Computer Science II
At-A-Glance - Lamar CISD

Ongoing Skills Imbedded All Year	**Professional Standards/Employability Skills/Technical Skills**		
	● Increase and diversify participation in computer science ● Students, regardless of prior experience in computing, will develop confidence using computer science as a tool to express themselves and solve problems, and this confidence will prepare them for success in future endeavors in the field of computer science ● Students will understand the core principles of computing, a field which has and continues to change the world ● Students will be able to develop computational artifacts to solve problems, communicate ideas, and express their own creativity ● Students will be able to collaborate with others to solve problems and develop computational artifacts 1● Students will be able to explain the impact computing has on society, economy, and culture ● Students will be able to analyze existing artifacts, identify and correct errors, and explain how the artifact functions ● Students will be able to explain how data, information, or knowledge is represented for computational use ● Students will be able to explain how abstractions are used in computation and modeling ● Students will learn to be informed and responsible users of technology		
Ongoing Ways to Show	Complete labs and assignments individually or as a team.		
Grading Period	**Unit Name**	**Estimated Time Frame**	**TEKS**
Grading Period 1 **29 Days**	**Introductory Skills/Set Up**	**4 Days**	
	• Classroom rules and expectations • Introduction to computer lab and resources		
	Unit 1: System Administration	**12 Days**	6A, 6B, 6F
	• Operating Systems • Software and Applications • Application Security • Browser Configuration • System Administration • Command Line Interface 6(A) The student will compare and contrast types of operating systems, software applications, hardware platforms, and programming languages. 6(B) The student will demonstrate knowledge of major hardware components, including primary and secondary memory, a central processing unit (CPU), and peripherals. 6(F) The student will differentiate among the categories of programming languages, including machine, assembly, high-level compiled, high-level interpreted, and scripted.		
	Unit 2: Networking	**13 Days**	3A, 3B, 3C, 5A, 5B, 5C, 6C, 6D, 6E
	• Introduction to the Internet • Notational Systems • Data Representation • Internet Hardware • Internet Addresses • Domain Name System (DNS) • Routing • Packets and Protocols • The Internet and Cybersecurity • Creative Credit & Copyright • Impact of the Internet 3(A) The student will use local area networks (LANs) and wide area networks (WANs), including the Internet and intranets, in research, file management, and collaboration. 3(B) The student will understand programming file structure and file access for required resources 3(C) The student will acquire and process information from text files, including files of known and unknown sizes. 5(A) The student will model ethical acquisition and use of digital information. 5(B) The student will demonstrate proper digital etiquette, responsible use of software, and knowledge of acceptable use policies. 5(C) The student will investigate digital rights management.		

	6(C) The student will demonstrate knowledge of major networking components, including hosts, servers, switches, and routers. 6(D) The student will demonstrate knowledge of computer communication systems, including single-user, peer to-peer, workgroup, client-server, and networked. 6(E) The student will demonstrate knowledge of computer addressing systems, including Internet Protocol (IP) address and Media Access Control (MAC) address.		
Grading Period 2 **27 Days**	**Unit 3: Intro to Programming (Karel)**	**6 Days**	1D, 4A, 4B, 4C, 4D, 4E
	• Commands • Defining vs. Calling Methods • Designing Methods • Program Entry Points • Control Flow • Looping • Conditionals • Classes • Commenting Code • Preconditions and Postconditions • Top-Down Design 1(D) The student will use compare and contrast design methodologies and implementation techniques such as topdown, bottom-up, and black box. 4(A) The student will develop sequential algorithms using branching control statements, including nested structures, to create solutions to decision-making problems. 4(B) The student will develop choice algorithms using selection control statements based on ordinal values. 4(C) The student will demonstrate proficiency in the use of short-circuit evaluation. 4(D) The student will demonstrate proficiency in the use of Boolean algebra, including De Morgan's Law. 4(E) The student will develop iterative algorithms using nested loops.		
	Unit 4A: Basic Java	**20 Days**	4A, 4B, 4C, 4D, 4E
	• Printing • Variables • Types • Arithmetic Expressions • Casting ints and doubles • Input/Output • Errors • Loops • Conditionals 4(A) The student will develop sequential algorithms using branching control statements, including nested structures, to create solutions to decision-making problems. 4(B) The student will develop choice algorithms using selection control statements based on ordinal values. 4(C) The student will demonstrate proficiency in the use of short-circuit evaluation. 4(D) The student will demonstrate proficiency in the use of Boolean algebra, including De Morgan's Law. 4(E) The student will develop iterative algorithms using nested loops.		
Grading Period 3 **28 Days**	**Unit 4B Basic Java**	**15 Days**	4A, 4B, 4C, 4D, 4E
	• De Morgan's Laws • Short Circuit Evaluation • Debugging • Nested Control Structures • Working with the Java String class • Computer Ethics 4(A) The student will develop sequential algorithms using branching control statements, including nested structures, to create solutions to decision-making problems. 4(B) The student will develop choice algorithms using selection control statements based on ordinal values. 4(C) The student will demonstrate proficiency in the use of short-circuit evaluation. 4(D) The student will demonstrate proficiency in the use of Boolean algebra, including De Morgan's Law. 4(E) The student will develop iterative algorithms using nested loops.		
	Unit 5 Methods	**15 Days**	3D, 4N, 4V
	• Methods • Parameters • Return values • Javadocs		

