

UTeach Computer Science

AP Computer Science A Syllabus and Planning Guide (2023–2024)

Table of Contents

Curricular Requirements	2
Curriculum Description	3
Course Framework	4
Course Content	6
Unit Guides	7
Unit 1: Introductions Are in Order	9
Unit 1 Description and Topics	9
Unit 1 Schedule	10
Unit 2: Primitive Control	11
Unit 2 Description and Topics	11
Unit 2 Schedule	13
Unit 3: Strings and Iteration	15
Unit 3 Description and Topics	15
Unit 3 Schedule	16
Unit 4: Objects, Classes, and Methods	18
Unit 4 Description and Topics	18
Unit 4 Schedule	19
Unit 5: Arrays, ArrayLists, and 2D Arrays	22
Unit 5 Description and Topics	22
Unit 5 Schedule	24
Unit 6: Inheritance	27
Unit 6 Description and Topics	27
Unit 6 Schedule	28
Unit 7: Searching, Sorting, and Recursion	30
Unit 7 Description and Topics	30
Unit 7 Schedule	31
Pedagogical Approaches	32
Resources and Technical Requirements	36

Curricular Requirements

Curricular Requirements		Pages
CR-1	The teacher and students have access to a college-level computer science textbook, in print or electronic format.	3
CR-2	The course provides opportunities to deepen student understanding of the required content outlined in each of the units described in the AP Course and Exam description.	7–8
CR-3	The course provides opportunities to deepen student understanding of the Big Ideas.	6
CR-4	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 1: Program Design and Algorithm Development.	4
CR-5	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 2: Code Logic.	4–5
CR-6	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 3: Code Implementation.	5
CR-7	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 4: Code Testing.	5
CR-8	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 5: Documentation.	5
CR-9	The course provides students with hands-on lab experiences to practice programming through designing and implementing computer-based solutions to problems.	34

Curriculum Description

Developers

UTeach AP Computer Science A has been developed by the UTeach Institute ([UTeach Computer Science](#)) in collaboration with A+ College Ready Alabama.

Curriculum Overview

UTeach AP Computer Science A has been designed as a year-long high school curriculum that fully addresses the Big Ideas, computational thinking practices and skills, and sequenced curriculum units, as specified by the College Board’s AP Computer Science A curriculum framework.

The lessons and materials used throughout this curriculum incorporate Project-Based Learning (PBL), a pedagogical approach that actively engages students in the educational process, improves retention, and develops problem-solving, critical-thinking, and group communication skills. Through this collaborative, learner-centric approach, students are encouraged to explore the advantages and societal impact of computational technology while developing their programming and computational thinking skills through Java. *It is required for students to have daily access to the internet.*

Textbook

[CR-1] UTeach CS A has an interactive online textbook available for students and teachers. The textbook, assessments, projects, and built-in programming environments are hosted on the cloud-based Codio platform, which can be integrated with most learning management systems and is FERPA compliant.

Programming Language Requirements

Students use the Java programming language throughout the curriculum activities and assignments.

Bibliography

- Almossawi, A. (2014). *An illustrated book of bad arguments*. New York, NY: The Experiment.
- Parlante, Nick. (2017) *CodingBat Java*, CodingBat. <https://codingbat.com/java>
- Touretzky, D. S. (2013). Chapter 8: Recursion. *COMMON LISP: A gentle introduction to symbolic computation* (pp. 281–286). Mineola, NY: Dover Publications.
- Williams, L. A., & Kessler, R. R. (2000). All I really need to know about pair programming I learned in kindergarten. *Communications of the ACM*, 43(5), 108–114. <https://doi.org/10.1145/332833.332848>

Course Framework

The course framework consists of two components: 1) Computational Thinking Practices and 2) Course Content, which includes Big Ideas, Enduring Understandings, Learning Objectives, and Essential Knowledge Statements.

Computational Thinking Practices

Computational Thinking Practices: Skills [CR 4-8]				
1 Program Design and Algorithm Development: Determine required code segments to produce a given output.	2 Code Logic: Determine the output, value, or result of given program code given initial values.	3 Code Implementation: Write and implement program code.	4 Code Testing: Analyze program code for correctness, equivalence, and errors.	5 Documentation: Describe the behavior and conditions that produce identified results in a program.
1.A Determine an appropriate program design to solve a problem or accomplish a task (not assessed).	2.A Apply the meaning of specific operators.	3.A Write program code to create the objects of a class and call methods.	4.A Use test-cases to find errors or validate results.	5.A Describe the behavior of a given segment of program code.
1.B Determine code that would be used to complete code segments.	2.B Determine the result or output based on statement execution order in a code segment without method calls (other than output).	3.B Write program code to define a new type by creating a class.	4.B Identify errors in program code.	5.B Explain why a code segment will not compile or work as intended.
1.C Determine code that would be used to interact with completed program code.	2.C Determine the result or output based on the statement execution order in a code segment containing method calls.	3.C Write program code to satisfy method specifications using expressions, conditional statements and iterative statements.	4.C Determine if two or more code segments yield equivalent results.	5.C Explain how the result of program code changes, given a change to the initial code.
	2.D Determine the number of times a code segment will execute.	3.D Write program code to create, traverse, and manipulate elements in a 1D array or ArrayList objects.		5.D Describe the initial conditions that must be met for a program segment to work as intended or described.
		3.E Write program code to create, traverse, and manipulate elements in 2D array objects.		

The following are examples in the curriculum of an instructional approach or activity that describes how students will engage with these skills:

CR-4: 1.B For most of the programming assignments and the unit projects, students build on starter code in Java using instructions on how to complete the code to make a working solution to a given problem, which assesses mastery of the topics presented in the lesson.

CR-5: 2.D Students engage in an activity in Unit 7 during the lesson Compare Big O Informally in which they read through a code segment and determine how many times the data structure

is being accessed. Then they open a program with the same code and run the program to compare their prediction with the actual statement execution times.

CR-6: **3.B** The Unit 4 project, Disease Diagnoser, and the Unit 6 project, Hospital Locator, ask students to create a class and write the code to test the implementation of the class.

3.D **3.E** The Unit 5 project, Air Quality Analyzer, has students create, traverse, and manipulate data in 1D arrays, ArrayLists, and a 2D array

CR-7: **4.A** Throughout the programming activities, students are asked to test their program code using various test cases. One example is the Palindrome program that students write in Unit 3 in which students are to test their code using at least 12 different words to check if they are palindromes. For the Unit 7 project in which students write searching and sorting algorithms, students use a small data set for the initial testing and then are given a much larger data set to test the efficiency of their algorithms.





CR-8: **5.B** One of the instructional strategies used as a check for understanding is Thumbs Up/Thumbs Down. Students review a code segment and show a thumbs up if the code will work as intended or thumbs down if the code will not compile or work as intended. The students explain the syntax or logic error for thumbs down code segments. This strategy is used in Unit 2 for conditionals and Unit 3 for iteration.




Course Content – Big Ideas

The Big Ideas serve as the foundation of the course. They are overarching concepts or themes that are spiraled throughout all seven of the curriculum units and connected to the topics and activities within each unit.

