use the “Use Cases” methodology and produce an analysis model
2. Use UML concepts such as class diagrams, sequence diagrams, state transition charts and package diagrams to create a logical model of a system.
3. Use advanced OO techniques such as patterns and frameworks to make the design robust.
4. Write code in appropriate OO language to implement the designed system.

**ACADEMIC INTEGRITY**

The student should refrain from cheating, plagiarizing, submitting work previously used (or currently being used for another class), or doing anything that is deemed to be dishonest. Please refer to your catalog for regulations governing academic dishonesty.

**GRADING**

Exam 1    25 % (Outcome 1 2/14)
Exam 2    25 % (Outcome 2 3/14)
Final Exam 25 % (Outcome 3 TBD)
Project 20 % (Outcome 1-4 ) (TBD)
          (Project description 2/2, use cases presentation 2/25, class diagrams due 3/12 – code 20% of the grade, Project presentation immediately after final exam)
Homework and Research 5 % (Outcome 1-4) (5/5) (1-2 pages on any pattern that could be used in your project … this is an individual assignment)

**ADDITIONAL NOTES**

1. You are required to attend classes regularly. If you miss a class, it is your responsibility to obtain notes from a fellow student.
2. Late submissions of projects will not be accepted.
3. Make-up exams will not be given.
4. The student is expected to participate in class discussions.
5. Details of the projects will be given in class. *Students are responsible for research (based on the software engineering course) activity that lends to successful completion of the project.*