1. *Course number and name*
   CSC 413: Software Development

2. *Credits and contact hours*
   3 credits
   Contact hours: 150 minutes of lecture sessions /week

3. *Instructor's or course coordinator's name*
   Course coordinator: Ilmi Yoon, Professor of Computer Science

4. *Text book, title, author, and year*

   Understanding Object-Oriented Programming with Java, Budd, T., Addison- Wesley, 2000

   Core Java(TM) 2, Volume I--Fundamentals, Horstmann, C.S. and Cornell, G. Prentice-Hall

   other supplemental materials
   Lecture Slides

5. *Specific course information*
   a. *brief description of the content of the course (catalog description)*

      Design and development of modern software applications. Object-oriented techniques: encapsulation, inheritance, and polymorphism as mechanisms for data design and problem solution. Software design, debugging, testing, and UI design. Software maintenance. Software development tools.

   b. *prerequisites or co-requisites*

      CSC 340 and CSC 412 with grades of C or better.

   c. *indicate whether a required, elective, or selected elective course in the program*
      Required for Computer Engineering.

6. *Specific goals for the course*
   a. *specific outcomes of instruction, ex. The student will be able to explain the significance of current research about a particular topic.*

      At the end of this course students will

- Be able to write Java programs utilizing an integrated development environment
- Utilize a debugger when doing software development
- Apply object oriented programming principles effectively when developing small to medium sized projects
- Write robust code utilizing exception handling language features
- Use a code profiler to tune a program's performance

b. *explicitly indicate which of the student outcomes listed in Criterion 3 or any other outcomes are addressed by the course.*
   Course addresses ABET Student Outcome(s): a, b, c, e, j, k.

7. *Brief list of topics to be covered*

- **Introduction to Software Development**
- **Introduction to Object-Oriented Programming - OOP**
  Information Hiding, Class Hierarchy
- **The Java Language**
- **Object Oriented Design**
  Plan for Change, Software Components, Interfaces vs. Implementation Naming
- **A Comparision of Java and C++**
- **A Compiler**
  Extended Example, Source, Tokens, AST, Decorated AST's, Code generation, Bytecodes
- **Lexical Analysis**
  Parsing - Syntax Analysis of the Token Stream Yielding the AST
  Grammar for X, ASTS Built from Source Programs
- **Tree Visitors**
- **Inheritance**
  Subclass, Subtypes and Substitutability, Forms of Inheritance, Modifiers Benefits of Inheritance, Cost of Inheritance
- **The Interpreter**
  Frames (Activation Records)
  Javadoc Documentation of Selected Interpreter Classes The Runtime Stack, The Virtual Machine
- **Constraining (Decorating the AST; Type Checking)**
  Variable Scopes, Symbol Tables, Constraining Activities:
- **Code Generation**
  Frames (Activation Records), Runtime stack, Blocks
- **Mechanisms for Software Reuse**
  Inheritance vs. Composition (aggregation), Abstract classes vs. Interfaces, Combining Composition and Inheritance, Dynamic Composition
- **Implications of Inheritance**
  Polymorphic Variables, Memory Layout, Assignment, Clones (Shallow vs. Deep) Garbage Collection
- **Polymorphism**
  Polymorphic Variables, Overloading, Overriding, Replacement and Refinement Abstract Methods, Efficiency and Polymorphism
- **Input and Output Streams - Effective Uses of Inheritance with Composition**
  Readers, InputStreams
- **Exception Handling in Java Collection Classes**
  Arrays, Lists, Properties, System Properties
- **Application Profiling**
  Used to tune performance