Big Ideas [CR 3]	
<p>Big Idea 1: MODULARITY (MOD) Incorporating elements of abstraction, by breaking problems down into interacting pieces, each with their own purposes, makes writing complex programs easier. Abstracting simplifies concepts and processes by looking at the big picture rather than being overwhelmed by the details. Modularity in object-oriented programming allows us to use abstraction to break complex programs down into individual classes and methods.</p>	<p>Along with several smaller programming assignments within Unit 6, the unit project, Hospital Locator, allows students to demonstrate mastery of the Modularity Big Idea by constructing a database consisting of a superclass and several subclasses, populating the database with real-world data, and searching the data.</p>
<p>Big Idea 2: VARIABLES (VAR) Information used as a basis for reasoning, discussion, or calculation is referred to as <i>data</i>. Programs rely on variables to store data, on data structures to organize multiple values when program complexity increases, and on algorithms to sort, access, and manipulate this data. Variables create data abstractions, as they can represent a set of possible values or a group of related values.</p>	<p>Along with several smaller programming assignments within Unit 5, the unit project, Air Quality Analyzer, allows students to demonstrate mastery of the Variables Big Idea by analyzing real-world data stored in each of the three data structures: 1D array, ArrayList, and 2D array.</p>
<p>Big Idea 3: CONTROL (CON) Doing things in order, making decisions, and doing the same process multiple times are represented in code by using control structures and specifying the order in which instructions are executed. Programmers need to think algorithmically to define and interpret processes that are used in a program.</p>	<p>Students will have many opportunities to demonstrate the use of control structures throughout the units. Some example activities:</p> <ul style="list-style-type: none"> • In Unit 2, students write a program using conditionals and nested <i>if</i> statements to create an interactive story using user input. • In Unit 3, students create iterative algorithms such as reversing all the characters in a word and taking the average of all digits in a number.
<p>Big Idea 4: IMPACT OF COMPUTING (IOC) Computers and computing have revolutionized our lives. To use computing safely and responsibly, we need to be aware of privacy, security, and ethical issues. As programmers, we need to understand how our programs will be used and be responsible for the consequences.</p>	<p>In Unit 1, students research and report on computing innovations, then discuss as a class the legal issues and intellectual property concerns surrounding the development and creation of programs and software.</p> <p>In Unit 4, students discuss the importance of system reliability and how programmers should make every effort to maximize system reliability and how using classes and objects (object-oriented programming) can help with this.</p> <p>In Unit 5, students discuss the ethical and security issues of gathering and storing personal information such as names and birthdates.</p>

Unit Guides

Curriculum Units [CR 2]		
<p>Unit 1: Introductions Are in Order Unit 1 establishes the foundation for the curriculum by introducing the AP Computer Science A course, the Java programming language, and the course narrative.</p>	<p>Big Ideas: Modularity Control Impact of Computing</p> <p>Enduring Understandings: MOD-1, MOD-2, CON-2, IOC-1</p>	<p>Computational Thinking Practices: </p> <p>Aligned to College Board CED Units: None – this unit is an introduction to the UTeach CS A curriculum narrative and using a text-based programming language.</p>
<p>Unit 2: Primitive Control This unit introduces object-oriented programming, primitive data types, mathematical expressions and operators, Boolean expressions, conditionals, using objects, and the Math class.</p>	<p>Big Ideas: Modularity Variables Control</p> <p>Enduring Understandings: MOD-1, VAR-1, CON-1, CON-2</p>	<p>Computational Thinking Practices: </p> <p>Aligned to College Board CED Units: 1, 2, and 3</p>
<p>Unit 3: Strings and Iteration Unit 3 concentrates on the String class and the String class methods. This unit introduces many of the most common string manipulating algorithms as well as iteration and the Java control structures <i>while loop</i> and <i>for loop</i> that allow for the use of repetition in programs.</p>	<p>Big Ideas: Modularity Variables Control</p> <p>Enduring Understandings: MOD-1, VAR-1, CON-2</p>	<p>Computational Thinking Practices: </p> <p>Aligned to College Board CED Units: 2 and 4</p>
<p>Unit 4: Objects, Classes, and Methods Unit 4 takes a deep dive into Object-Oriented Programming by introducing all the components of a class including private instance variables, constructors, accessor methods, mutator methods, and helper methods. This unit also covers accessibility and the different categories of methods from static and non-static to void and non-void.</p>	<p>Big Ideas: Modularity Variables Impact of Computing</p> <p>Enduring Understandings: MOD-1, MOD-2, MOD-3, VAR-1, IOC-1</p>	<p>Computational Thinking Practices: </p> <p>Aligned to College Board CED Unit: 5</p>

<p>Unit 5: Arrays, ArrayLists, and 2D Arrays Unit 5 introduces three different data structures used to store data in Java – the array, ArrayList, and 2D array – and the syntax and traversal methods for the most common algorithms for processing data.</p>	<p>Big Ideas: Variables Control Impact of Computing</p> <p>Enduring Understandings: VAR-1, VAR-2, CON-2, IOC-1</p>	<p>Computational Thinking Practices: </p> <p>Aligned to College Board CED Units: 6, 7, and 8</p>
<p>Unit 6: Inheritance Unit 6 takes a comprehensive look at two relationships between classes in Java: inheritance and composition. Topics covered in depth are superclasses and subclasses, constructors of the subclasses, overriding methods, composition, the Object class, and polymorphism.</p>	<p>Big Ideas: Modularity</p> <p>Enduring Understandings: MOD-2, MOD-3</p>	<p>Computational Thinking Practices: </p> <p>Aligned to College Board CED Unit: 9</p>
<p>Unit 7: Searching, Sorting, and Recursion This unit covers different search algorithms, the selection sort and insertion sort, Big O, recursion, and recursive search and sort algorithms.</p>	<p>Big Ideas: Control</p> <p>Enduring Understanding: CON-2</p>	<p>Computational Thinking Practices: </p> <p>Aligned to College Board CED Units: 6, 7, and 10</p>

NOTES: The following topics are taught throughout the curriculum as appropriate:

- Exception (CED Units 2, 6, 8)
- Ethical and Social Implications (CED Units 5, 7)

Unit 1: Introductions Are in Order

Unit 1 is designed to be an introduction to text-based programming using a special Java class for graphical programming using turtles. Throughout Unit 1, students master basic coding tools and concepts, such as algorithms and abstraction, which can be easily applied in later units.

The Big Ideas of Modularity, Control, and Impact of Computing are all covered in Unit 1.

These are the Enduring Understandings covered in this unit:

- 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.
- 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.
- 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.
- IOC-1: While programs are typically designed to achieve a specific purpose, they may have unintended consequences.

Unit 1 Topics

Over the course of this unit and the unit project, students will learn to:

- Collaborate with classmates and establish group norms
- Write code to draw graphics using Java
- Write an algorithm
- Implement an algorithm in Java
- Debug a program through error detection and testing
- Use abstraction by incorporating built-in methods from a Java Turtle class to draw shapes

Unit 1 Schedule (14 Days)

Assignment #	Topic	Learning Objectives	Essential Knowledge Statements
1.1	Avatar Creator Project (1 of 5)	MOD-1.B, MOD-1.D, MOD-1.E, MOD-2.C, CON-2.D	MOD-1.B.1, MOD-1.B.2, MOD-1.D.1, MOD-1.D.3, MOD-1.E.1, MOD-1.E.2, MOD-1.E.6, MOD-2.C.1
1.2	Introduction to Computer Science	IOC-1.A	IOC-1.A.2, IOC-1.A.3
1.3	Introduction to AP CS A	None	None
1.4	Introduction to Java Graphics	MOD-1.B, MOD-1.D, MOD-1.E, MOD-2.C	MOD-1.B.1, MOD-1.B.2, MOD-1.D.1, MOD-1.D.3, MOD-1.E.1, MOD-1.E.6, MOD-2.C.1
1.5	Algorithms	CON-2.D	None
1.6	Abstraction	MOD-1.E	MOD-1.E.2
1.1	Avatar Creator Project (2–4 of 5)	MOD-1.B, MOD-1.D, MOD-1.E, MOD-2.C, CON-2.D	MOD-1.B.1, MOD-1.B.2, MOD-1.D.1, MOD-1.D.3, MOD-1.E.1, MOD-1.E.2, MOD-1.E.6, MOD-2.C.1
1.7	Unit 1 Review	None	None
1.8	Unit 1 Test	None	None
1.1	Project Presentations (5 of 5)	None	None

Unit 2: Primitive Control

Unit 2 covers the College Board Units 1–3. Over the course of this unit, object-oriented programming, a style of programming that uses classes and objects, is introduced. Also, primitive data types in Java, mathematical expressions operators, Boolean expressions, conditionals, using objects, and a few of the Math class methods are studied.

