# ITE 205 Software Design and Programming I 4 cr.

## Catalog description:

This course introduces a set of fundamental design principles and problem-solving techniques for the development of computer algorithms and their implementation as programs. Problem solutions are developed with the help of an appropriate modeling language and then coded in an object-oriented programming language. (Consult the Computer Science Department for the languages and tools currently in use.) Topics such as problem specification, object-oriented analysis and design, standard data types, control structures, methods and parameter passing, and design for reuse are presented through a study of specific example problems and solutions. Style, documentation, solution robustness, and conformance with specifications are emphasized throughout. Three lecture hours and three hours of scheduled laboratory per week plus extensive programming work outside of class.

## Prerequisite(s):

ITE 105

## Course Narrative

This is the first programming language course that introduces students to an object oriented programming language. Students are first familiarized with the syntax of the language. One of the most important steps in programming is to correctly analyze a problem statement. This can be accomplished through simple programs and lab assignments initially.

After students are able to write simple programs, compile, and execute, they are introduced to the concepts of how objects are created and used. This accomplishes the objective of being able to apply the fundamentals of object-oriented design methodology. Testing and debugging the programs is taught as the course proceeds. Students learn various programming structures like sequence, selection, and loop/repetition. One-dimensional arrays are also covered.

After the students learn the basics of a foundational programming course like this, they can tackle more complex programming tasks that form the basis of programming language courses taught after this course at higher levels.

## Goals:

The purpose of this course is to develop students' understanding of a coherent set of tools and techniques for creating computer solutions to simple problems in data manipulation. Upon completion of the course, a student should be able to do the following:

- G1: analyze a problem statement for completeness and clarity;
- G2: use the methodology of object-oriented design to develop class diagrams (data descriptions and methods) for a problem solution;
- G3: convert this solution into source code in the designated high-level programming language in accordance with a well-defined set of style rules;
- G4: debug and test the program;
- G5: provide clear documentation for the result

## Objectives:

Upon successful completion of the course, a student will have:

O1: demonstrated knowledge of the syntax elements of an object-oriented programming language;
O2: gained experience in analyzing problem statements for completeness and consistency;
O3: practiced standard techniques of problem analysis;
O4: applied the fundamentals of object-oriented design methodology;
O5: learned and utilized simple techniques for validation and verification of programs;
O6: created full documentation for several completed projects.

**Program Objective / Course Objective matrix** (For ABET Accreditation Purposes)

(The following Matrix maps the Program Objectives for Information Technology Program outlined by Accreditation Board of Engineering Technology (ABET) with the Course Objectives. The check marks below the course objective represent that those course objectives accomplish specific program objectives set forth by ABET. The program objectives that have a * in front of them means that that course does not address those program objectives.)

| Program Objective | **O1** | **O2** | **O3** | **O4** | **O5** | **O6** |
|---|---|---|---|---|---|---|
| **PO-A:** An ability to apply knowledge of computing and mathematics appropriate to the program's student outcomes and to the discipline. | ✓ | | | ✓ | ✓ | |
| **PO-B:** An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution. | | ✓ | ✓ | | ✓ | ✓ |
| **PO-C:** An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs. | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| ***PO-D:** An ability to function effectively on teams to accomplish a common goal. | | | | | | |
| ***PO-E:** An understanding of professional, ethical, legal, security and social issues and responsibilities. | | | | | | |
| ***PO-F:** An ability to communicate effectively with a range of audiences. | | | | | | |
| ***PO-G:** An ability to analyze the local and global impact of computing on individuals, organizations, and society. | | | | | | |
| ***PO-H:** Recognition of the need for and an ability to engage in continuing professional development. | | | | | | |
| **PO-I:** An ability to use current techniques, skills, and tools necessary for computing practice. | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| **PO-J:** An ability to use and apply current technical concepts and practices in the core information technologies. | | | ✓ | ✓ | | |
| **PO-K:** An ability to identify and analyze user needs and take them into account in the selection, creation, evaluation and administration of computer-based systems. | | | ✓ | ✓ | ✓ | |
| ***PO-L:** An ability to effectively integrate IT-based solutions into the user environment. | | | | | | |

| Program Objective | O1 | O2 | O3 | O4 | O5 | O6 |
|---|---|---|---|---|---|---|
| ***PO-M:** An understanding of best practices and standards and their application. | | | | | | |
| ***PO-N:** An ability to assist in the creation of an effective project plan. | | | | | | |

**Topics:**

The column on the right hand side represents the Body of Knowledge and number of hours (in parenthesis) set forth by ABET accreditation board for accomplishing minimum required hours assigned for different categories. More information on this body of knowledge can be found in Appendix A "The IT Body of Knowledge" on Page 68 of the following document.
http://www.acm.org//education/curricula/IT2008%20Curriculum.pdf

