ECE 448/528
Application Software Design

Lecture 1. Introduction
Spring 2025

Won-Jae Yi, Ph.D.

Department of Electrical and Computer Engineering
Illinois Institute of Technology

Course Syllabus

Syllabus

Instructor	Dr. Won-Jae Yi (wyi3@iit.edu) Office Hours: T/TR 2 PM to 3 PM; or appointment in advance Office Location: SH 316 or Zoom with an appointment in advance
Teaching Assistant	TBD Office Hours: TBD Office Location: TBD
Class Time & Location	Wishnick Hall Auditorium (WH 113) Tuesdays & Thursdays 3:15 PM – 4:30 PM Live Zoom Session Link Available on Canvas No class: March 18 & March 20 (Spring Break)
Prerequisites	Computer programming. If you haven't been writing programs for a while, please refer to the following book for introductory Java programming: "Head First Java", 2nd Edition, K. Sierra et al., O'Reilly Media, 2005
Course Description	This course provides an introduction to languages and environments for application software development utilizing Software as a Service (SaaS) for electrical and computer engineers. Students will develop a data-rich web application with a server back-end that connects mobile devices and Internet of Things using Agile software engineering practices.

Syllabus

Class Website	Illinois Tech Canvas (Lecture slides will be uploaded with password protections)
Textbooks	• "Java Network Programming", 4th Edition, E.R. Harold, O'Reilly Media, 2013 • "Core Java Volume I-Fundamentals", 10th or later, C.S. Horstmann, Prentice Hall • "Core Java Volume II-Advanced Features", 10th or later, C.S. Horstmann, Prentice Hall • "Head First Design Patterns", E. Freeman et al., O'Reilly Media, 2004
Recommended Computer Specification	A course VM (VMWare) image will be provided to everyone. The course VM is recommended as all project environments are configured in advance. For those who have less than 16 GB RAM and a quad-core CPU, a standalone setup guide will be provided separately.

Syllabus

Topics Covered	• Java Ecosystem • Client-server architectures and RESTful services • Pub/Sub middleware • Web user interface design • Security; database; data visualization • Software engineering and Agile development	
Grading	ECE 448 Homework Assignments: 10% Projects: 110% (20% extra) 6 mandatory + 1 extra projects	ECE 528 Homework Assignments: 10% Projects: 95% (5% extra) 7 mandatory projects
Homework and Project Policy	Late homework submissions will not be accepted or graded. Homework assignments will be graded based on general approach and completion. Working together on homework assignments and projects is encouraged, but copying assignments will call for disciplinary action.	

Syllabus

- **4 Homework assignments**
  - Contribute to 10% of your final grade
  - Submit online

- **7 Projects**
  - ECE 448: 15% * 6 + 20% * 1 = 110% (including 20% extra)
  - ECE 528: 10% * 2 + 15% * 5 = 95% (including 5% extra)
  - Project Report: submit online via Canvas
  - Source Code: via Endeavour Git Repo only
    - Endeavour Git access will be provided soon.

- **Late submissions will not be graded**

Project Workload

- This is a project-based course, and you are required to design and implement the software system
  - Projects will be discussed in lectures, but step-by-step instructions will not be provided
  - Expect to practice your problem solving and troubleshooting skills, as you have gained in a programming/design course like CS 115/116/201 and ECE 218/242
- **DO NOT expect to finish a project during the weekend right before the deadline**
- Project 1 & 2: Introducing you to Java ecosystem, thus the coding part should not require a lot of effort
- Project 3 to 7: May need to spend 10+ hours/week in addition to watch lectures depending on your background

Late and Resubmission/Regrading Policy

- Software developers should learn how to manage deadlines, especially with all the available tools
- Late homework assignments and projects will NOT be graded, unless
  - A request to extend the deadline is **received by email 48 hours BEFORE the deadline**
  - Extension request made within 48 hours of the deadline must be accompanied with **extraordinary reasons with documented proof (e.g., doctor's notes) or it will be rejected.**
- Extraordinary reasons do NOT include
  - Code loss due to software, hardware, networking failures
  - Forgot account password
  - **Please push your code to Git repository frequently (preferably daily)**
- Resubmission/regrading of projects are not allowed
  - Project grading program will be provided to **check and correct any errors BEFORE the deadline**