The Big Ideas covered in this unit are Modularity, Variables, and Control. These are the Enduring Understandings covered in this unit:

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

These are the Computational Thinking Skills addressed:

- 1.A Determine an appropriate program design to solve a problem or accomplish a task.
- 1.B Determine code that would be used to complete code segments.
- 1.C Determine code that would be used to interact with completed program code.
- 2.A Apply the meaning of specific operators.
- 2.B Determine the result or output based on statement execution order in a code segment without method calls (other than output).
- 2.C Determine the result or output based on the statement execution order in a code segment containing method calls.
- 3.A Write program code to create objects of a class and call methods.
- 3.C Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.
- 4.A Use test cases to find errors or validate results.
- 4.B Identify errors in program code.
- 4.C Determine if two or more code segments yield equivalent results.
- 5.A Describe the behavior of a given segment of program code.
- 5.B Explain why a code segment will not compile or work as intended.

Unit 2 Topics

Our Unit 2 covers the College Board’s Units 1, 2, and 3. Over the course of this unit and the unit project, the following topics will be covered, and teachers may use these in the AP Classroom to filter questions for formative and summative assessments:

- 1.1 Why Programming? Why Java?
- 1.2 Variables and Data Types
- 1.3 Expressions and Assignment Statements
- 1.4 Compound Assignment Operators
- 1.5 Casting and Ranges of Variables
- 2.1 Objects: Instances of Classes
- 2.2 Creating and Storing Objects (Instantiation)
- 2.3 Calling a Void Method
- 2.4 Calling a Void Method with Parameters
- 2.5 Calling a Non-void Method
- 2.6 `String` Objects: Concatenation, Literals, and More
- 2.7 `String` Methods
- 2.8 `Wrapper` Classes: Integer and Double
- 2.9 Using the `Math` Class
- 3.1 Boolean Expressions
- 3.2 `if` Statements and Control Flow
- 3.3 `if-else` Statements
- 3.4 `else-if` Statements
- 3.5 Compound Boolean Expressions
- 3.6 Equivalent Boolean Expressions
- 3.7 Comparing Objects

Unit 2 Schedule (25 Days)

Assignment #	Topic	Learning Objectives	Essential Knowledge Statements
2.1	Resource Finder Project (1 of 5)	MOD-1.A, VAR-1.A	MOD-1.A.1, MOD-1.A.2, VAR-1.A.1
2.2	Variable and Data Types	MOD-1.A, VAR-1.B, VAR-1.C	MOD-1.A.1, MOD-1.A.2, VAR-1.A.1, VAR-1.B.1, VAR-1.B.2, VAR-1.B.3, VAR-1.C.1, VAR-1.C.2, VAR-1.C.3, VAR-1.C.4
2.3	Expressions and Assignment Statements	CON-1.A, CON-1.B	CON-1.A.1, CON-1.A.2, CON-1.A.3, CON-1.A.4, CON-1.A.5, CON-1.A.6, CON-1.A.6, CON-1.A.7, CON-1.A.8, CON-1.B.1, CON-1.B.2, CON-1.B.3, CON-1.B.4, CON-1.B.5
2.4	Compound Statements Coding Assignment	CON-1.A, CON-1.B	CON-1.A.6, CON-1.B.4
2.5	Casting and Range of Variables	CON-1.C	CON-1.C.1, CON-1.C.2, CON-1.C.3, CON-1.C.4, CON-1.C.5, CON-1.C.6
2.1	Resource Finder Project (2 of 5)	MOD-1.A, VAR-1.A	MOD-1.A.1, MOD-1.A.2, VAR-1.A.1
2.6	Using Objects	MOD-1.B, MOD-1.C, MOD-1.D, VAR-1.D	MOD-1.B.1, MOD-1.B.2, MOD-1.C.1, MOD-1.C.2, MOD-1.C.3, MOD-1.C.5, MOD-1.C.6, MOD-1.D.1, MOD-1.D.2, MOD-1.D.3, MOD-1.D.4, VAR-1.D.1, VAR-1.D.2
2.7	Scanner Class	MOD-1.E, MOD-1.F, MOD-1.G, VAR-1.D	MOD-1.E.1, MOD-1.E.2, MOD-1.E.3, MOD-1.E.4, MOD-1.E.5, MOD-1.E.6, MOD-1.E.7, MOD-1.E.8, MOD-1.F.1, MOD-1.F.2, MOD-1.F.3, MOD-1.G.1, VAR-1.D.1, VAR-1.D.2
2.8	String Objects & Methods	VAR-1.E	VAR-1.E.1, VAR-1.E.2, VAR-1.E.6, VAR-1.E.7, VAR-1.E.8, VAR-1.E.9, VAR-1.E.12

2.9	Wrapper Classes	VAR-1.F	VAR-1.F.1, VAR-1.F.2, VAR-1.F.3, VAR-1.F.4, VAR-1.F.5, VAR-1.F.6, VAR-1.F.7
2.10	Math Class	CON-1.D, MOD-1.H	CON-1.D.1, CON-1.D.2, CON-1.D.3, CON-1.D.4, MOD-1.H.1
2.11	Boolean Expression	CON-1.E, CON-1.F, CON-1.G	CON-1.E.1, CON-1.E.2, CON-1.E.3, CON-1.F.1, CON-1.F.2, CON-1.F.3, CON-1.G.1, CON-1.G.2, CON-1.G.3
2.12	DeMorgan's Law	CON-1.E, CON-1.F, CON-1.G	CON-1.E.1, CON-1.E.2, CON-1.E.3, CON-1.F.1, CON-1.F.2, CON-1.F.3, CON-1.G.1, CON-1.G.2, CON-1.G.3
2.13	Conditional Statements	CON-2.A, CON-2.B	CON-2.A.1, CON-2.A.2, CON-2.A.3, CON-2.A.4, CON-2.A.5, CON-2.B.1
2.14	Conditional Statements Coding Assignments	CON-2.A, CON-2.B	None
2.15	Conditional Quiz	CON-2.A	None
2.1	Resource Finder Project (3 of 5)	MOD-1.A, VAR-1.A	MOD-1.A.1, MOD-1.A.2, VAR-1.A.1
2.16	Comparing Objects	CON-1.H	CON-1.H.1, CON-1.H.2, CON-1.H.3, CON-1.H.4
2.1	Resource Finder Project (4 of 5)	MOD-1.A, VAR-1.A	MOD-1.A.1, MOD-1.A.2, VAR-1.A.1
2.17	Unit 2 Review	None	None
2.18	Unit 2 Test	None	None
2.1	Project Presentations (5 of 5)	None	None

Unit 3: Strings and Iteration

Unit 3 concentrates on the String class and the String class methods that are tested on the AP CS A exam. Manipulating strings is a very common task in programming and students will learn many of the most common string manipulating algorithms. This unit also introduces iteration, and the Java control structures that allow for the use of repetition in programs.

The Big Ideas covered in this unit are Modularity, Variables, and Control. These are the Enduring Understandings covered in this unit:

- 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.
- 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.
- 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.

