Syllabus and Course Description

SIE 508 Object Oriented Programming

Course Description

This course introduces advanced programming skills and focuses on the core concepts of object-oriented programming and design using a high-level language, either Python or Java. Object-oriented programming represents the integration of software components into a large-scale software architecture. Software development in this way represents the next logical step after learning coding fundamentals, allowing for the creation of sprawling programs. The course focuses on the understanding and practical mastery of object-oriented concepts such as classes, objects, data abstraction, methods, method overloading, inheritance and polymorphism. Practical applications in the domain of data science and as seen in stacks, queues, lists, and trees will be examined.

Credits: 3

Prerequisite: Graduate standing, SIE507, or programming experience in Python, or permission of the instructor

URL for Course: forthcoming

Faculty Information

Dr. Silvia Nittel (regular semester instructor: Python)
Associate Professor of Spatial Informatics
School of Computing and Information Science
334 Boardman Hall
University of Maine
silvia.nittel@maine.edu

Christopher Dufour (regular summer session instructor: Java)
Lecturer of Computer Science
School of Computing and Information Science
238 Boardman Hall
University of Maine
christopher.dufour@maine.edu

Office Hours:

Office hours for this course are announced at the beginning of each session. Alternatively, contact the instructor.

Instructional Materials:

1. Computers:

Each student is required to have access to a laptop, which will be used in class for hands-on exercises during class (any platform is ok, Windows, Mac OS or Linux but not Chrome book)

2. Textbooks:

Instructor Nittel:

Mastering Object-Oriented Python: Build powerful applications with reusable code using OOP design patterns and Python 3.7, 2nd Edition, Steven F. Lott (June 14, 2019), **ISBN-10:** 1789531365

Instructor Dufour:

Java Foundations: Introduction to Program Design & Data Structures, 4th edition
By John Lewis, Peter DePasquale, and Joseph Chase

3. Software:

Instructor Nittel:

Latest version of the Python Programming language and an Integrated Development Environment (IDE) such as PyCharm.

Instructor Dufour:

Latest version of both Eclipse IDE and Java JDK (currently version 12)

Course Goals:

- Introduce the principles of object-oriented programming in a higher-level programming language, such as Python or Java
- Analyze a problem statement to develop a mental model of objects necessary to create a software architecture
- Utilize object-oriented programming to frame software architectures, with care towards separation of concerns and abstraction
- Gain skills in designing, and programming software for reuse of code.
- Establish development methods in object-oriented programming to qualify students for teaching the language in other settings

Student Learning Outcomes:

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

- Develop understanding of writing object-oriented programs that combine functions and data.
- Analyze a problem statement to develop a mental model of objects necessary to create a software architecture
- Combine previously written code into larger programs
- Translate abstract concepts into Class's in software
- Apply the object-oriented programming language to develop software, including programs utilizing multiple Class's
- Instruct others in the use of the object-oriented programming language

Course Outline A

(Regular Semester Course Offering using Python, Instructor: Silvia Nittel)

1. Course Schedule

This course consists of regularly offered on-campus lectures and student-engaged code writing sessions with on-campus and online live graduate students. These sessions are also recorded for viewing by distance students that are unable to attend live virtually.

Evening hour virtual office hours are made available such that all distance students are able to attend at a minimum of one hour per week and on-campus students are as well invited to attend these sessions.

Week 1	1. Intro and Overview 2. Principle of Software Engineering and Reusing and Extending Code
Week 2	3. Review of Fundamentals of Procedural Programming
Week 3	4. Objects
Week 4	5. Data Abstraction 6. Information Hiding & Encapsulation
Week 5	7. Constructors, destructors, and object creation 8. Name space and references
Week 6	9. Class Methods 10. Methods Overloading
Week 7	11. Inheritance 12. Inheritance
Week 8	13. Polymorphism 14. Polymorphism
Week 9	15. Abstract Classes 16. Abstract Methods
Week 10	17. Exceptions 18. Exception Handling
Week 11	19. Templates
Week 12	20. Practical Example: Data Science Classes
Week 13	21. Practical Example: Data Science Classes
Week 14	22. Example translations of concepts in Python to Java 23. Practical Example: Data Science Classes
Week 15	24. Student final project presentations

2. Grading and Course Expectations
As a graduate level course, you are expected to exhibit high quality work that demonstrates sound understanding of the concepts and their complexity. Earning an “A” represents oral and written work that is of exceptionally high quality and demonstrates superb understanding of the course material. A “B” grade represents oral and written work that is of good quality and demonstrates a sound understanding of course material. A “C” grade represents a minimally adequate completion of assignments and participation demonstrating a limited understanding of course material. This class has no exams; only homework and/or project assignments. Active live class participation (virtual or on-campus) is expected and may take the form of active participation in the live class sessions or regular participation in virtual office hours with the instructor with at least one time each week meeting the schedule needs of all students.

