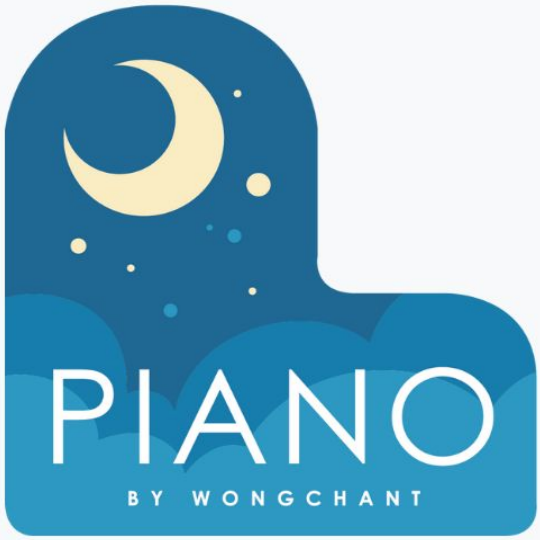

CURRICULUM OUTLINE COMPUTER SCIENCE A ('22-'23)

A Guide for Computer Science
A ('22-'23) Curriculum

COMPUTER SCIENCE A ('22-'23)

13 - 17 YEARS

MODULE 1

UNIT 1

OBJECT-ORIENTED PROGRAMMING

DESCRIPTION

This unit exposes students to object-oriented programming principles as they explore The Neighborhood. Students learn fundamental Java concepts as they navigate and interact with The Neighborhood with `Painter` objects and create new types of `Painter` to expand the capabilities of their programs. Students practice predicting the outcome of program code and use their `Painter` objects with conditional statements, while loops, and Boolean expressions to navigate mazes and create drawings. `while` students work with the `Painter` , they practice identifying syntax and logic errors to explain why code segments will not compile or work as intended. Additionally, students learn to document program code using comments to describe the behavior of specific code segments. Throughout this unit, students discover their identity as software engineers and use debugging strategies and code reviews to improve their programming skills.

AP Classroom Personal Progress Checks

To assign questions from the AP Classroom Question Bank that align with this unit, create a custom quiz in AP Classroom by searching the Question Bank for the Essential Knowledge statements on the unit's Standards page. You can find instructions and video demonstrations to do this on AP Central.

- **Survey**
  - CSA Pre-Survey

- **Lesson 1: Welcome to CSA**
  - **DESCRIPTION:** Believing in one's own software engineering identity develops confidence and perseverance. In this lesson, students are introduced to key characteristics and skills of software engineers and explore common perceptions. Students realize that they have many of the same characteristics and skills as software engineers and discover their software engineering identity.

- **Lesson 2: Java Lab**
  - **DESCRIPTION:** Students are introduced to the Java Lab programming environment to navigate Java source code files and identify correct Java syntax. Students experiment with the structure of a Java program to identify syntax rules of the Java programming language. While experimenting with Java syntax, students explore The Neighborhood and how the Painter can perform a task when given a set of instructions.
    - Investigate: Java Basics
    - Check for Understanding

MODULE 1

➢ **Lesson 3: The Neighborhood**

- **DESCRIPTION:** Students explore the structure of The Neighborhood to identify the organization of the grid and potential obstacles. Students learn that classes are blueprints that define the attributes and behaviors that an object can have and analyze the `Painter` class to identify these components. Students then practice creating `Painter` objects using the `new` keyword and determining the current state of the objects.
  - Investigate: The Painter
  - Instantiate a Painter Object
  - Check for Understanding

➢ **Lesson 4: Navigating and Painting**

- **DESCRIPTION:** Students learn the syntax for calling methods in Java to practice navigating and painting The Neighborhood. Students call `void` methods from the `Painter` class and practice debugging syntax errors.
  - Predict: Painter Class Methods
  - Moving and Painting
  - Check for Understanding

➢ **Lesson 5: One-Way Selection Statements**

- **DESCRIPTION:** Students incorporate `if` statements into their programs to make decisions based on the current state of a `Painter` object. Students learn about methods that return `boolean` values and how to execute a specific block of code based on the method call result. As the complexity of their programs increases, students identify debugging strategies to find and correct syntax and logic errors.
  - Predict: Conditionals and More Methods
  - Conditional Moving and Painting
  - Check for Understanding

➢ **Lesson 6: PainterPlus**

- **DESCRIPTION:** Students consider designing specialized types of classes and use inheritance to create the `PainterPlus` class. Students learn to create a new class and write a constructor that calls the superclass constructor using the `super` keyword. Students realize that while the `PainterPlus` class can perform the same behaviors as the `Painter` class, they can add new behaviors to expand its capabilities.
  - App: Is It Your Birthday?
  - App: National Parks
  - App: How Many Countries
  - App: My Famous Birthday
  - App: Stamp Notebook
  - App: The Fastest Finger
  - App: Guess the Number
  - App: Where Should I Live?