These are the Computational Thinking Skills addressed:

- 1.B Determine code that would be used to complete code segments.
- 1.C Determine code that would be used to interact with completed program code.
- 2.A Apply the meaning of specific operators.
- 2.B Determine the result or output based on statement execution order in a code segment without method calls (other than output).
- 2.C Determine the result or output based on the statement execution order in a code segment containing method calls.
- 3.A Write program code to create objects of a class and call methods.
- 3.C Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.
- 4.C Determine if two or more code segments yield equivalent results.
- 5.A Describe the behavior of a given segment of program code.
- 5.C Explain how the result of program code changes, given a change of the initial code.

Unit 3 Topics

Our Unit 3 covers the College Board Unit 4 and a review of Strings. Over the course of this unit and the unit project, the following topics will be covered, and teachers may use these in the AP Classroom to filter questions for formative and summative assessments:

- 2.6 String Objects: Concatenation, Literals, and More
- 2.7 String Methods
- 4.1 while Loops
- 4.2 for Loops
- 4.3 Developing Algorithms using Strings
- 4.4 Nested Iteration
- 4.5 Informal Code Analysis

Unit 3 Schedule (19 Days)

Assignment #	Topic	Learning Objectives	Essential Knowledge Statements
3.1	Language Interpreter Project (1 of 6)	VAR-1.E, MOD-1.G	VAR-1.E.1, VAR-1.E.3, VAR-1.E.10, VAR-1.E.11, VAR-1.E.12, MOD-1.G.1
3.2	Introduction to Strings	VAR-1.A, VAR-1.E, MOD-1.G	VAR-1.A.1, VAR-1.E.1, VAR-1.E.2, VAR-1.E.3, VAR-1.E.4, VAR-1.E.5, VAR-1.E.6, VAR-1.E.7, VAR-1.E.8, VAR-1.E.9, VAR-1.E.10, VAR-1.E.11, VAR-1.E.12, VAR-1.E.13, MOD-1.G.1
3.3	Palindrome Checker Coding Assignment	VAR-1.E, MOD-1.G	None
3.1	Language Interpreter Project (2 of 6)	VAR-1.E, MOD-1.G	VAR-1.E.1, VAR-1.E.3, VAR-1.E.10, VAR-1.E.11, VAR-1.E.12, MOD-1.G.1
3.4	Iteration – For Loops	CON-2.D, CON-2.E	CON-2.D.1, CON-2.E.1, CON-2.E.2, CON-2.E.3, CON-2.E.5
3.5	Reverse Word Coding Assignment	CON-2.D, CON-2.E	None
3.6	Iteration – While Loops	CON-2.C, CON-2.D, CON-2.E	CON-2.C.1, CON-2.C.2, CON-2.C.3, CON-2.C.4, CON-2.C.5, CON-2.D.1, CON-2.D.2, CON-2.E.4

3.7	Tracing Code	CON-2.C, CON-2.E	None
3.8	Nested Iteration	CON-2.G	CON-2.G.1, CON-2.G.2
3.9	“Backwards” Triangle Coding Assignment	CON-2.G	None
3.1	Language Interpreter Project (3 of 6)	VAR-1.E, MOD-1.G	VAR-1.E.1, VAR-1.E.3, VAR-1.E.10, VAR-1.E.11, VAR-1.E.12, MOD-1.G.1
3.10	Strings Algorithms	CON-2.F, CON-2.H	CON-2.F.1, CON-2.H.1
3.11	More String Algorithms	CON-2.F	CON-2.F.1
3.12	Replace Nth Occurrence Coding Assignment	CON-2.F	None
3.1	Language Interpreter Project (4–5 of 6)	VAR-1.E, MOD-1.G	VAR-1.E.1, VAR-1.E.3, VAR-1.E.10, VAR-1.E.11, VAR-1.E.12, MOD-1.G.1
3.13	Unit 3 Review	None	None
3.14	Unit 3 Test	None	None
3.1	Project Presentations (6 of 6)	None	None

Unit 4: Object, Classes, and Methods

Unit 4 takes a deep dive into Object-Oriented Programming by introducing all the components of a class including private instance variables, constructors, accessor methods, mutator methods, and helper methods. This unit also covers private vs. public access to data and methods, the different categories of methods from static and non-static to void and non-void. Within the unit, students will create entire classes and then instantiate objects of that class.

The Big Ideas covered in this unit are Modularity, Variables, and Impact of Computing. These are the Enduring Understandings covered in this unit:

- 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.
- 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.
- 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.
- 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.
- IOC-1: While programs are typically designed to achieve a specific purpose, they may have unintended consequences.

These are the Computational Thinking Skills addressed:

- 1.A Determine an appropriate program design to solve a problem or accomplish a task.
- 1.B Determine code that would be used to complete code segments.
- 1.C Determine code that would be used to interact with completed program code.
- 2.C Determine the result or output based on the statement execution order in a code segment containing method calls.
- 3.B Write program code to define a new type by creating a class.
- 4.B Identify errors in program code.
- 5.A Describe the behavior of a given segment of program code.
- 5.B Explain why a code segment will not compile or work as intended.
- 5.D Describe the initial conditions that must be met for a program segment to work as intended or described.

Unit 4 Topics

Our Unit 4 covers the College Board Unit 5. Over the course of this unit and the unit project, the following topics will be covered, and teachers may use these in the AP Classroom to filter questions for formative and summative assessments:

- 5.1 Anatomy of a Class
- 5.2 Constructors
- 5.3 Documentation with Comments
- 5.4 Accessor Methods
- 5.5 Mutator Methods
- 5.6 Writing Methods
- 5.7 Static Variables and Methods
- 5.8 Scope and Access
- 5.9 `this` Keyword
- 5.10 Ethical and Social Implications of Computing Systems

Unit 4 Schedule (20 Days)

Assignment #	Topic	Learning Objectives	Essential Knowledge Statements
4.1	Disease Diagnoser Project (1 of 5)	MOD-1.A, VAR-1.A	MOD-1.A.1, MOD-1.A.2, VAR-1.A.1
4.2	Object-Oriented Programming	MOD-1.B, MOD-2.D, MOD-2.E, MOD-2.F, MOD-3.A, VAR-1.G, IOC-1.A	MOD-1.B.1, MOD-1.B.2, MOD-2.D.1, MOD-2.D.2, MOD-2.D.3, MOD-2.D.4, MOD-2.D.5, MOD-2.D.6, MOD-2.D.7, MOD-2.E.1, MOD-2.E.2, MOD-2.F.1, MOD-2.F.2, MOD-2.F.3, MOD-2.F.4, MOD-2.F.5, MOD-2.F.6, MOD-3.A.1, MOD-3.A.2, MOD-3.A.3, MOD-3.A.4, VAR-1.G.1, VAR-1.G.2, VAR-1.G.3, VAR-1.G.4, IOC-1.A.1
4.3	The Object Superclass	MOD-3.E	MOD-3.E.1, MOD-3.E.2, MOD-3.E.3, MOD-3.E.4
4.4	Writing Classes	MOD-2.B, MOD-2.C, MOD-2.D, MOD-2.E, MOD-2.F	MOD-2.B.1, MOD-2.B.2, MOD-2.B.3, MOD-2.B.4, MOD-2.C.1, MOD-2.C.2, MOD-2.C.3, MOD-2.C.4, MOD-2.C.5, MOD-2.D.1,

