CompuScholar, Inc.

Alignment to Nevada **"Computer Science"**
Course Standards

Nevada Course Details:

Course Name:	Computer Science
Primary Cluster:	Information and Media Technologies Standards
Course Code(s):	n/a
Credit:	1
Grade Level:	9th-12th
State Standards	http://cteae.nv.gov/Career_and_Technical_Education/Standards/

CompuScholar Course Details:

Course Title:	TeenCoder: Java Programming (Abridged)
Course ISBN:	978-0-9887070-4-7
Course Year:	2015

Note 1: Citation(s) listed may represent a subset of the instances where objectives are met throughout the

Introduction

TeenCoder: Java Programming (Abridged) is an computer science course based on the Java language. This document demonstrates how the course meets standards within the Nevada Computer Science sequence. The Nevada standards listed below represent a 3-year / 3-credit program, so TeenCoder: Java Programming (Abridged) would be appropriate for the first-year course. Areas marked as "n/a" are deferred to second or third-year courses.

CONTENT STANDARD 1.0 : UNDERSTAND THE RELATIONSHIP BETWEEN HARDWARE AND SOFTWARE		
PERFORMANCE STANDARD 1.1 : DEMONSTRATE KNOWLEDGE OF THE RELATIONSHIP BETWEEN HARDWARE AND SOFTWARE		**CITATION(S)**
1.1.1	Demonstrate proper use of industry-standard terminology	New terms are introduced and used throughout the course
1.1.2	Examine the numbers systems: binary and hexadecimal	Chapter 17, Lesson 2
1.1.3	Describe machine limitations of finite representations (e.g., integer bounds, imprecision of floating- point representations, and round-	Chapter 17, Lesson 2
1.1.4	Describe the central processing unit (CPU) and memory	Chapter 1, Lesson 1
1.1.5	Compare and contrast low- and high-level programming languages	Chapter 1, Lesson 3

CONTENT STANDARD 2.0 : UNDERSTAND CONCEPTS OF PROBLEM SOLVING AND ALGORITHM DEVELOPMENT		
PERFORMANCE STANDARD 2.1 : UNDERSTAND A PROBLEM DESCRIPTION		**CITATION(S)**
2.1.1	Describe the problem	Chapter 17, Lesson 4
2.1.2	Demonstrate the solution(s) by hand	Chapter 17, Lesson 4
2.1.3	Explain how to validate the solution(s)	Chapter 17, Lesson 4
PERFORMANCE STANDARD 2.2 : DEVELOP AN ALGORITHM		**CITATION(S)**
2.2.1	Identify expected input and output	Chapter 17, Lesson 4
2.2.2	Utilize basic steps in algorithmic problem solving	Chapter 17, Lesson 4
2.2.3	Use logical thinking to create an algorithm utilizing pseudo code and/or a flow chart	Chapter 17, Lesson 4
2.2.4	Discuss top-down versus bottom-up development	n/a
PERFORMANCE STANDARD 2.3 : TEST ALGORITHMS		**CITATION(S)**
2.3.1	Generate test cases and expected results	Suppl. Lesson 4
2.3.2	Utilize test cases to walk through algorithms	n/a
2.3.3	Use results of the walk-through to adjust or modify algorithms	n/a

CONTENT STANDARD 3.0 : UNDERSTAND PROGRAMMING LANGUAGE CONCEPTS		
PERFORMANCE STANDARD 3.1 : UTILIZE PROGRAMMING CONSTRUCTS		**CITATION(S)**
3.1.1	Differentiate between syntax and semantics	n/a
3.1.2	Incorporate primitive data types	Chapter 4, Lesson 1 Chapter 4, Lesson 2
3.1.3	Demonstrate input from different sources	Chapter 6 (All) Chapter 13 (All) Chapter 18 (All)
3.1.4	Compare and contrast constants and variables	Chapter 4, Lesson 2
3.1.5	Select and implement conditional control	Chapter 7, Lesson 2
3.1.6	Select and implement iteration	Chapter 7, Lesson 4 Chapter 7, Lesson 5
3.1.7	Recognize and implement sequential control	Chapter 2, Lesson 2
3.1.8	Demonstrate output to different destinations	Chapter 6 (All) Chapter 13 (All) Chapter 18 (All)
3.1.9	Design and implement user-defined data types	Chapter 10, Lesson 2
3.1.10	Select and implement recursion	Chapter 19, Lesson 1
3.1.11	Illustrate pointers and reference variables	Chapter 4, Lesson 1

