CompuScholar, Inc.

Alignment to Utah's **Computer Programming II** Standards

Course Title: **TeenCoder: Java Programming**
Course ISBN: **978-0-9887070-2-3**
Course Year: **2015**

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

Computer Programming II, 2013 revision

Levels: **10-12**
Units of Credit: **1.00**
CIP Code: **11.0202**
Core Code: **35-02-00-00-040**
Prerequisites: **Secondary Math I, Computer Programming I**
Skill Test: **#830 Computer Programming II (C++)**
#835 Computer Programming II (Java)
#831 Computer Programming II (Python)
AP Computer Science
IB Computer Science
MTA Software Development Fundamentals (98-361)
MTA Windows Developmental Fundamentals (98-362)
MTA Microsoft .NET Fundamentals (98-372)
MTA Web Development Fundamentals (98-363)

COURSE DESCRIPTION

This is an advanced course in computer programming/software engineering and applications. It reviews and builds on the concepts introduced in CP I. It Introduces students to dynamic allocation of data, advanced utilization of classes, advanced GUI techniques, and advanced applications of recursion through the application of mathematical concepts.

CORE STANDARDS, OBJECTIVES, AND INDICATORS

STANDARD 1	CITATION(S)
Students will develop applications which make advanced use of the skills and concepts developed in Computer Programming IA & IB.	
Objective 1: Demonstrate the ability to develop complex applications.	
a. Develop complex applications using input, calculations, and output.	Chapter 17, Activity 1
b. Develop complex applications using control structures.	Chapter 7 Activity
c. Develop complex applications in object-oriented programming.	Chapter 16 Activities
d. Develop complex applications using data structures	Chapter 14 Activity, Chapter 19
e. Develop complex applications using files (sequential files).	Chapter 18
Objective 2: Utilize recursive algorithms	
a. Analyze and solve recursive methods	Chapter 19, Lesson 1
b. Utilize recursive algorithms to solve a problem	Chapter 19 Activity
c. Identify the base case, recursive case, and action of each recursive function or method	Chapter 19, Lesson 1
d. Understand the use of a recursive helper function or method	Chapter 19, Lesson 1
Objective 3: Create advanced functions and methods	
a. Understand overloading.	Chapter 15, Lesson 4
b. Create and use overloaded operators (C++).	n/a

STANDARD 2	CITATION(S)
Students will use searching and sorting algorithms.	
Objective 1: Demonstrate the ability to search data structures in programs.	
a. Develop a binary search.	Chapter 19 Activity
b. Compare efficiency and appropriateness of sequential and binary searches.	Chapter 19, Lesson 3
Objective 2: Demonstrate the ability to sort data structures in programs.	
a. Sort arrays using iterative sorting algorithms (selection, insertion, bubble).	Chapter 19, Lesson 2
b. Recognize recursive sorting algorithms (merge, quick, heap).	Chapter 19, Lesson 2
c. Compare the efficiency of differing sorting algorithms.	Chapter 19, Lesson 2

STANDARD 3	CITATION(S)
Students will utilize multidimensional arrays.	
Objective 2: Utilize multidimensional arrays.	
a. Initialize multidimensional arrays.	Chapter 14, Lesson 1
b. Input and output data into and from multidimensional arrays.	Chapter 14, Lesson 1
c. Perform operations on multidimensional arrays.	Chapter 14, Lesson 1
d. Perform searches on multidimensional arrays.	Chapter 14, Lesson 1

STANDARD 4	CITATION(S)
Students will properly employ dynamic data structures and abstract data types (ADTs).	
Objective 1: Demonstrate the ability to use stacks in programs.	
a. Declare stack structures.	Supplemental Lesson 3
b. Initialize stacks.	Supplemental Lesson 3
c. Check for empty and full stacks.	Supplemental Lesson 3
d. Push on to and pop off values from stacks.	Supplemental Lesson 3
e. Develop an application that utilizes stacks.	Supplemental Lesson 3 Activity
Objective 2: Demonstrate the ability to use queues in programs.	
a. Declare queue structures.	Supplemental Lesson 3
b. Check for empty and full queues.	Supplemental Lesson 3
c. Initialize queues.	Supplemental Lesson 3
d. Enqueue values on to and dequeue values off of queues.	Supplemental Lesson 3
e. Develop an application that utilize queues.	Supplemental Lesson 3 Activity
Objective 3: Utilize type-safe data structures (generics or templates)	
a. Utilize data structures that are type-safe using generics or templates.	Chapter 14, Lesson 2
b. Understand how type-safe data structures prevent errors.	Chapter 14, Lesson 2
c. Optional -- Demonstrate how to create a template or generic class.	n/a

STANDARD 5	CITATION(S)
Students will design and implement advanced objected oriented concepts.	
Objective 1: Implement object oriented programs	
a. Create classes that have high cohesion and low coupling.	Chapter 10, Lesson 1
b. Understand and use composition and aggregation (HAS-A) relationships.	Chapter 10, Lesson 2
c. Understand the use of class variables (static variables).	Chapter 10, Lesson 2 Chapter 11, Lesson 3
Objective 2: Implement Inheritance in an objected oriented program.	
a. Utilize class hierarchies (parent-child relationships).	Chapter 10, Lesson 2
b. Understand IS-A Relationships.	Chapter 10, Lesson 2
c. Override methods. Understand how to call the overriding method from the child.	Chapter 15, Lesson 4
d. Understand the protected modifier	Chapter 10, Lesson 3
e. Call a parent class constructor from the child's constructor	Chapter 11, Lesson 1
Objective 3: Create and use abstract classes	
a. Create and implement abstract classes	Chapter 15, Lesson 2
b. Implement interfaces (purely abstract classes).	Chapter 11, Lesson 2
Objective 4: Implement polymorphism	
a. Demonstrate that a parent object variable can hold an instance of a child class.	Chapter 15, Lesson 3
b Demonstrate typecasting of inherited objects	Chapter 15, Lesson 3
c. Determine IS-A relationships via code (eg: instanceof, typeof, isa)	n/a
Objective 5: Demonstrate overriding techniques.	
a. Demonstrate method overloading	Chapter 15, Lesson 4
b. Utilize method virtualization (implicit in Java, Python, explicit in C++, C#)	Chapter 15, Lesson 4

STANDARD 6	CITATION(S)
Students will use Unified Modeling Language (UML) to design object oriented programs	
Objective 1: Demonstrate the use of UML in design.	
a. Create an activity diagram.	n/a
b. Create a class diagram for the class hierarchy of a program.	Chapter 15, Lesson 2
c. Create a sequence diagram for a method.	n/a
d. Translate an activity diagram to code.	n/a

STANDARD 7	CITATION(S)
Students will develop a program of significant complexity as part of a portfolio.	
Objective 1: Create an individual program of significant complexity and size (300-500 lines).	
a. Create design documentation for the project.	Chapter 25 Activity 2
b. Follow accepted object-oriented programming methodology when writing the code.	Chapter 25 Activity 3
Objective 2: Compile a portfolio of the individual and group programs developed.	
a. Include sample design work	Chapter 25
b. Include sample program source code and output	Chapter 25

Optional STANDARD 8	CITATION(S)
Students will participate in a work-based learning experience and/or competition.	
Objective 1: Participate in a work-based learning experience.	
a. Take a field trip to a software engineering firm	n/a
b. Participate in a Job shadow	n/a
c. Work an internship	n/a
d. Listen to an Industry guest speaker or post-secondary guest speaker	n/a
e. Intverview with an industry representive	n/a
f. Participate in a competition	n/a