Project Setup

- A desktop or a laptop capable of running Virtual Machine is recommended for this course
  - SSD equipped device
  - A minimum of 16 GB RAM
  - A minimum of quad-core processor
- For those who have laptop/PC with less powerful specifications..
  - A separate standalone setup procedure will be given (but at your own risk..)
- Course Virtual Machine
  - A full-featured Linux installation with all the necessary tools
- A VMWare VM image will be shared with you soon to set up VM on your computer
  - Link will be provided via Canvas for you to download
- Git repository
  - *I will send you a separate email soon*

How to ~~survive~~ succeed in the course?

- Read: all instructions are written
  - Tutorials, source code, documents, and **don't overlook command outputs**
- Plan ahead: manage your time wisely!
- **Do not expect anyone else but yourself to fully debug your code**
  - https://stackoverflow.com/help/how-to-ask
- Learn to use AI assistants
  - https://www.deeplearning.ai/short-courses/chatgpt-prompt-engineering-for-developers/
- Tools
  - Commit and push your code to Git server in case your computer fails
  - Use the grading program (provided along with initial project codes) to know your project grade and correct any error before the deadline
- Feel free to explore new computer hardware and software but make sure they do not interfere with your schedule to meet deadlines

How to ~~survive~~ succeed in the course?

- https://leetcode.com
  - Practice your programming skills.
- https://www.youtube.com
  - Find tutorials.
- http://stackoverflow.com
  - Learn how to communicate with professionals.

Academic Honesty

- Projects and homework assignments must be done individually
- Discussions are encouraged
- Source code from the lectures and instructions in this course can be used directly
- Source code from other places not directly related to this course can be used with proper references
- All writings including figures, tables, and screenshots must be **by yourself**

Review Illinois Tech Academic Honesty Guidelines

https://www.iit.edu/academic-affairs/academic-honesty-guidelines

Academic Honesty

- **Code must be your own creation!**
- **Typing someone else's code into your project is plagiarism**
- **Understood someone else's code and implemented their ideas into your project is also plagiarism**
- **Interactions from AI assistants (prompts, answers, etc.) should NEVER be shared with anyone else.**
- **NEVER POST YOUR PROJECT CODE OR ASK FOR HELP DIRECTLY ONLINE!!**

Review Illinois Tech Academic Honesty Guidelines

https://www.iit.edu/academic-affairs/academic-honesty-guidelines

Syllabus

Academic Honesty	It is your responsibility to be familiar with Illinois Tech Code of Academic Honesty: https://web.iit.edu/student-affairs/handbook/fine-print/code-academic-honesty Working together on the assignments is encouraged **but copying assignments will call for disciplinary action**. **All submissions including program source codes and homework assignment solutions must be 100% your work, your writing, and your code. (Typing someone else's work into your code is also considered plagiarism).** If the above policy and/or any part of the Illinois Tech Code of Academic Honesty is violated in any similarity within the homework solutions, project programming codes, comments, customized program behavior, any writings and/or figures are found, both the helper (source of work submission) and the requestor (duplicated/modified work submission) will be called for academic disciplinary action (zero score of the submission/exam, degrading course letter grade by one, and/or failing grade E for the course). Furthermore, if the above policy and/or any part of the Illinois Tech Code of Academic Honesty is violated in any similarity within the exam paper submissions, both the helper (source of work submission) and the requestor (duplicated/modified work submission) will be notified to the designated dean of academic discipline (DDAD).

Syllabus

ADA Statement	Reasonable accommodations according to American Disability Act (ADA) 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 them as soon as possible. The Center for Disability Resources (CDR) is located at 3424 S. State St. Suite 1C3-2, (312) 567-5744 or disabilities@iit.edu

Lecture Schedule (tentative)

Date	Topic	HW Out	Project Due
Week 1	Introduction		
Week 2	Software Engineering and Java		
Week 3	TCP/IP Networking	HW #1	
Week 4	TCP Server Design		
Week 5	HTTP	HW #2	Project 1
Week 6	Observer and Pub/Sub		
Week 7	MQTT	HW #3	Project 2
Week 8	Web Application		
Week 9	RESTful Service		Project 3
Spring Break			
Week 10	JavaScript and DOM		
Week 11	Model-View-Controller (MVC)	HW #4	Project 4
Week 12	Web UI Design		
Week 13	Data Visualization		Project 5
Week 14	Security	HW #5	
Week 15	Database Integration		Project 6
Week 16	**No Final Exam**		Project 7