PERFORMANCE STANDARD 3.2 : PRACTICE PROCEDURAL PROGRAMMING		CITATION(S)
3.2.1	Design functions/methods	Chapter 8 (All)
3.2.2	Properly apply scope (i.e., global versus local)	Chapter 10, Lesson 2
3.2.3	Select appropriate parameter passing by value or by reference	Chapter 8, Lesson 2
3.2.4	Select when to use void versus non-void functions/methods	Chapter 8, Lesson 2
PERFORMANCE STANDARD 3.3 : PRACTICE OBJECT-ORIENTED PROGRAMMING (OOP)		**CITATION(S)**
3.3.1	Describe and implement abstract data types (i.e., data and functions)	Chapter 15 (All)
3.3.2	Employ modularity and reusability	Chapter 10, Lesson 1 Chapter 10, Lesson 2
3.3.3	Employ encapsulation and information hiding	Chapter 10, Lesson 1 Chapter 10, Lesson 3
3.3.4	Select and implement composition (“has a”) and/or inheritance (“is a”)	Chapter 10, Lesson 2 Chapter 15 (All) Chapter 16 (All)
3.3.5	Apply polymorphism	Chapter 15 (All) Chapter 16 (All)
3.3.6	Establish abstract base classes and interfaces	Chapter 15 (All) Chapter 16 (All)

CONTENT STANDARD 4.0 : DEVELOP PROGRAMS		
PERFORMANCE STANDARD 4.1 : USE PROPER IMPLEMENTATION STRATEGIES		**CITATION(S)**
4.1.1	Evaluate and select language and tools (i.e., development environment, IDE, debugger)	Chapter 2, Lesson 1 Chapter 2, Lesson 3 Chapter 3, Lesson 1 Chapter 9, Lesson 4
4.1.2	Select procedural versus OOP paradigm	Chapter 10, Lesson 1
4.1.3	Distribute code among multiple files	Chapter 10, Lesson 2 Chapter 15, Lesson 2
4.1.4	Resolve runtime exceptions and handle errors	Chapter 9, Lesson 2 Chapter 9, Lesson 3
PERFORMANCE STANDARD 4.2 : TEST AND DEBUG PROGRAMS		**CITATION(S)**
4.2.1	Employ debugging techniques (e.g., debugger, extra output statements, or hand-tracing codes) to identify and correct errors	Chapter 9, Lesson 3 Chapter 9, Lesson 4
4.2.2	Identify boundary cases and generate appropriate test data	Chapter 9, Lesson 3
4.2.3	Categorize errors (e.g., compile-time, run-time, logic, etc.)	Chapter 9, Lesson 1
4.2.4	Test classes and libraries in isolation	n/a
4.2.5	Perform integration testing of modules from multiple programmers	n/a

PERFORMANCE STANDARD 4.3 : ANALYZE ALGORITHMS		CITATION(S)
4.3.1	Informally compare and contrast run times (i.e., best- and worst-case scenarios)	Chapter 19, Lesson 2
4.3.2	Assess algorithms using Big-O notation	n/a (covered in our AP CS A Java course)

CONTENT STANDARD 5.0 : CULTIVATE GOOD PROGRAMMING STYLE		
PERFORMANCE STANDARD 5.1 : EMPLOY CODING STANDARDS		**CITATION(S)**
5.1.1	Use consistent naming conventions	Chapter 4, Lesson 2
5.1.2	Use meaningful identifiers	Chapter 4, Lesson 2
5.1.3	Use white space appropriately (e.g., indentation, blank lines, etc.)	Chapter 2, Lesson 2
5.1.4	Use named constants appropriately	Chapter 4, Lesson 2
5.1.5	Implement functions/methods to perform a single task	Chapter 8 (All)
5.1.6	Produce a code that is clear, concise, and easy to maintain	Chapter 2, Lesson 2 and all activities
5.1.7	Produce a code that compiles cleanly with no warning	Chapter 2, Lesson 3 and all activities
PERFORMANCE STANDARD 5.2 : DEMONSTRATE GOOD DOCUMENTATION SKILLS		**CITATION(S)**
5.2.1	Compose pre- and post-conditions for all functions and methods	n/a
5.2.2	Compose meaningful comments explaining critical or complex code	Chapter 2, Lesson 2
5.2.3	Describe program modules, variables, constants, and data types	Chapter 2, Lesson 4 Chapter 4, Lesson 1 Chapter 4, Lesson 2
5.2.4	Construct Unified Modeling Language (UML) class diagrams	Suppl. Lesson 5