Computer Science II
Lab Safety and Scientific Processes
Readiness Standards
Supporting Standards

- @param
- @return
- Iterating Over Characters
- Java Exceptions
- Compile-Time vs Run-Time Exceptions
- Java String Class and Methods

Java Character Class and Methods
3(D) The student will manipulate data structures using string processing.
4(N) The student will identify and debug compile, syntax, runtime, and logic errors.
4(V) The student will create object-oriented definitions using class declarations, variable declarations, constant declarations, method declarations, parameter declarations, and interface declarations.

Grading Period 4
31 Days

Unit 6 Classes and Object-Oriented Programming	**20 Days**	4T, 4U, 4W, 4X, 4Y, 4Z, 4AA, 4BB, 4CC, 4DD, 4EE, 4FF, 4GG, 4HH, 4II, 4JJ, 4KK, 4LL, 4MM

- Using Classes as a Client
- Classes vs Objects
- Class Methods
- Instance Variables
- Constructors
- Visibility
- Information Hiding
- this
- static
- super
- The Java Math Class and Methods
- Creating Random Values
- Designing Classes
- Creating Classes
- Getter and Setter Methods
- Inheritance
- Method Overloading
- Local Variables and Scope
- Comparing Objects vs Primitive Types
- Abstract Classes
- Packages
- Polymorphism
- Interfaces
- Modifying Classes
- Object is the Superclass

4(T) The student will identify, understand, and create class specifications and relationships among classes, including composition and inheritance relationships.
4(U) The student will understand and explain object relationships among defined classes, abstract classes, and interfaces.
4(V) The student will create object-oriented definitions using class declarations, variable declarations, constant declarations, method declarations, parameter declarations, and interface declarations.
4(W) The student will create robust classes that encapsulate data and the methods that operate on that data and incorporate overloading to enrich the object's behavior.
4(X) The student will design and implement a set of interactive classes.
4(Y) The student will design, create, and evaluate multiclass programs that use abstract classes and interfaces.
4(Z) The student will understand and implement a student-created class hierarchy.
4(AA) The student will extend, modify, and improve existing code using inheritance.
4(BB) The student will create adaptive behaviors, including overloading, using polymorphism.
4(CC) The student will understand and use reference variables for object and string data types.
4(DD) The student will understand and implement access scope modifiers.
4(EE) The student will understand and demonstrate how to compare objects.
4(FF) The student will duplicate objects using the appropriate deep and/or shallow copy.
4(GG) The student will define and implement abstract classes and interfaces in program problem solutions.
4(HH) The student will apply functional decomposition to a program solution.
4(II) The student will create simple and robust objects from class definitions through instantiation.
4(JJ) The student will apply class membership of variables, constants, and methods.
4(KK) The student will examine and mutate the properties of an object using accessors and modifiers.
4(LL) The student will understand and implement a composite class.
4(MM) The student will design and implement an interface.

	Unit 7A: Data Structures	**11 Days**	3D, 3E, 3F, 3G, 3H
	• Declaring and Initializing Arrays • Constructing ArrayLists • Indexing Into Arrays/ArrayLists • Iterating Over Arrays/ArrayLists • Getting the Length of an Array/ArrayLists • ArrayIndexOutOfBoundsException • IndexOutOfBoundsException • Object References • Arrays/ArrayLists as Parameters • Arrays/ArrayLists as Return Values • Inserting and Deleting Elements • Wrapper Classes • Storing Objects in Arrays vs. ArrayLists • Numerical Representations of Integers • The List Interface • Declaring and Initializing 2D Rectangular Arrays • Using Nested Loops to Iterate through 2D Arrays • Row-Major order • HashMaps 3(D) The student will manipulate data structures using string processing. 3(E) The student will manipulate data values by casting between data types. 3(F) The student will identify and use the structured data type of one-dimensional arrays to traverse, search, modify, insert, and delete data. 3(G) The student will identify and use the structured data type of two-dimensional arrays to traverse, search, modify, insert, and delete data. 3(H) The student will identify and use a list object data structure to traverse, search, insert, and delete data.		
Grading Period 5 **30 Days**	**Unit 7B: Data Structures**	**9 Days**	3D, 3E, 3F, 3G, 3H
	• Declaring and Initializing Arrays • Constructing ArrayLists • Indexing Into Arrays/ArrayLists • Iterating Over Arrays/ArrayLists • Getting the Length of an Array/ArrayLists • ArrayIndexOutOfBoundsException • IndexOutOfBoundsException • Object References • Arrays/ArrayLists as Parameters • Arrays/ArrayLists as Return Values • Inserting and Deleting Elements • Wrapper Classes • Storing Objects in Arrays vs. ArrayLists • Numerical Representations of Integers • The List Interface • Declaring and Initializing 2D Rectangular Arrays • Using Nested Loops to Iterate through 2D Arrays • Row-Major order • HashMaps 3(D) The student will manipulate data structures using string processing. 3(E) The student will manipulate data values by casting between data types. 3(F) The student will identify and use the structured data type of one-dimensional arrays to traverse, search, modify, insert, and delete data. 3(G) The student will identify and use the structured data type of two-dimensional arrays to traverse, search, modify, insert, and delete data. 3(H) The student will identify and use a list object data structure to traverse, search, insert, and delete data.		
	Unit 8: Lab (Teacher Selected)	**6 Days**	
	Options: Steganography Lab, Celebrity Lab, Picture (Filtering) Lab		