Course Objectives (ABET) – ECE 448

After completing this course, the student should be able to do the following:

1. Understand application software architectures and application software development processes
2. Utilize event-driven programming to support networking and graphical user interface in application software
3. Design and implement testable class types. Document and validate functionality via unit testing
4. Reuse existing class libraries to improve code quality and productivity
5. Construct reusable class libraries using polymorphism
6. Utilize design patterns when designing and reusing class libraries
7. Be familiar with advanced topics including security, database, and data visualization
8. Design and implement a networked application software with graphical user interface following test-driven and iterative/incremental software engineering practices

Course Objectives – ECE 528

After completing this course, the student should be able to do the following:

1. Understand application software architectures and application software development processes
2. Utilize event-driven programming to support networking and graphical user interface in application software
3. Design and implement testable class types. Document and validate functionality via unit testing
4. Reuse existing class libraries to improve code quality and productivity
5. Construct reusable class libraries using polymorphism
6. Utilize design patterns when designing and reusing class libraries
7. Be familiar with advanced topics including security, database, and data visualization
8. Design and implement a networked application software with graphical user interface following test-driven and iterative/incremental software engineering practices
9. Develop the skills to adapt existing code to meet the newer software version requirements and standards

Software Development

Opportunities

- Our modern society depends increasingly more on computer systems.
  - e.g., Internet of Things/Big Data/Machine Learning
- As engineers, particularly as Electrical Engineer, Computer Engineer,
  - We develop software by ourselves.
  - We work closely with software developers.
- A lot of career and business opportunities with a good understanding of how software packages are developed.
- Generative AI tools? Can you trust them? How do you know if ChatGPT is telling you the correct answer?

Economics of Software Development

- Alternative to SW development: HW development (FPGA, ASIC)
- Trade-offs between HW and SW development
  - Time-to-market and non-recurring engineering (NRE) cost
  - Functionality: more features? less bugs?
  - Performance: optimize for now? or leave room for future?
- Software development is more flexible than hardware development
  - For many software systems, performance is not a concern, especially in the very beginning stage of development.
  - Software systems can be updated in field, making it possible to improve both functionality and performance later.
  - Shorter time-to-market and much less NRE cost
- Software innovations are less risky than hardware innovations.
  - You can have a demo much sooner to attract clients, collaborators, and investors.

Challenges

- Software systems are complicated.
  - Consumer requirements may change rapidly and suddenly.
  - Complex business logic.
  - Different platforms for deployment.
  - Need room for future improvement.
- Software development involves many collaborators over a long period of time.
  - People are directly involved in the project, or
  - Contribute indirectly via OS and libraries.
  - Very risky!
- It is necessary to have project management mechanisms that
  - Facilitates reasoning of complex systems, e.g. decomposing them into manageable pieces.
  - Facilitates collaboration between developers and management of development progress to mitigate risks.

Elements of Software Development

- Programming paradigms
  - Define abstraction levels to help developers effectively analyze and understand complex computing systems.
  - Various trade-offs lead to different language designs.
- Software engineering
  - Define methodologies that developers can follow to reduce risk.
  - Effective methodologies differ across software projects and evolve with changes in the workforce.
  - Will be discussed in detail in the next lecture.

Programming Paradigm I: Imperative Programming

- Imperative: computation as state and state transitions.
- Depends on the abstraction level that state and state transitions are defined:
  - Synchronous sequential logic: flip-flops, combinational gates
  - Assembly language: registers, opcodes, etc.
  - Procedural and structured programming: variables, control structures and functions.
  - OOP: objects, member functions, etc.
  - Event-driven programming: actors, messages.
- A popular and easy-approach choice
  - Intuitive for most developers.
  - Excellent performance achieved through a close alignment with hardware implementations.