CONTENT STANDARD 6.0 : UNDERSTAND STANDARD DATA STRUCTURES		
PERFORMANCE STANDARD 6.1 : UTILIZE SIMPLE DATA TYPES		**CITATION(S)**
6.1.1	Select appropriate primitive types	Chapter 4, Lesson 1
6.1.2	Implement declarations and initialization	Chapter 4, Lesson 2
6.1.3	Explain operators and order of operations	Chapter 7, Lesson 1
PERFORMANCE STANDARD 6.2 : DEMONSTRATE KNOWLEDGE OF ARRAYS		**CITATION(S)**
6.2.1	Manipulate strings as arrays	Chapter 5, Lesson 3
6.2.2	Assess implementation strategy (i.e., static or dynamic)	n/a
6.2.3	Access arrays (i.e., sequential, random)	Chapter 14, Lesson 1 Chapter 14, Lesson 3
6.2.4	Search arrays (i.e., sequential, binary)	Chapter 19, Lesson 3
6.2.5	Sort arrays (i.e., bubble, selection, insertion, and merge)	Chapter 19, Lesson 2

PERFORMANCE STANDARD 6.3 : DEMONSTRATE KNOWLEDGE OF CLASSES		CITATION(S)
6.3.1	Compare and contrast classes and objects	Chapter 10, Lesson 1 Chapter 10, Lesson 2
6.3.2	Implement class declaration	Chapter 10, Lesson 2
6.3.3	Create constructors and destructors	Chapter 11, Lesson 1
6.3.4	Implement overloaded and overridden functions/methods	Chapter 8, Lesson 2 Chapter 15, Lesson 4
6.3.5	Apply OOP concepts, including inheritance, polymorphism, interfaces, and abstract classes	Chapter 10 (All) Chapter 11 (All) Chapter 15 (All) Chapter 16 (All)
6.3.6	Manipulate strings as objects	Chapter 5 (All)
6.3.7	Explore template/generics	Chapter 14, Lesson 2
PERFORMANCE STANDARD 6.4 : DEMONSTRATE KNOWLEDGE OF LISTS		**CITATION(S)**
6.4.1	Use singly-linked lists	Chapter 14, Lesson 2
6.4.2	Explore doubly- and circularly-linked lists	Chapter 14, Lesson 2
6.4.3	Assess implementation strategy (i.e., array, dynamic)	n/a
6.4.4	Access list with and without iterators	Chapter 14, Lesson 2 Chapter 14, Lesson 3
6.4.5	Search lists (i.e., sequential and binary)	Chapter 14, Lesson 2 Chapter 14, Lesson 3
6.4.6	Implement insertion and deletion functions/methods	n/a (student will use, but not implement)
6.4.7	Compare and contrast sorting algorithms (i.e., bubble, selection, insertion, and merge)	n/a in context of lists. Array sorting found Chapter 14, Lesson 2
6.4.8	Select and implement stacks and queues	Suppl. Lesson 3
6.4.9	Use iterators	Chapter 14, Lesson 3

CONTENT STANDARD 7.0 : DEMONSTRATE KNOWLEDGE OF COMPUTING IN SOCIETAL CONTEXT		
PERFORMANCE STANDARD 7.1 : RECOGNIZE THE SYSTEM RELIABILITY ISSUES		**CITATION(S)**
7.1.1	Research examples of system failures and their impact	n/a
7.1.2	Explain cross-platform issues	n/a
PERFORMANCE STANDARD 7.2 : EXAMINE ETHICAL AND LEGAL ISSUES		**CITATION(S)**
7.2.1	Debate intellectual property, patent, and copyright laws	Chapter 1, Lesson 4
7.2.2	Describe the meaning of privacy in relation to the use of technology	Chapter 1, Lesson 4
7.2.3	Describe conflict of interest	Chapter 1, Lesson 4
7.2.4	Analyze non-disclosure agreements	n/a
7.2.5	Compare and contrast the code of ethics of the Institute of Electrical and Electronic Engineers (IEEE) and Association for Computing Machinery (ACM)	Chapter 1, Lesson 4 (ACM only)