Computer Science II
Lab Safety and Scientific Processes
Readiness Standards
Supporting Standards

	Unit 9: Algorithms and Recursion	**15 Days**	4F, 4G, 4H, 4I, 4J, 4K, 4L, 4M, 4O, 4P, 4Q, 4R, 4S
	• Implementing and using Sequential Search • Implementing and using Binary Search • Comparing relative run times of Sequential and Binary Search • Brief introduction to Big-Oh • Counting comparisons in searches and sorts • Insertion Sort • Selection Sort • Merge Sort • Pros and cons of each sorting algorithm • Divide and Conquer • Recursion • java.util.Arrays • Sorting and searching with both arrays and ArrayLists 4(F) The student will identify, trace, and appropriately use recursion in programming solutions, including algebraic computations. 4(G) The student will design, construct, evaluate, and compare search algorithms, including linear searching and binary searching. 4(H) The student will identify, describe, design, create, evaluate, and compare standard sorting algorithms, including selection sort, bubble sort, insertion sort, and merge sort; (I) measure time/space efficiency of various sorting algorithms. 4(I) The student will measure time/space efficiency of various sorting algorithms. 4(J) The student will compare and contrast search and sort algorithms, including linear, quadratic, and recursive strategies, for time/space efficiency. 4(K) The student will analyze algorithms using "big-O" notation for best, average, and worst-case data patterns. 4(L) The student will develop algorithms to solve various problems, including factoring, summing a series, finding the roots of a quadratic equation, and generating Fibonacci numbers. 4(M) The student will test program solutions by investigating boundary conditions; testing classes, methods, and libraries in isolation; and performing stepwise refinement. 4(O) The student will compare and contrast algorithm efficiency by using informal runtime comparisons, exact calculation of statement execution counts, and theoretical efficiency values using "big-O" notation, including worst-case, best-case, and average-case time/space analysis. 4(P) The student will demonstrate the ability to count, convert, and perform mathematical operations in the binary and hexadecimal number systems. 4(Q) The student will demonstrate knowledge of the maximum integer boundary, minimum integer boundary, imprecision of real number representations, and round-off errors. 4(R) The student will create program solutions to problems using the mathematics library class. 4(S) The student will use random algorithms to create simulations that model the real world.		
Grading Period 6 **27 Days**	**Java Level 1 Industry-Based Certification: Preparation and Test**	**10 Days**	
	• GMetrix Preparation • Certiport Examination		
	Enrichment Topics	**17 Days**	1A, 1B, 1C, 1E, 1F, 1G, 1H, 1I, 2A, 2B, 2C, 2D, 2E, 2F, 2G,
	• Robots • Arcade Games 1(A) The student will use program design problem-solving strategies to create program solutions. 1(B) The student will use demonstrate the ability to read and modify large programs, including the design description and process development. 1(C) The student will use follow the systematic problem-solving process of identifying the specifications of purpose and goals, the data types and objects needed, and the subtasks to be performed. 1(E) The student will use analyze, modify, and evaluate existing code by performing a case study on a large program, including inheritance and black box programming. 1(F) The student will use identify the data types and objects needed to solve a problem. 1(G) The student will choose, identify, and use the appropriate abstract data type, advanced data structure, and supporting algorithms to properly represent the data in a program problem solution. 1(H) The student will use object-oriented programming development methodology, data abstraction, encapsulation with information hiding, and procedural abstraction in program development and testing. 1(I) The student will create, edit, and manipulate bitmap images that are used to enhance user interfaces and program functionality		

Computer Science II
Lab Safety and Scientific Processes
Readiness Standards
Supporting Standards

	2(A) The student will use the principles of software engineering to work in software design teams, break a problem statement into specific solution requirements, create a program development plan, code part of a solution from a program development plan while a partner codes the remaining part, team test the solution for correctness, and develop presentations to report the solution findings. 2(B) The student will create interactive console display interfaces with appropriate user prompts. 2(C) The student will create interactive human interfaces to acquire data from a user and display program results using an advanced Graphical User Interface (GUI). 2(D) The student will write programs and communicate with proper programming style to enhance the readability and functionality of the code by using meaningful descriptive identifiers, internal comments, white space, indentation, and a standardized program style. 2(E) The student will improve data display by optimizing data visualization. 2(F) The student will display simple vector graphics to interpret and display program results. 2(G) The student will display simple bitmap images.