| | |
|---|---|
| The "problem-solving universe" | |
| o operational definition of computer (specifically, electronic stored-program digital computer) | |
| o components of a typical computer | |
| o fundamental computer capabilities | |
| Strategies and tools for problem solving | **PF2(1.5), PF4(1)** |
| o formulating precise specifications for a problem and its solution | |
| o algorithms | |
| o modular design of user requirements in measurable units | |
| o preconditions and post-conditions | |
| o specification of user requirements in measurable terms | |
| • Programming languages and programming language paradigms | **PF2(0.5)** |
| • Brief history and overview of the Java language | **IPT7(1)** |
| • Data types | **PF1(2)** |
| o basic data types: integer, real, character, Boolean | |
| o literals of each type | |
| • Variables and constants | **PF1(2)** |
| • Reference Types (including string and pre-defined wrapper classes) | **PF1(1), PF2(2)** |
| • Console output | **PF1(0.5)** |
| • Console input | **PF1(0.5)** |
| • Simplified graphical user interface (GUIs) | **PF5(3)** |
| o JOptionPane | |
| • Object-centered problem analysis | **PF3(0.5), PF4(0.5)** |
| • Object-centered design and implementation | **PF3(1)** |
| • Introduction to UML | **PF5(0.5)** |
| • Classes | **PF3(0.5), PF2(4)** |
| o overview | |
| o attributes | |
| o methods | **PF1(2)** |
| ▪ parameter and argument lists, return values, signatures | |
| ▪ use of modular design in creating methods | |
| ▪ visibility rules (scope, context) | |
| o objects (instances) | **PF3(0.5)** |
| ▪ handles | |
| ▪ copying objects (shallow vs. deep copies, clones) | |
| • Selection control structures | **PF1(2), PF2(0.5)** |

    - review of boolean expressions
    - single and double alternative structures
    - multiple-choice (switch) structure
- Testing and verification **SIA5(0.5)**
- Debugging **SIA5(0.5)**
- Repetition control structures (loops) **PF1(1), PL4(0.5)**
    - *while* and *do while* structures
    - *for* structure
- Object-oriented program design techniques **PF2(2), PF3(2.5),**
    - objects as self-contained entities
    - objects as entities that act upon themselves
        - contrast to other design methodologies in which external agents act upon objects
    - programming for reusability
- Collections (conceptual discussion) **PF2(0.5)**
- Arrays of one dimension **PF3(0.5), PF2(1)**
    - syntax rules
    - static nature of arrays; physical vs. logical size of an array
    - common algorithms: **PF4(3)**
        - storing a value in an array
        - removing a value from an array
        - linear traversal
        - linear search

**Student Experiences:**

**Programming assignments**
Ten to twelve programming assignments are given. One or more of these may be group projects. Each programming assignment normally involves the design, writing, testing and debugging of a program and the submission of an appropriate laboratory report. Each assignment has a specific due date, with a short grace period during which the assignment may be submitted for reduced credit. When the grace period has expired, the assignment will no longer be accepted.

All programs must be coded in the programming language currently used for instruction in ITE 205. The version of the language being used will be the currently accepted standard version: any extensions or variations in student-owned compilers must be approved in advance by the instructor, who may choose to forbid their use.

**Laboratory exercises**
There will be short programming exercises to be completed during weekly scheduled laboratory sessions. Each exercise focuses on a specific language feature or programming technique presented in recent lectures. Performance on these exercises will be incorporated into the course grade.

**Exams and quizzes**
There will be two examinations, two shorter quizzes, and a comprehensive written two-hour final examination.

**Final Grade**
Final grade will be determined using the following grading weights.

| | |
|---|---|
| Homework Assignments/ Lab Exercises/Quizzes | 50% |
| Midterm Examination | 25% |
| Final examination | 25% |

**Student Experiences by Course Outcome (Objective) matrix:**

| | O1 | O2 | O3 | O4 | O5 | O6 |
|---|---|---|---|---|---|---|
| Test/Quiz Questions | ✓ | ✓ | ✓ | ✓ | ✓ | |
| Homework Programming | | ✓ | ✓ | ✓ | ✓ | |
| Programming Projects | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Lab Exercises | ✓ | ✓ | ✓ | ✓ | ✓ | |

**Bibliography:**

Deitel, Harvey; Deitel, Paul. **Java How to Program: Early Objects Version. Tenth Edition.** Prentice Hall, 2014.

Flanagan, David; Evans J, Benjamin. **Java in a Nutshell**. **Sixth Edition**. O'Reilly,2014.

Gaddis, Tony. **Starting Out with Java™: Early Objects, Fifth Edition.** Addison-Wesley, 2014.

Gaddis, Tony. **Starting Out with Java: From Control Structures through Objects. Fifth Edition**. Addison Wesley, 2012.

Gaddis, Tony. **Starting Out with Java: From Control Structures through Data Structures. Second Edition**. Addison Wesley, 2011.

Horstmann, Cay. **Big Java: Early Objects**. **Fifth Edition.** John Wiley & Sons,2013.

Hosch, Frederick A.; Nino, Jaime. **An Introduction to Programming and Object-Oriented Design Using Java**. **Third Edition.** Wiley, 2008.

Liang, Y. Daniel. **Introduction to Java Programming, Comprehensive Version. Ninth Edition.** Prentice Hall, 2012.

Lewis, John; Loftus, William. **Java Software Solutions. Eight Edition.** Addison-Wesley, 2014.

Oaks, Scott; Wong, Henry. **Java Threads**. **Third Edition.** O'Reilly Media, 2005.

Schildt, Herbert. **Java : The Complete Reference**. **Ninth Edition.** Osborne/McGraw-Hill, 2014.

Bravaco, Ralph; Simonson, Ralph; Simonson, Shai. **Java Programming: From the Ground Up.** McGraw-Hill, 2009.

Carrano, Frank. **Imagine! Java: Programming Concepts in Context. First Edition**. Prentice Hall, 2010.

Dale, Nell; Weems, Chip; Headington, Mark. **Programming and Problem Solving With Java**. **Second Edition.** Jones & Bartlett, 2007.