Revised 6/2021

NOVA COLLEGE-WIDE COURSE CONTENT SUMMARY ITP 120 - JAVA PROGRAMMING I (4 CR.)

Course Description
Entails instruction in fundamentals of object-oriented programming using Java. Emphasizes program construction, algorithm development, coding, debugging, and documentation of console and graphical user interface applications. Lecture 4 hours per week.

General Purpose
This course provides a comprehensive foundation sufficient for a student to write Java programs from scratch in order to meet the minimum programming goals of students who plan to transfer and students who take the course for employment purposes.

Course Prerequisites/Corequisites
None

Course Objectives

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

- Design, develop, code, and test Java Programs console applications
- Use primitive data types and programming statements that are the building blocks of all programming
- Identify programming terminology and basic mechanics of programming necessary for success in programming.

Major Topics to be Included
- Design process for structured programs
- Primitive data types
- Datatypes
- Input and Output methods and techniques
- Selection Statements
- Boolean operators (and, or and not)
- Looping
- Methods
- Arrays (1-dimentional and parallel)
- Java Built-In Classes and Objects
- Files and Text I/O

Student Learning Outcomes
Primitive Data Types
- Design and Develop program algorithms using Java.
- Write Java statements to declare and initialize variables with appropriate datatypes.
- Write Java statements to declare named constants
- Write Java statements for Input and Output
- Write Java statements using Boolean operators: and (&&), or (||), not (!), exclusive or (^)
- Write Java statements using the arithmetic operators to include add, subtract, multiply, divide, and modulus
- Write Java statements using the increment and decrement operators

- Write Java statements incorporating cast operators for primitive data types in arithmetic expressions appropriately
- GUI Input
- o Write Java statements to complete GUI input using JOptionPane and showInputDialog o Write Java statements to include a prompt message and a window title for the GUI input
- GUI Output
- o Write Java statements to complete GUI output using JOptionPane and showMessageDialog
- o Write Java statements including an output message and window title for the GUI output
- Write Java statements to validate input of data

Selection Statements

- Write Java statements to create and use an if statement without an else statement (one branch)
- Write Java statements to create and use an if/else statement (two branches)
- Write Java statements to create and use an if/elseif/else statement (multi-level, many branches)
- Write Java statements to create and use an if statement within the body of another if statement (nested)
- Write Java statements to create and use switch/case statements including default option
- Write Java statements using the conditional operator

Loop Statements

- Write Java statements to construct and use a while loop
- Write Java statements to construct and use a do-while loop
- Write Java statements to construct and use a for loop
- Write Java statements to use loop statements appropriately for the following situations
- Be able to create and use a counter control loop appropriately
- Be able to create and use a data validation loop appropriately
- Be able to create a loop that is controlled with a sentinel value

Methods

- Write Java statements to create a method that takes one or more input arguments
- Write Java statements to create a method that takes no input arguments
- Write Java statements to create a method that returns a value
- Write Java statements to create a method that does not return a value
- Be able to differentiate between pass by value and pass by reference
- Describe the difference between static methods and non-static methods
- Write Java statements to create a method that takes an array as an input argument
- Write Java statements to create a method that returns an array
- Write Java statements to use local variables in a method
- Write Java statements which use common methods in the Math class: Math.sqrt, Math,abs, Math.max, Math.min, Math.random, Math.pow

Arrays

- Write Java statements to declare, create, and initialize a one-dimensional array
- Describe the memory allocation for a one-dimensional array including both the array reference and the memory for the array elements
- Write Java statements to fill an array with values
- Write Java statements to print out the contents of an array
- Write Java statements to copy one array to another array
- Write Java statements to access every element in an array for analysis for the data
- Write Java statements to load and process parallel arrays
- Be able to pass arrays to methods

Classes and Objects (Java Built-In)

- Be able to define the relationship between a class and an object

- Understand and use Java built-in classes such as, Scanner, String, File, Random, Math, Arrays, etc.

Strings

- Write Java statements to declare, create, and use a String object.
- Write Java statements using the following methods from the String class
  - equals, equalsIgnoreCase
  - compareTo, toCharArray( )
  - toUpperCase, toLowerCase
  - valueOf, charAt( )
  - concat
  - length
  - indexOf and lastIndexOf
  - substring
- Write Java statements to declare, create, and use a Character object.
- Write Java statements using the following methods from the Character class
  - compareTo
  - equals
- Write Java statements to declare, create, and use a StringBuffer object.
- Write Java statements using the toString from the StringBuffer class
- Write Java statements to create a toString method for a class that you have created.

Files

- Write Java statements to open a text file for reading
- Write Java statements to read data from a text file
- Write Java statements to close a text file
- Write Java statements to open a text file for writing
- Write Java statements to write data to a text file.
- Write Java statements using exceptions as required for file input and output.

Programs

- Be able to write Java programs using the results of an appropriate design methodology as a starting point
- Be able to write Java programs combining the statements described in this document
- Be able to write, compile, test, debug, and run Java programs

Required Time Allocation per Topic

In order to standardize the core topics of ITP 120 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 60 student contact-hours per semester for a four credit course. (This includes 14 weeks of instruction and does not include the final exam week so 14* 4.285 = 60 hours. Sections of the course that are given in alternative formats from the standard 15 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**
Structured Program Design	**5**	**8%**
Primitive data types	**6**	**10%**
Input and Output	**6**	**10%**
Selection Statements	**6**	**10%**
Loop Statements	**7**	**12%**
Methods	**8**	**13%**
Arrays	**8**	**13%**
Classes and Objects	**5**	**8%**
Strings	**4**	**7%**
Files	**5**	**8%**
Total	**60**	**100%**