Grading criteria:

Homework assignments (programming, presentations) – 70%
Active participation – 10%
Final Project (programming project and presentation) – 30%

Policies:

Students are expected to attend class sessions (in person or virtual) or office hours (virtual).
Late assignments will result in 10% deduction in grade.

<u>Course Outline B</u>

(Summer Six-Week Course Offering using Java, Instructor: Chris Dufour)

This course consists of regularly offered on-campus lectures and student-engaged code writing sessions with on-campus and online live graduate students. These sessions are also recorded for viewing by distance students that are unable to attend live virtually. Students are expected to have reviewed assigned reading material prior to attending class live or prior to viewing each video lecture.

Evening hour virtual office hours are made available through Zoom such that all distance students are able to attend at a minimum of one hour per week and on-campus students are as well invited to attend these sessions.

In addition to regular homework delivered through Blackboard, two projects will be due at the midpoint and end of the term.

1. Course Schedule

Week 1	Java Introduction and Fundamentals	Chapters 2 and 4
Week 2	Object-oriented Programming Introduction	Chapter 3
Week 3	Abstraction and Encapsulation	Chapter 5
Week 4	Inheritance and Polymorphism	Chapter 8 and 9
Week 5	Data Structures Introduction	Sections 7.1, 7.2, 7.3, 7.6, 12.1, 12.2, 12.3, 12.6
Week 6	Finish Data Structures and Hash Tables	Sections 13.1, 13.2, 13.3, 13.4, 15.1, 15.2, 15.3, 15.5

2. Grading and Course Expectations

Grading:

Grades are primarily based upon performance on homework (40%), two projects (40%), and their proposals (20%).

Project 1 Proposal Date:	TBA
Project 1 Due Date:	TBA
Project 2 Proposal Date:	TBA
Project 2 Due Date:	TBA

Homework:
There will be several assignments in this course that will typically be due on TBA. Each may include conceptual questions, which will require a written answer. There will also be a coding component, generally involving the creation of a small program demonstrating our most recently covered topic. Depending on the breadth of the assignment, there will be either one or two weeks to complete them.

Once during the course, you may submit an assignment late without any penalty. The only requirement for this is that you state before the due date that you need this time (through Blackboard). In this case, you will receive three extra days to complete your assignment, with it due on TBA. *There are no other late assignments allowed during the course semester. If necessary you must submit what you have!*

Projects:
As part of the course you will develop two programming projects. Each project may be of your own design, though you will be required to write a proposal and get your idea approved. Your projects must demonstrate the breadth of programming knowledge you gain during the course. You are encouraged to pick a problem related to your interests or work that you would like to attempt to solve, regardless of how difficult it appears. You will be graded only on what you submit, not by how closely you accomplish your proposal.

Collaboration Policy (applies to both the regular semester and summer offerings)

Because it is important for students to be able to develop and demonstrate the ability to create programs, this course has very strict rules on the degree of collaboration allowed on homework and projects. Because it is important for software developers to be able to work both independently and in teams, we will allow for collaboration on some assignments. Whenever a student collaborates in any way with anyone, he/she must list the names and email addresses of those they collaborated with in the header to their assignment.

Actions that are always allowed:

- Getting help from the instructor
- Sharing or copying notes from lecture
- Discussing general assignment design with classmates (list their names in your header comment)
- Comparing the results of your program with that of others
- Discussing how to test your code with classmates
- Forming study groups to discuss course concepts (other than assignments)
- Working together on practice problems
- Talking about solutions to an assignment after the due date has passed and all involved students have already turned in their assignment
- Open discussion of issues during live virtual class sessions and virtual office hours with the instructor (no need to list others present)

Actions that are never allowed:

- Copying someone else's code
- Writing code for someone else
- Looking at someone's code without their permission

- Looking for or copying the solution to an assignment from the Internet
- Collaborating with someone who is not a current SIE 508 student
- Buying a solution to an assignment
- Collaborating in an allowable manner but not listing collaborator names in your header comment

Standard Syllabus Notices

- Important Disability Notice
- Academic Honesty Notice
- Nondiscrimination Notice
- UMaine Student Code of Conduct
- Classroom Civility
- Sexual Discrimination Reporting
- Course Schedule Disclaimer
- Contingency Plans in the Event of an Epidemic