			MOD-2.D.2, MOD-2.D.3, MOD-2.D.4, MOD-2.D.5, MOD-2.D.6, MOD-2.D.7, MOD-2.E.1, MOD-2.E.2, MOD-2.F.1, MOD-2.F.2, MOD-2.F.3, MOD-2.F.4, MOD-2.F.5, MOD-2.F.6
4.5	African Life Activity	MOD-2.B, MOD-2.C, MOD-2.D, MOD-2.E, MOD-2.F	MOD-2.B.1, MOD-2.B.2, MOD-2.B.3, MOD-2.B.4, MOD-2.C.1, MOD-2.C.2, MOD-2.C.3, MOD-2.C.4, MOD-2.C.5, MOD-2.D.1, MOD-2.D.2, MOD-2.D.3, MOD-2.D.4, MOD-2.D.5, MOD-2.D.6, MOD-2.D.7, MOD-2.E.1, MOD-2.E.2, MOD-2.F.1, MOD-2.F.2, MOD-2.F.3, MOD-2.F.4, MOD-2.F.5, MOD-2.F.6
4.6	Creating Instances of Classes	MOD-2.B	MOD-2.B.1, MOD-2.B.2, MOD-2.B.3, MOD-2.B.4, MOD-2.B.5
4.7	Accessibility	MOD-2.A, MOD-3.A	MOD-2.A.1, MOD-2.A.2, MOD-2.A.3, MOD-2.A.4, MOD-2.A.5, MOD-2.A.6, MOD-3.A.1, MOD-2.A.2, MOD-3.A.3, MOD-3.A.4
4.8	Static Variables and Methods	MOD-1.H, MOD-2.G, MOD-2.H	MOD-1.H.1, MOD-2.G.1, MOD-2.G.2, MOD-2.G.3, MOD-2.G.4, MOD-2.G.5, MOD-2.H.1, MOD-2.H.2, MOD-2.H.3
4.9	this and super	MOD-3.B, VAR-1.G, VAR-1.H	MOD-3.B.1, MOD-3.B.2, MOD-3.B.4, MOD-3.B.14, MOD-3.B.15, VAR-1.G.1, VAR-1.G.2, VAR-1.G.3, VAR-1.G.4, VAR-1.H.1, VAR-1.H.2,
4.1	Disease Diagnoser Project (2 of 5)	MOD-1.A, VAR-1.A	MOD-1.A.1, MOD-1.A.2, VAR-1.A.1
4.10	Void Methods	MOD-1.E, MOD-2.E	MOD-1.E.1, MOD-1.E.2, MOD-1.E.3, MOD-1.E.4, MOD-1.E.5, MOD-1.E.6, MOD-1.E.7, MOD-2.E.1

4.11	Non-void Methods	MOD.1.G, MOD-2.D, MOD-2.F	MOD-1.G.1, MOD-2.D.2, MOD-2.D.3, MOD-2.D.4, MOD-2.D.5, MOD-2.F.1, MOD-2.F.2, MOD-2.F.3, MOD-2.F.4
4.1	Disease Diagnoser Project (3 of 5)	MOD-1.A, VAR-1.A	MOD-1.A.1, MOD-1.A.2, VAR-1.A.1
4.12	Overloaded Methods	MOD-1.C, MOD-1.F	MOD-1.C.4, MOD-1.F.3
4.1	Disease Diagnoser Project (4 of 5)	MOD-1.A, VAR-1.A	MOD-1.A.1, MOD-1.A.2, VAR-1.A.1
4.13	Unit 4 Review	None	None
4.14	Unit 4 Test	None	None
4.1	Project Presentations (5 of 5)	None	None

Unit 5: Arrays, ArrayLists, and 2D Arrays

Unit 5 focuses on three different data structures used to store data in Java: the array, ArrayList, and 2D array. Arrays are static, one-dimensional data structures. ArrayList is a built-in Java class that stores data in a dynamic list, and 2D arrays store data in a two-dimensional structure like a table or spreadsheet. Topics include the syntax for creation and access and traversal methods for each data structure and the most common algorithms for processing data.

The Big Ideas covered in this unit are Variables, Control, and Impact of Computing. These are the Enduring Understandings covered in this unit:

- 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.
- 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.
- 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.
- 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.
- 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.
- 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.
- IOC-1: While programs are typically designed to achieve a specific purpose, they may have unintended consequences.

These are the Computational Thinking Skills addressed:

- 1.B Determine code that would be used to complete code segments.
- 1.C Determine code that would be used to interact with completed program code.
- 2.B Determine the result or output based on statement execution order in a code segment without method calls (other than output).
- 2.C Determine the result or output based on the statement execution order in a code segment containing method calls.
- 3.D Write program code to create, traverse, and manipulate elements in 1D array or ArrayList objects.

- 3.E Write program code to create, traverse, and manipulate elements in 2D array objects.
- 4.A Use test cases to find errors in program code.
- 4.B Identify errors in program code.
- 4.C Determine if two or more code segments yield equivalent results.
- 5.D Describe the initial conditions that must be met for a program segment to work as intended or described.

Unit 5 Topics

Our Unit 5 covers the College Board Units 6, 7, and 8. Over the course of this unit and the unit project, the following topics will be covered, and teachers may use these in the AP Classroom to filter questions for formative and summative assessments:

- 6.1 Array Creation and Access
- 6.2 Traversing Arrays
- 6.3 Enhanced `for` Loop for Arrays
- 6.4 Developing Algorithms using Arrays
- 7.1 Introduction to `ArrayList`
- 7.2 `ArrayList` Methods
- 7.3 Traversing `ArrayLists`
- 7.4 Developing Algorithms using `ArrayLists`
- 7.7 Ethical Issues Around Data Collection
- 8.1 2D Arrays
- 8.2 Traversing 2D Arrays

Unit 5 Schedule (22 Days)

Assignment #	Topic	Learning Objectives	Essential Knowledge Statements
5.1	Air Quality Analyzer Project (1 of 6)	VAR-1.F, VAR-2.A, VAR-2.B, VAR-2.C, VAR-2.D, VAR-2.E, VAR-2.F, VAR-2.G, CON-2.D, CON-2.I, CON-2.J, CON-2.N	VAR-1.F.1, VAR-1.F.2, VAR-1.F.3, VAR-1.F.4, VAR-1.F.5, VAR-1.F.6, VAR-1.F.7, VAR-2.A.1, VAR-2.A.2, VAR-2.A.3, VAR-2.A.4, VAR-2.A.5, VAR-2.A.6, VAR-2.A.7, VAR-2.B.1, VAR-2.B.2, VAR-2.B.3, VAR-2.C.1, VAR-2.C.2, VAR-2.C.3, VAR-2.C.4, VAR-2.D.1, VAR-2.D.2, VAR-2.D.3, VAR-2.D.4, VAR-2.D.5, VAR-2.D.6, VAR-2.D.7, VAR-2.E.1, VAR-2.E.2, VAR-2.E.3, VAR-2.E.4, VAR-2.F.1, VAR-2.F.2, VAR-2.F.3, VAR-2.F.4, VAR-2.F.5, VAR-2.G.1, VAR-2.G.2, VAR-2.G.3, CON-2.D.2, CON-2.I.1, CON-2.I.2, CON-2.J.1, CON-2.J.2, CON-2.N.2
5.2	Array Creation and Access	IOC-1.B, VAR-2.A	IOC-1.B.1, IOC-1.B.2, VAR-2.A.1, VAR-2.A.2, VAR-2.A.3, VAR-2.A.4, VAR-2.A.5, VAR-2.A.6, VAR-2.A.7
5.3	Managing People as Data	IOC-1.B, VAR-2.A	IOC-1.B.1, IOC-1.B.2, VAR-2.A.1, VAR-2.A.2, VAR-2.A.3, VAR-2.A.4, VAR-2.A.5, VAR-2.A.6, VAR-2.A.7
5.4	Traversing Arrays	VAR-2.B	VAR-2.B.1, VAR-2.B.2, VAR-2.B.3
5.5	Enhanced For Loop for Arrays	VAR-2.C	VAR-2.C.1, VAR-2.C.2, VAR-2.C.3, VAR-2.C.4
5.6	Developing Algorithms Using Arrays	CON-2.I	CON-2.I.1, CON-2.I.2
5.1	Air Quality Analyzer Project (2 of 6)	VAR-1.F, VAR-2.A, VAR-2.B, VAR-2.C,	VAR-1.F.1, VAR-1.F.2, VAR-1.F.3, VAR-1.F.4,

