Revised 1/2018

NOVA COLLEGE-WIDE COURSE CONTENT SUMMARY
ITP 100 SOFTWARE DESIGN (3 CR.)

Course Description

Introduces principles and practices of software development. Includes instruction in critical thinking, problem solving skills, and essential programming logic in structured and object- oriented design using contemporary tools. Lecture 3 hours per week.

General Purpose

In this course students will learn to design software using both structured programming and object-oriented programming concepts. Students will use pseudo-code, flow charts, design tools and UML. A specific programming language will NOT be used in this course.

Course Prerequisites/Corequisites

Prerequisite: Placement in MTH 154 or higher.

Course Objectives

Upon completion of this course, the student will be able to:

- Identify programming terminology and basic mechanics of programming necessary for success in programming courses.
- Describe the structured design building blocks applied within programming solutions.
- Illustrate structured design using an appropriation notational language (such as pseudo code and/or flow charting)
- Design the core concepts of object-oriented design
- Illustrate O-O designs using the standard Unified Modeling Language using an appropriate design tool

Major Topics to be Included

Program Design

- Program analysis and design within the system development life cycle.
- Evolution and development of both programming languages and program design.
- Difference between a design notational language and a design tool.
- Development of program algorithms using sequence, selection, and looping tools

Structured Design

- Design process for structured program design.
- Notational languages (pseudo code and/or flow charting) and design tools for structured design.
- Sequence, selection, and loop structures within a structured design solution for an operation.
- Decisions using null ELSE selections, nested selections, and CASE structures within a structured design solution.
- WHILE, UNTIL, FOR loops, and nested loops within a structured design
- Single dimensional and parallel arrays
- Modular code using predefined programming functions.
- Modular code with user written programming functions.
- Structured design solutions that involve one operation calling other operations with received and returned arguments.

Object-Oriented Design

- Design process for object-oriented program design.
- Unified Modeling Language (UML) within object oriented design.
- Tools for standard O-O notational language, the Unified Modeling Language.
- Candidate classes given a problem description.
- Attributes and methods for candidate classes given a problem description.
- Operations with a full UML signature including received and returned arguments.
- Object, class, message, data-hiding, information-hiding, encapsulation concepts.
- UML class diagrams.

Cooperative Team Work
- Design team approach to software design

Student Learning Outcomes

Program Design
- Relate the place of program analysis and design within the system development life cycle.
- Describe the evolution and development of both programming languages and program design.
- Describe the difference between a design notational language and a design tool.
- Develop program algorithms

Structured Design
- Describe a reasonable design process for structured program design.
- Identify one or more appropriate notational languages (such as pseudo code and/or flow-charting) and tools for structured design.
- Illustrate sequence, selection, and loop structures within a structured design solution.
- Illustrate decisions using null ELSE selections, nested selections, and CASE structures within a structured design solution for an operation.
- Demonstrate a working knowledge of when to apply the appropriate loop structure within a structured design.
- Illustrate WHILE, UNTIL, FOR loops, and nested loops within a structured design solution for an operation.
- Illustrate single dimensional and parallel arrays within a structured design solution.
- Illustrate modular code using predefined programming functions.
- Illustrate modular code with user written programming functions.
- Create structured design solutions that involve one operation calling other with received and returned arguments.

Object-Oriented Design
- Describe a reasonable design process for object-oriented program design.
- Explain the use and importance of the Unified Modeling Language within object oriented design.
- Identify appropriate tools to use with the standard O-O notational language, the Unified Modeling Language.
- Identify appropriate candidate classes given a problem description.
- Identify appropriate attributes and operations for candidate classes given a problem description.
- Show operations with a full UML signature including received and returned arguments.
- Define the terms object, class, message, data-hiding, information-hiding, encapsulation.
- Create a UML class diagram.

Cooperative Team Work
- Operate as a member of a design team on a given design task.
- Design a single task of a project while other teams or individuals are working on separate tasks of the same project.

Required Time Allocation per Topic

In order to standardize the core topics of ITP 100 so that a course taught at one campus is equivalent to the same course taught at another campus, the following student contact hours per topic are required. Each syllabus should be created to adhere as closely as possible to these allocations. Of course, the topics cannot be followed sequentially. Many topics are taught best as an integrated whole, often revisiting the topic several times, each time at a higher level. There are normally 45 student-contact-hours per semester for a three credit course. (This includes 15 weeks of instruction and does not include the final exam week so 15* 3 = 45 hours. Sections of the course that are given in alternative formats from the standard 16 week section still meet for the same number of contact hours.) The final exam time is not included in the time table. The category, 'Other Optional Content', leaves ample time for an instructor to tailor the course to special needs or resources.

Topic	Hours	Percentage
Introductory Programming Concepts	**3**	**6.5%**
Structured Program Design	**3**	**6.5%**
Decisions	**6**	**13%**
Loops	**7.5**	**17%**
Arrays	**7**	**16%**
Functions and Modular Programming	**7**	**16%**
Object Oriented Programming	**7**	**16%**
Other Optional Content	**2.5**	**5%**
Exams and Quizzes (NOT including Common Final Exam)	**2**	**4%**
Total	**45**	**100%**