Programming Paradigm I: Functional Programming

- Functional: computation as functions and their compositions
  - A function, as defined in mathematics, represents a relationship between inputs and outputs, where the function itself operates as a "black box.“
- Effective to specify
  - Computations that have well-defined mathematical counterparts (basic data structures and algorithms)
  - Computations that are easily specified as input/output relations.
- As old as imperative programming and is gaining momentum
  - Performance is no longer the major concern. Why?
  - Correctness has grown increasingly important, where connections to mathematics can provide valuable support.
- Most modern imperative programming languages support certain level of functional programming.

Static vs. Dynamic Typing (Data Types)

- Each variable has a type defining what operations are legal and their meanings.
  - This type system is used to guard against misunderstandings of system components and against careless typos.
- Static typing: the type cannot be changed once defined (C++/Java)
  - Type information is available at compile time
    - Makes it easier for others to understand the program and for compilers to identify type violations, regardless of the program's size.
    - Choosing and specifying the correct data type during programming can sometimes be challenging.
- Dynamic typing: the type can be changed (Python/JavaScript)
  - Allows to write code without knowing much about/being aware of the type system: easier for beginners or for small programs.
  - Type violations can only be identified at runtime, requiring extensive testing to ensure program quality, particularly in large programs.

Resource/Memory Management

- Resource management is essential for any program, especially memory management, and has a great impact on performance.
- **Manual management**: developers are responsible to allocate and free memory pieces correctly.
  - Typical when performance was the major concern, e.g. in C.
  - However, it is very challenging for the majority of the programmers as software systems become very complicated.
  - Too many bugs related to memory management simply lead to failure of software projects.
- **Automatic management**
  - Garbage collection (GC): fully automatic, usually for memory management only, trade-off certain amount of performance, almost all languages designed since ’90s have this feature.
  - Exception-safe resource management: deterministic, mostly from C++, partially adopted by major languages.

Java

- Features
  - Simplified from C++, similar syntax.
  - Use static typing so compiler helps to detect errors.
  - Use garbage collection for memory management.
  - “Write once, run anywhere”
  - Very good backward compatibility.
- Brief history
  - 1990s: created as an alternative of C++ for embedded system and later web applets. Allow to train a large body of workforce quickly to meet the demand of software development.
  - 2000s: enterprise applications involving complex team structures, business logics, and IT environments. Many tools and libraries were created and many developers got familiar with them.
  - 2010s: as part of the industrial effort toward open-source ecosystems. Support almost every aspects of software development via tools and libraries.

Java

- Java is neither simple nor perfect.
- Complex features were introduced/added back over the years to address the complexity of software itself.
  - e.g., lambda expressions.
- Many features/libraries were outdated/deprecated.
  - But they remain there for good backward compatibility.
  - e.g., `java.util.Date`.

TIOBE Index

Monthly Programming Language Market Share (2018/5, TIOBE Index)				
1. Java	2. C	3. C++	4. Python	5. C#
16.380 %	**14.000 %**	**7.668 %**	**5.192 %**	**4.402 %**
6. VB.NET	7. PHP	8. JavaScript	9. SQL	10. Ruby
4.124 %	**3.321 %**	**2.923 %**	**1.987 %**	**1.182 %**

Monthly Programming Language Market Share (2019/2, TIOBE Index)				
1. Java	2. C	3. Python	4. C++	5. VB.NET
15.876 %	**12.424 %**	**7.574 %**	**7.444 %**	**7.095 %**
6. JavaScript	7. C#	8. PHP	9. SQL	10. Objective-C
2.848 %	**2.846 %**	**2.271 %**	**1.900 %**	**1.447 %**

TIOBE Index

Monthly Programming Language Market Share (2019/2, TIOBE Index)				
1. Java	2. C	3. Python	4. C++	5. VB.NET
15.876 %	**12.424 %**	**7.574 %**	**7.444 %**	**7.095 %**
6. JavaScript	7. C#	8. PHP	9. SQL	10. Objective-C
2.848 %	**2.846 %**	**2.271 %**	**1.900 %**	**1.447 %**

Monthly Programming Language Market Share (2020/2, TIOBE Index)				
1. Java	2. C	3. Python	4. C++	5. C#
17.358 %	**16.766 %**	**9.345%**	**6.164%**	**5.927 %**
6. VB.NET	7. JavaScript	8. PHP	9. SQL	10. Swift
5.862 %	**2.060 %**	**2.018 %**	**1.526 %**	**1.460 %**