		VAR-2.D , VAR-2.E, VAR-2.F, VAR-2.G, CON-2.D, CON-2.I, CON-2.J, CON-2.N	VAR-1.F.5, VAR-1.F.6, VAR-1.F.7, VAR-2.A.1, VAR-2.A.2, VAR-2.A.3, VAR-2.A.4, VAR-2.A.5, VAR-2.A.6, VAR-2.A.7, VAR-2.B.1, VAR-2.B.2, VAR-2.B.3, VAR-2.C.1, VAR-2.C.2, VAR-2.C.3, VAR-2.C.4, VAR-2.D.1, VAR-2.D.2, VAR-2.D.3, VAR-2.D.4, VAR-2.D.5, VAR-2.D.6, VAR-2.D.7, VAR-2.E.1, VAR-2.E.2, VAR-2.E.3, VAR-2.E.4, VAR-2.F.1, VAR-2.F.2, VAR-2.F.3, VAR-2.F.4, VAR-2.F.5, VAR-2.G.1, VAR-2.G.2, VAR-2.G.3, CON-2.D.2, CON-2.I.1, CON-2.I.2, CON-2.J.1, CON-2.J.2, CON-2.N.2
5.7	Introduction to ArrayList	VAR-2.D	VAR-2.D.1, VAR-2.D.2, VAR-2.D.3, VAR-2.D.4, VAR-2.D.5
5.8	ArrayList Methods	VAR-2.D	VAR-2.D.6, VAR-2.D.7
5.9	Wrapper Classes: Integer and Double	VAR-1.F	VAR-1.F.1, VAR-1.F.2, VAR-1.F.3, VAR-1.F.4, VAR-1.F.5, VAR-1.F.6, VAR-1.F.7
5.10	Traversing ArrayLists	VAR-2.E	VAR-2.E.1, VAR-2.E.2, VAR-2.E.3, VAR-2.E.4
5.11	Developing Algorithms Using ArrayLists	CON-2.J	CON-2.J.1, CON-2.J.2
5.1	Air Quality Analyzer Project (3 of 6)	VAR-1.F, VAR-2.A, VAR-2.B, VAR-2.C, VAR-2.D , VAR-2.E, VAR-2.F, VAR-2.G, CON-2.D, CON-2.I, CON-2.J, CON-2.N	VAR-1.F.1, VAR-1.F.2, VAR-1.F.3, VAR-1.F.4, VAR-1.F.5, VAR-1.F.6, VAR-1.F.7, VAR-2.A.1, VAR-2.A.2, VAR-2.A.3, VAR-2.A.4, VAR-2.A.5, VAR-2.A.6, VAR-2.A.7, VAR-2.B.1, VAR-2.B.2, VAR-2.B.3, VAR-2.C.1, VAR-2.C.2, VAR-2.C.3, VAR-2.C.4, VAR-2.D.1, VAR-2.D.2, VAR-2.D.3, VAR-2.D.4, VAR-2.D.5,

			VAR-2.D.6, VAR-2.D.7, VAR-2.E.1, VAR-2.E.2, VAR-2.E.3, VAR-2.E.4, VAR-2.F.1, VAR-2.F.2, VAR-2.F.3, VAR-2.F.4, VAR-2.F.5, VAR-2.G.1, VAR-2.G.2, VAR-2.G.3, CON-2.D.2, CON-2.I.1, CON-2.I.2, CON-2.J.1, CON-2.J.2, CON-2.N.2
5.12	2D Arrays	VAR-2.F, VAR-2.G, CON-2.N	VAR-2.F.1, VAR-2.F.2, VAR-2.F.3, VAR-2.F.4, VAR-2.F.5, VAR-2.G.1, VAR-2.G.2, VAR-2.G.3, CON-2.N.2
5.13	Developing Algorithms Using 2D Arrays	VAR-2.F, VAR-2.G, CON-2.N	VAR-2.F.1, VAR-2.F.2, VAR-2.F.3, VAR-2.F.4, VAR-2.F.5, VAR-2.G.1, VAR-2.G.2, VAR-2.G.3, CON-2.N.2
5.1	Air Quality Analyzer Project (4–5 of 6)	VAR-1.F, VAR-2.A, VAR-2.B, VAR-2.C, VAR-2.D, VAR-2.E, VAR-2.F, VAR-2.G, CON-2.D, CON-2.I, CON-2.J, CON-2.N	VAR-1.F.1, VAR-1.F.2, VAR-1.F.3, VAR-1.F.4, VAR-1.F.5, VAR-1.F.6, VAR-1.F.7, VAR-2.A.1, VAR-2.A.2, VAR-2.A.3, VAR-2.A.4, VAR-2.A.5, VAR-2.A.6, VAR-2.A.7, VAR-2.B.1, VAR-2.B.2, VAR-2.B.3, VAR-2.C.1, VAR-2.C.2, VAR-2.C.3, VAR-2.C.4, VAR-2.D.1, VAR-2.D.2, VAR-2.D.3, VAR-2.D.4, VAR-2.D.5, VAR-2.D.6, VAR-2.D.7, VAR-2.E.1, VAR-2.E.2, VAR-2.E.3, VAR-2.E.4, VAR-2.F.1, VAR-2.F.2, VAR-2.F.3, VAR-2.F.4, VAR-2.F.5, VAR-2.G.1, VAR-2.G.2, VAR-2.G.3, CON-2.D.2, CON-2.I.1, CON-2.I.2, CON-2.J.1, CON-2.J.2, CON-2.N.2
5.14	Unit 5 Review	None	None
5.15	Unit 5 Test	None	None
5.1	Project Presentations (6 of 6)	None	None

Unit 6: Inheritance

Unit 6 takes a comprehensive look at two relationships between classes in Java: inheritance and composition. Class interaction is a fundamental concept of OOP — building a new class from an existing class or using features of one class in another class leads to reliability and reliability is the main goal of OOP. Topics covered in depth are superclasses and subclasses, constructors of the subclasses, overriding methods, composition, the Object class, and polymorphism.

The Big Idea covered in this unit is Modularity. These are the Enduring Understandings covered in this unit:

- 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.
- 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.

These are the Computational Thinking Skills addressed:

- 1.A Determine an appropriate program design to solve a problem or accomplish a task.
- 1.C Determine code that would be used to interact with completed program code.
- 3.A Write program code to create objects of a class and call methods.
- 3.B Write program code to define a new type by creating a class.
- 5.A Describe the behavior of a given segment of program code.
- 5.B Explain why a code segment will not compile or work as intended.
- 5.D Describe the initial conditions that must be met for a program segment to work as intended or described.

