1 Eunduring Understanding, Learning Objective, Lesson Plan Map

2 Legend

2.1 Enduring Understanding

2.1.1 Learning Objective

2.1.1.1 Lesson Plan

3 Mapping

3.1 CON-1 The way variables and operators are sequenced and combined in an expression determines the computed result.

3.1.1 CON-1.A Evaluate arithmetic expressions in a program code.

3.1.1.1 2.01 Basic Data Concepts

3.1.1.2 2.02 Declaring & Assigning Variables

3.1.2 CON-1.B Evaluate what is stored in a variable as a result of an expression with an assignment statement.

3.1.2.1 2.03 String Concatenation & Increment Decrement Operators

3.1.3 CON-1.C Evaluate arithmetic expressions that use casting.

3.1.3.1 2.04 Mixing Types & Casting

3.1.4 CON-1.D Evaluate expressions that use the Math class methods.

3.1.4.1 3.02 Limitations of Parameters & Multiple Parameters

3.1.4.2 3.05 Using Objects & String Processing

3.1.5 CON-1.E Evaluate Boolean expressions that use relational operators in program code.

3.1.5.1 3.09 Relational Operators & if/else

3.1.5.2 3.10 Nested if/else Statements

3.1.6 CON-1.F Evaluate compound Boolean expressions in program code.

3.1.6.1 3.15 Fencepost & Sentinel Loops

3.2 CON-2 Programmers incorporate iteration and selection into code as a way of providing instructions for the computer to process each of the many possible input values.

3.2.1 CON-2.C Represent iterative processes using a while loop.

3.2.1.1 3.12 Cumulative Algorithms

3.2.1.2 3.13 while Loops

3.2.2 CON-2.D For algorithms in the context of a particular specification that does not require the use of traversals

3.2.2.1 3.14 Random Numbers

3.2.3 CON-2.E Represent iterative processes using a for loop

3.2.3.1 3.14 Random Numbers

3.2.4 CON-2.K Apply sequential/linear search algorithms to search for specific information in array or ArrayList objects.

3.2.4.1 7.01 Searching Algorithms

3.2.5 CON-2.L Apply selection sort and insertion sort algorithms to sort the elements of array or ArrayList objects.

3.2.5.1 7.02 Sorting Algorithms

3.2.6 CON-2.O Determine the result of executing recursive methods.

3.2.6.1 8.01 Thinking Recursively

3.2.6.2 8.02 Writing Recursive Solutions

3.2.6.3 8.03 Mechanics of Recursion

3.2.7 CON-2.P Apply recursive search algorithms to information in String, 1D array, or ArrayList objects.

3.2.7.1 8.04 MergeSort

3.3 MOD-1 Some objects or concepts are so frequently represented that programmers can draw upon existing code that has already been tested, enabling them to write solutions more quickly and with a greater degree of confidence.

3.3.1 MOD-1.A Call System class methods to generate output to the console.

3.3.1.1 1.03 String & Console Output

3.3.2 MOD-1.B Explain the relationship between a class and an object.

3.3.2.1 5.01 Object Oriented Programming

3.3.3 MOD-1.C Identify, using its signature, the correct constructor being called

3.3.3.1 3.01 Parameters

3.3.3.2 5.03 Object Initialization: Constructors

3.3.4 MOD-1.E Call non-static void methods without parameters.

3.3.4.1 5.02 Object State & Behavior

3.3.4.2 5.04 Encapsulation

3.4 MOD-2 Programmers use code to represent a physical object or nonphysical concept, real or imagined, by defining a class based on the attributes and/or behaviors of the object or concept.

3.4.1 MOD-2.A Designate access and visibility constraints to classes, data, constructors, and methods.

3.4.1.1 5.03 Object Initialization: Constructors

3.4.1.2 5.04 Encapsulation

3.4.1.3 5.05 Finding & Fixing Errors

3.4.2 MOD-2.C Describe the functionality and use of program code through comments.

3.4.2.1 1.04 Common Errors & Comments

3.4.3 MOD-2.D Define behaviors of an object through non-void methods without parameters written in a class.

3.4.3.1 5.01 Object Oriented Programming

3.4.3.2 6.02 Overriding Methods & Accessing Inherited Code

3.4.4 MOD-2.G Define behaviors of a class through static methods.

3.4.4.1 1.05 Static Methods & Method Calls

3.4.4.2 1.06 Static Methods & Method Calls

3.4.4.3 5.02 Object State & Behavior

3.5 MOD-3 When multiple classes contain common attributes and behaviors, programmers create a new class containing the shared attributes and behaviors forming a hierarchy. Modifications made at the highest level of the hierarchy apply to the subclasses

3.5.1 MOD-3.B Create an inheritance relationship from a subclass to the superclass.

3.5.1.1 6.03 Interacting with the Object Superclass

3.5.1.2 6.01 Inheritance Basics

3.5.2 MOD-3.C Define reference variables of a superclass to be assigned to an object of a subclass in the same hierarchy.

3.5.2.1 6.05 Has-a Relationships

3.5.3 MOD-3.D Call methods in an inheritance relationship.

3.5.3.1 6.04 Polymorphism

3.6 VAR-1 To find specific solutions to generalizable problems, programmers include variables in their code so that the same algorithm runs using different input values.

3.6.1 VAR-1.E For String class: a. Create String objects. b. Call String methods.

3.6.1.1 3.03 Return Values

3.6.2 VAR-1.G Explain where variables can be used in the program code.

3.6.2.1 1.02 Algorithms & Computational Thinking

3.7 VAR-2 To manage large amounts of data or complex relationships in data, programmers write code that groups the data together into a single data structure without creating individual variables for each value.

3.7.1 VAR-2.A Represent collections of related primitive or object reference data using two-dimensional (2D) array objects.

3.7.1.1 4.01 Array Basics

3.7.2 VAR-2.C Traverse the elements in a 1D array object using an enhanced for loop.

3.7.2.1 4.02 For-Each Loop & Arrays Class

3.7.3 VAR-2.D Represent collections of related object reference data using ArrayList objects.

3.7.3.1 4.07 ArrayList

3.7.4 VAR-2.F Traverse the elements in an ArrayList object using an enhanced for loop.

3.7.4.1 4.06 Nested Loop Algorithms & Rectangular Arrays

formatted by Markdeep 1.093