TIOBE Index

Monthly Programming Language Market Share (2020/2, TIOBE Index)				
1. Java	2. C	3. Python	4. C++	5. C#
17.358 %	**16.766 %**	**9.345%**	**6.164%**	**5.927 %**
6. VB.NET	7. JavaScript	8. PHP	9. SQL	10. Swift
5.862 %	**2.060 %**	**2.018 %**	**1.526 %**	**1.460 %**

Monthly Programming Language Market Share (2021/2, TIOBE Index)				
1. C	2. Java	3. Python	4. C++	5. C#
16.34 %	**11.29 %**	**10.86%**	**6.88%**	**4.44 %**
6. VB.NET	7. JavaScript	8. PHP	9. SQL	10. Assembly
4.33 %	**2.27 %**	**1.75 %**	**1.72 %**	**1.65 %**

TIOBE Index

Monthly Programming Language Market Share (2021/2, TIOBE Index)				
1. C	2. Java	3. Python	4. C++	5. C#
16.34 %	**11.29 %**	**10.86%**	**6.88%**	**4.44 %**
6. VB.NET	7. JavaScript	8. PHP	9. SQL	10. Assembly
4.33 %	**2.27 %**	**1.75 %**	**1.72 %**	**1.65 %**

Monthly Programming Language Market Share (2022/2, TIOBE Index)				
1. Python	2. C	3. Java	4. C++	5. C#
15.33 %	**14.08 %**	**12.13%**	**8.01%**	**5.37 %**
6. VB.NET	7. JavaScript	8. PHP	9. Assembly	10. SQL
5.23 %	**1.83 %**	**1.79 %**	**1.60 %**	**1.55 %**

TIOBE Index

Monthly Programming Language Market Share (2022/2, TIOBE Index)				
1. Python	2. C	3. Java	4. C++	5. C#
15.33 %	**14.08 %**	**12.13%**	**8.01%**	**5.37 %**
6. VB.NET	7. JavaScript	8. PHP	9. Assembly	10. SQL
5.23 %	**1.83 %**	**1.79 %**	**1.60 %**	**1.55 %**

Monthly Programming Language Market Share (2023/1, TIOBE Index)				
1. Python	2. C	3. C++	4. Java	5. C#
16.36 %	**16.26 %**	**12.91%**	**12.21%**	**5.73 %**
6. VB.NET	7. JavaScript	8. SQL	9. Assembly	10. PHP
4.64 %	**2.87 %**	**2.50 %**	**1.60 %**	**1.39 %**

TIOBE Index

Monthly Programming Language Market Share (2023/1, TIOBE Index)				
1. Python	2. C	3. C++	4. Java	5. C#
16.36 %	**16.26 %**	**12.91%**	**12.21%**	**5.73 %**
6. VB.NET	7. JavaScript	8. SQL	9. Assembly	10. PHP
4.64 %	**2.87 %**	**2.50 %**	**1.60 %**	**1.39 %**

Monthly Programming Language Market Share (2023/12, TIOBE Index)				
1. Python	2. C	3. C++	4. Java	5. C#
13.86 %	**11.44 %**	**10.01 %**	**7.99 %**	**7.30 %**
6. JavaScript	7. PHP	8. Visual Basic	9. SQL	10. Assembly
2.90 %	**2.01 %**	**1.82 %**	**1.61 %**	**1.11 %**

TIOBE Index

Monthly Programming Language Market Share (2023/12, TIOBE Index)				
1. Python	2. C	3. C++	4. Java	5. C#
13.86 %	**11.44 %**	**10.01 %**	**7.99 %**	**7.30 %**
6. JavaScript	7. PHP	8. Visual Basic	9. SQL	10. Assembly
2.90 %	**2.01 %**	**1.82 %**	**1.61 %**	**1.11 %**

Monthly Programming Language Market Share (2024/12, TIOBE Index)				
1. Python	2. C++	3. Java	4. C	5. C#
23.84 %	**10.82 %**	**9.72 %**	**9.10 %**	**4.87 %**
6. JavaScript	7. Go	8. SQL	9. Visual Basic	10. Fortran
4.61 %	**2.17 %**	**1.99 %**	**1.96 %**	**1.79 %**