Unit 6 Topics

Our Unit 6 covers the College Board Unit 9. Over the course of this unit and the unit project, the following topics will be covered, and teachers may use these in the AP Classroom to filter questions for formative and summative assessments:

- 9.1 Creating Superclasses and Subclasses
- 9.2 Writing Constructors for Subclasses
- 9.3 Overriding Methods
- 9.4 `super` Keyword
- 9.5 Creating References using Inheritance Hierarchies
- 9.6 Polymorphism
- 9.7 `Object` Superclass

Unit 6 Schedule (19 Days)

Assignment #	Topic	Learning Objectives	Essential Knowledge Statements
6.1	Hospital Locator Project (1 of 6)	MOD-3.B, MOD-3.C, MOD-3.D, MOD-3.E	MOD-3.B.1, MOD-3.B.2, MOD-3.B.3, MOD-3.B.4, MOD-3.B.5, MOD-3.B.6, MOD-3.B.7, MOD-3.B.8, MOD-3.B.9, MOD-3.B.10, MOD-3.B.11, MOD-3.B.12, MOD-3.B.13, MOD-3.B.14, MOD-3.B.15, MOD-3.C.1, MOD-3.C.2, MOD-3.C.3, MOD-3.C.4, MOD-3.D.1, MOD-3.D.2, MOD-3.D.3, MOD-3.E.3, MOD-3.E.4
6.2	Class Hierarchies	MOD-3.B, MOD-3.E	MOD-3.B.1, MOD-3.B.2, MOD-3.E.3, MOD-3.E.4
6.3	Warriors & Wizards Coding Assignment	MOD-3.B, MOD-3.E	MOD-3.B.1, MOD-3.B.2, MOD-3.E.3, MOD-3.E.4
6.4	Extending Classes	MOD-3.B	MOD-3.B.1, MOD-3.B.2, MOD-3.B.4, MOD-3.B.11, MOD-3.B.12, MOD-3.B.13
6.5	The “is-a” Inheritance Relationship	MOD-3.B	MOD-3.B.3
6.6	The “has-a” Composition Relationship	MOD-2.B	MOD-2.B.1
6.7	Overriding Methods	MOD-3.B	MOD-3.B.10, MOD-3.B.11, MOD-3.B.12, MOD-3.B.13, MOD-3.B.14, MOD-3.B.15
6.1	Hospital Locator Project (2 of 6)	MOD-3.B, MOD-3.C, MOD-3.D, MOD-3.E	MOD-3.B.1, MOD-3.B.2, MOD-3.B.3, MOD-3.B.4, MOD-3.B.5, MOD-3.B.6, MOD-3.B.7, MOD-3.B.8, MOD-3.B.9, MOD-3.B.10, MOD-3.B.11, MOD-3.B.12, MOD-3.B.13, MOD-3.B.14, MOD-3.B.15, MOD-3.C.1, MOD-3.C.2, MOD-3.C.3, MOD-3.C.4, MOD-3.D.1, MOD-3.D.2, MOD-3.D.3, MOD-3.E.3, MOD-3.E.4

6.8	Writing Constructors for Subclasses	MOD-3.B	MOD-3.B.5, MOD-3.B.6, MOD-3.B.7, MOD-3.B.8, MOD-3.B.9, MOD-3.B.14, MOD-3.B.15
6.9	Creating Buildings with Java	MOD-3.B	MOD-3.B.5, MOD-3.B.6, MOD-3.B.7, MOD-3.B.8, MOD-3.B.9, MOD-3.B.14, MOD-3.B.15
6.10	Declaring Objects of Parent Type	MOD-3.C	MOD-3.C.1, MOD-3.C.2, MOD-3.C.3, MOD-3.C.4
6.1	Hospital Locator Project (3 of 6)	MOD-3.B, MOD-3.C, MOD-3.D, MOD-3.E	MOD-3.B.1, MOD-3.B.2, MOD-3.B.3, MOD-3.B.4, MOD-3.B.5, MOD-3.B.6, MOD-3.B.7, MOD-3.B.8, MOD-3.B.9, MOD-3.B.10, MOD-3.B.11, MOD-3.B.12, MOD-3.B.13, MOD-3.B.14, MOD-3.B.15, MOD-3.C.1, MOD-3.C.2, MOD-3.C.3, MOD-3.C.4, MOD-3.D.1, MOD-3.D.2, MOD-3.D.3, MOD-3.E.3, MOD-3.E.4
6.11	Polymorphism	MOD-3.D	MOD-3.D.1, MOD-3.D.2, MOD-3.D.3
6.1	Hospital Locator Project (4 & 5 of 6)	MOD-3.B, MOD-3.C, MOD-3.D, MOD-3.E	MOD-3.B.1, MOD-3.B.2, MOD-3.B.3, MOD-3.B.4, MOD-3.B.5, MOD-3.B.6, MOD-3.B.7, MOD-3.B.8, MOD-3.B.9, MOD-3.B.10, MOD-3.B.11, MOD-3.B.12, MOD-3.B.13, MOD-3.B.14, MOD-3.B.15, MOD-3.C.1, MOD-3.C.2, MOD-3.C.3, MOD-3.C.4, MOD-3.D.1, MOD-3.D.2, MOD-3.D.3, MOD-3.E.3, MOD-3.E.4
6.12	Unit 6 Review	None	None
6.13	Unit 6 Test	None	None
6.1	Project Presentations (6 of 6)	None	None

Unit 7: Searching, Sorting, and Recursion

Unit 7 covers different search algorithms, the selection sort and insertion sort, Big O, recursion, recursive search, and sort algorithms.

The Big Idea of Control is covered in Unit 7. These are the Enduring Understandings covered in this unit:

- 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.

These are the Computational Thinking Skills addressed:

- 1.B Determine code that would be used to complete code segments.
- 2.C Determine the result or output based on the statement execution order in a code segment containing method calls.
- 2.D Determine the number of times a code segment will execute.
- 3.D Write program code to create, traverse and manipulate elements in 1D array or ArrayList objects.
- 5.A Describe the behavior of a given segment of program code.
- 5.C Explain how the result of program code changes, given a change of the initial code.

Unit 7 Topics

Our Unit 7 covers the College Board Unit 10 and the searching and sorting topics from Unit 7. Over the course of this unit and the unit project, the following topics will be covered, and teachers may use these in the AP Classroom to filter questions for formative and summative assessments:

- 7.5 Searching
- 7.6 Sorting
- 10.1 Recursion
- 10.2 Recursive Searching and Sorting

Unit 7 Schedule (20 Days)

Assignment #	Topic	Learning Objectives	Essential Knowledge Statement
7.1	Data Decoder Project (1 of 6)	CON-2.H, CON-2.K, CON-2.L, CON-2.M	CON-2.H.1, CON-2.K.1, CON-2.K.2, CON-2.L.1, CON-2.M.1
7.2	Sequential and Iterative Binary Search	CON-2.K, CON-2.N, CON-2.P	CON-2.K.1, CON-2.K.2, CON-2.N.1, CON-2.P.1, CON-2.P.2, CON-2.P.3, CON-2.P.4
7.3	Selection Sort	CON-2.L, CON-2.M	CON-2.L.1, CON-2.M.1
7.4	Insertion Sort		
7.5	Create Your Own Sort		
7.1	Data Decoder Project (2–3 of 6)	CON-2.H, CON-2.K, CON-2.L, CON-2.M	CON-2.H.1, CON-2.K.1, CON-2.K.2, CON-2.L.1, CON-2.M.1
7.6	Compare Big O Informally	CON-2.H, CON-2.M	CON-2.H.1, CON-2.M.1
7.1	Data Decoder Project (4 of 6)	CON-2.H, CON-2.K, CON-2.L, CON-2.M	CON-2.H.1, CON-2.K.1, CON-2.K.2, CON-2.L.1, CON-2.M.1
7.7	Recursion Defined	CON-2.O	CON-2.O.1, CON-2.O.2, CON-2.O.3, CON-2.O.4, CON-2.O.5, CON-2.O.6
7.8	Recursive Methods		
7.9	Recursive Traversal Methods		
7.10	Binary Search	CON-2.P	CON-2.P.1, CON-2.P.2, CON-2.P.3, CON-2.P.4
7.11	Merge Sort	CON-2.Q	CON-2.Q.1
7.1	Data Decoder Project (5 of 6)	CON-2.H, CON-2.K, CON-2.L, CON-2.M	CON-2.H.1, CON-2.K.1, CON-2.K.2, CON-2.L.1, CON-2.M.1
7.12	Unit 7 Review	None	None
7.13	Unit 7 Test	None	None
7.1	Project Presentations (6 of 6)	None	None

Pedagogical Approaches

The UTeach AP Computer Science A curriculum uses Project-Based Learning (PBL) and Culturally Responsive Education (CRE) to better engage students in the learning process. By encouraging students to use critical thinking skills and challenging them to solve authentic and meaningful problems, PBL helps students to develop a deeper and more profound understanding of the power of computation in our everyday lives. Built using a student-centric approach, the curriculum leverages many important components of CRE including learning within the context of culture, encouraging the teacher to be a facilitator of learning, and mediating instruction within the culture. Teachers who are unfamiliar with the goals, methods, and techniques of PBL can learn more at the [Buck Institute for Education website](#). Likewise, more information about CRE can be found at the [Education Alliance website](#).

