Revised 4/2020

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 flow control statements that are the building blocks of all programming
- Use their foundation knowledge of object oriented coding techniques to create classes that are applied appropriately in a Java Program as a solution to a specific problem statement

Major Topics to be Included

- Primitive data types
- Input and Output methods and techniques
- Selection Statements and Looping
- Methods
- Arrays
- Classes and Objects
- Strings
- Files and Text I/O
- Inheritance and Polymorphism

Student Learning Outcomes

Primitive Data Types

- Write Java statements to declare and initialize integer, long, float, double, char, and boolean variables
- Write Java statements to assign values to variables using expressions containing mathematical and boolean operators
- Write Java statements to create arithmetic expressions following order of precedence rules
- Write Java statements using boolean operators: and (&&), or (||), not (!), exclusive or (^)
- Write Java statements using the relational operators to include greater than, greater than or equal to, less than, less than or equal to, is equal to, and not equal to
- 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 using pre and post increment and decrement operators
- Write Java statements using the short cut arithmetic assignment operators to include +=, -=, *=, /=, and %=
- Write Java statements incorporating cast operators for primitive data types in arithmetic expressions appropriately
- Write Java statements to declare named constants

- Write Java statements using both named and literal constants in java programming statements

Input and Output

- Console Input
  - Write Java statements to complete console input using Scanner class
  - Write Java statements including appropriate prompt messages for keyboard input
- Console Output Section
  - Write Java statements including print, println, and printf for output statements to the console
  - Write Java statements for console output including text labels and data values from the program
  - Write Java statements to include column headings and section headings in program output reports.
- GUI Input
  - Write Java statements to complete GUI input using JOptionPane and showInputDialog
  - Write Java statements to include a prompt message and a window title for the GUI input
- GUI Output
  - Write Java statements to complete GUI output using JOptionPane and showMessageDialog
  - Write Java statements including an output message and window title for the GUI output
- Write Java statements to validate input of data from the keyboard

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 create overloaded methods
- 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
- Describe the concept of parallel arrays and their relationship to classes and objects
- Write Java statements to declare, create, and initialize multi-dimensional arrays
- Be able to pass arrays to methods

Classes and Objects

- Be able to define the relationship between a class and an object
- Write Java statements to create a class that includes attributes and methods
- Write Java statements to create objects
- Be able to write visibility modifiers to include: public, private, protected, and no modifier
- Write Java statements to create and use static attributes
- Write Java statements to create and use constant attributes
- Be able to define the purpose of a constructor
- Write Java statements to create and use constructors
- Write Java statements to create and use an overloaded constructor
- Be able to define the purpose of an accessor method (get method)
- Write Java statements to create appropriate accessor methods for a class (get methods).
- Be able to define the purpose of a mutator method (set method).
- Write Java statements to create appropriate mutator methods for a class (set methods).
- Write Java statements to print out all attributes in a class
- Be able describe the following object oriented programming terms:
    - Encapsulation including information hiding and implementation hiding
    - Polymorphism
    - Inheritance
- Write Java statements that pass objects to methods
- Write Java statements that include objects as attributes
- Write Java statements that return an object from a method
- Use classes and objects in programs that you write from scratch.

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.

Inheritance

- Be able to use inheritance in programs and classes
- Be able to describe the relationship between a super class and a sub class.

- Be able to recognize a sub class from the extends label in a class header
- Be able to describe the purpose of the super statement when using inheritance
- Be able to write Java statements for simple classes using inheritance

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 15 weeks of instruction and does not include the final exam week so 15* 4 = 60 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
Primitive data types	**6**	**10.00%**
Input and Output	**4**	**6.67%**
Selection Statements	**4**	**6.67%**
Loop Statements	**4**	**6.67%**
Methods	**8**	**13.33%**
Arrays	**4**	**6.67%**
Classes and Objects	**8**	**13.33%**
Strings	**4**	**6.67%**
Testing to include quizzes, tests, and exams (not including common final exam)	**4**	**6.67%**
Files	**4**	**6.67%**
Inheritance	**4**	**6.67%**
Programs	**4**	**0.00%**
Other Optional Topics	**2**	**6.67%**
Total	**60**	**100%**