In teaching this curriculum, educators are encouraged to use the range of PBL and CRE techniques that have been incorporated into each unit, including narrative *anchoring videos*, overarching *unit projects*, clear *rubrics*, regular *benchmarks*, *scaffolding activities*, *final products*, and *reflection*. Used together in a coherent, unified manner that actively engages students in the educational process, PBL and CRE strategies can improve comprehension and retention and help students develop stronger problem-solving, critical-thinking, and group communication skills.

Instructional Sequencing

The year-long curriculum consists of seven instructional units (each approximately lasting four weeks) that have been carefully structured to guide students through an introduction to the Java programming language and the study of object-oriented programming, a style of programming that uses classes and objects mimicking the organization of the real world. The topics are sequenced so that each new unit builds on knowledge and skills from previous units, with a goal to prepare students to successfully complete the College Board AP Computer Science A exam in the spring.

Introduction

The first module of each instructional unit opens with an anchor video designed to introduce the driving questions, unit project, and key topics for the next few weeks of study. Students are expected to participate in small-group and/or whole-class discussion to identify areas of focus that will direct and drive learning throughout the unit.

Topic Lessons/Activities

Distributed throughout each unit, individual lessons, daily activities, programming assignments, and discussions allow students to explore and practice applying relevant skills and concepts in greater detail.

Project Workdays

Each of the projects has five or six dedicated work periods throughout each unit for students to work on the programming aspect of the project. Project workdays are not consecutive but instead are spread throughout the unit, strategically placed after relevant topics have been presented.

Assessments

In addition to informal, formative assessments throughout each unit, student learning and progress is also assessed at the end of each unit through a formal, summative assessment and evaluation of their independent and collaborative efforts on the unit project.

Formal assessments are modeled after the single-select, multiple-choice questions and the free response questions on the AP Computer Science A exam so that students can familiarize themselves with the scope, style, and timing of questions that they can expect to see on the AP exam in the spring.

Unit Projects

The curriculum is narrative-driven, with students following a diverse cast of characters who travel across the globe and solve real-world problems along the way. Students will come to know a cast of four main characters, Vee, Hinni, Ed, and Antonio, who take them on an adventure around the world. Hinni, Ed, and Antonio are helping Vee search for her mother, who is stationed at the McMurdo Station, a research facility in Antarctica, when disaster strikes. As the group of friends fly to each continent on their search, they take time out to help people along the way, solving problems using their computational thinking skills. The students in the classroom are actively involved with the characters as they represent a technical team of computer programmers who lend a helping hand in creating effective solutions to the authentic problems the group faces.

The opening module of each unit serves as a formal launch of the unit project, an overarching, product-oriented challenge for students to investigate, research, and solve over the course of the unit. The project launch starts with an anchor video that introduces the fundamental problem or challenge to be solved and is intended to spark the students' imaginations and inspire them to find a solution. Projects are designed to be collaborative with students working

together in groups and using the pair programming instructional strategy. These project workdays, along with smaller daily programming assignments, exceed the College Board Curricular Requirement 9, requiring students to spend at 20 hours in hands-on lab experiences [CR-9].

Unit Projects [CR 9]
<p>Unit 1: Avatar Creator Project Students design unique, personal avatars, using a special graphics class in Java, to represent themselves as a member of the technical team assisting the cast of characters in the curriculum narrative.</p>
<p>Unit 2: Resource Finder Project Students create a program using conditional statements to read and analyze data from different rock samples to identify an ideal area to rebuild a science research station in Antarctica.</p>
<p>Unit 3: Language Interpreter Project Students create a language interpreter program that collects a speech sample of an unknown indigenous language in South America, then uses iteration and String class methods to translate the new language into English.</p>
<p>Unit 4: Disease Diagnoser Project Students will design a program using classes and methods to diagnose different diseases based on patient symptoms in a clinic in South Africa.</p>
<p>Unit 5: Air Quality Analyzer Project Students write a program to store and analyze air quality data from cities in Sri Lanka using arrays, ArrayLists, and 2D arrays. After completing their analysis, students compare the use of the different data structures and decide if their program is scalable for a larger data set.</p>
<p>Unit 6: Hospital Locator Project Students reconstruct a compromised database of hospitals in Australia using superclass and subclass structures. Students will then use a clue to search for a specific hospital.</p>
<p>Unit 7: Data Decoder Project Students write a program that uses searching and sorting algorithms to analyze climate change data. They must optimize the algorithms for greater efficiency to create a presentation for the World Meteorological Organization in Switzerland.</p>

Rubrics

Each unit project is accompanied by a clearly defined rubric that specifies the set of expectations for student work throughout the unit, including an exhaustive list of assessment criteria for the artifacts that students will produce and detailed descriptors for each

performance level that a student might demonstrate. Teachers should provide students with the rubric at the start of the unit as part of the initial discussion immediately following the anchor video. Giving the students the rubric at the time of the project launch is critical for setting clear student expectations early in the research and learning process. Over the course of the unit, teachers should regularly refer to the goals and criteria of the rubric to ensure that students remain focused and on pace for meeting the stated requirements.

Benchmarks

Each unit provides the teacher with several benchmark activities, or subtasks, that feed into the larger unit project. Each of these subtasks contributes directly to the final product that the students will create. Teachers can use these benchmarks as intermediate, informal assessments to gauge the progress of each student and/or collaborative group in their mastery of the unit goals.

Scaffolding Activities

The bulk of each unit consists of a series of individual topic lessons, activities, discussions, and hands-on applications that allow the teacher to provide instruction, guidance, and support to students and collaborative groups as they design and implement the programming code for the unit project. These scaffolding activities serve to introduce, explain, and encourage the use of the unit's core concepts and skills by providing students with structured opportunities and incentives to explore the material in greater depth.

Final Products and Student Portfolio

At the culmination of each unit, student groups are expected to present a final product that represents the body of their work and code implementation on the unit project. Students demonstrate their mastery of the core content and skills for the unit by exhibiting the authentic and purposeful artifacts, in the form of working program code, that they have created. A key component of the project always includes a public presentation of each student's work before their peers as a way of providing motivation for each student and holding them accountable for their own learning.

Reflections

Students will be required to submit their project to the teacher in the form of a Student Submission Template, which provides the teacher with a link to the student's final program code along with thoughtful responses to specific questions about the topics covered in the unit project. This provides students an opportunity to reflect on the challenges and successes during the course of the unit project.

Resources and Technical Requirements

UTeach AP Computer Science A does not require additional materials for implementation besides the license, which may be purchased or provided through a grant.

Websites Accessed Throughout the Curriculum

The websites below will be accessed in this curriculum. Please advise the school IT department about these websites prior to beginning the course. Always test site access from your device as well as students' devices to ensure a smooth lesson.

**.codio.com, *.codio.io, uteachcs.github.io, youtube.com, vimeo.com, apcentral.collegeboard.org, docs.oracle.com, codingbat.com, topsoccerblog.com, epa.gov, scijinks.gov, aqicn.org, wunderground.com, wikipedia.org, google.com/maps*