MODULE 1

➢ **Lesson 7: Writing Methods**

- **DESCRIPTION:** Students learn to write and use a new `void` method in the `PainterPlus` class to expand its capabilities. Students first consider the accessibility of new behaviors between superclasses and subclasses to identify situations when to write new methods in the superclass or the subclass. In the process, students discover that methods written in a subclass are not accessible in the superclass.
  - Writing and Using a New Method
  - Check for Understanding

➢ **Lesson 8: Code Reviews**

- **DESCRIPTION:** Software engineers use code reviews to develop code that is easy for others to read, maintain, and modify over time. Students identify the importance of commenting and program structure to improve readability. Students learn about code reviews and practice giving and receiving feedback.
  - Programming Style, Commits, and the Backpack
  - Conducting a Code Review
  - Check for Understanding

➢ **Lesson 9: Loops**

- **DESCRIPTION:** Students are introduced to iteration using `while` loops to repeat instructions as long as a condition is `true`. Students combine `while` loops with methods that return information about the state of an object to implement a new method in an existing class. Students practice developing algorithms using selection statements and iteration and implementing them in Java.
  - Predict: More Boolean Methods
  - Using Loops and Boolean Methods
  - Code Review: New PainterPlus Methods
  - Check for Understanding

➢ **Lesson 10: Two-Way Selection Statements**

- **DESCRIPTION:** Students expand their knowledge of selection statements and Boolean expressions using two-way selection statements and the NOT (!) logical operator. Students practice using these to check the state of an object and executing a specific set of instructions based on the result.
  - Investigate: Selection and Logic
  - Using Two-Way Selection and NOT
  - Code Review: New PainterPlus Methods
  - Check for Understanding

MODULE 1

➢ **Lesson 11: Debugging Strategies**

- **DESCRIPTION:** Students learn to write and use a new `void` method in the `PainterPlus` class to expand its capabilities. Students first consider the accessibility of new behaviors between superclasses and subclasses to identify situations when to write new methods in the superclass or the subclass. In the process, students discover that methods written in a subclass are not accessible in the superclass.
  - Writing and Using a New Method
  - Check for Understanding

➢ **Lesson 12: Decomposition and Design**

- **DESCRIPTION:** Students are introduced to decomposition and top-down design to deconstruct problems into smaller tasks and develop algorithms for these tasks. Students analyze decomposition examples and identify the importance of writing clear and specific pseudocode. Students write and translate algorithms into methods and consider potential edge cases to improve their programs.
  - More PainterPlus Methods
  - Code Review: More PainterPlus Methods
  - Check for Understanding

➢ **Lesson 13: PatternPainter**

- **DESCRIPTION:** Students write a new subclass and practice decomposition to develop algorithms. Students translate algorithms to write methods in their new subclass.
  - PatternPainter
  - Code Review: PatternPainter
  - Check for Understanding

➢ **Lesson 14: BackgroundPainter**

- **DESCRIPTION:** Students write an additional new subclass and practice using decomposition to develop algorithms. Students translate algorithms to write methods in their new subclass and create multiple objects in their program.
  - BackgroundPainter
  - Code Review: BackgroundPainter
  - Check for Understanding

MODULE 1

- **Lesson 15: Open Source Code**
  - **DESCRIPTION:** Students examine open source code and connect real-world applications and the concepts they have learned in this unit. Students review the characteristics of software engineers and reflect on how they demonstrate these characteristics while planning and implementing solutions.
    - Check for Understanding
- **Lesson 16: Asphalt Art Project – Day 1**
  - **DESCRIPTION:** For this project, students use the skills they have developed to create asphalt art in The Neighborhood. Students choose a theme or concept for their asphalt art that represents something they are interested in or meaningful to them. After brainstorming and planning, students develop their programs to create their designs.
    - Asphalt Art Project
- **Lesson 17: Asphalt Art Project – Day 2**
  - **DESCRIPTION:** Students have designed a new class to create their asphalt art. In this lesson, students decide which classes and methods they have developed throughout the unit to use and their new class to create their design. Students reinforce software development skills and processes by participating in a code review to give and use feedback.
    - Asphalt Art Project
    - Code Review: Asphalt Art Project
- **Lesson 18: Asphalt Art Project – Day 3**
  - **DESCRIPTION:** Students participated in a code review at the end of the previous lesson to give and receive feedback to identify areas of improvement and fixes for problems they encountered. Students consider this feedback as they finalize development in this lesson. Students share their final projects with their peers to celebrate their successes.
    - Asphalt Art Project
    - Unit 1 Survey

MODULE 1

➢ **Lesson 19: FRQ Practice**

- **DESCRIPTION:** Students are introduced to AP CSA Free Response Questions (FRQs) and practice decomposition and problem-solving skills to develop solutions to a mock FRQ. Students use annotation strategies to identify the key components of the problem and validate their solutions using Scoring Guidelines.

➢ **Lesson 20: Unit 1 Assessment**

- **DESCRIPTION:** Students complete a multiple-choice assessment that covers the unit topics.
  - Unit 1 Assessment

➢ **Lesson 21: Sandbox: Console**

➢ **Lesson 22: Sandbox: The Neighborhood**

- With Paint Bucket
- No Paint Bucket

UNIT 2

CLASS STRUCTURE AND DESIGN

DESCRIPTION

This unit expands on the object-oriented programming principles introduced in Unit 1 to explore design principles as students develop classes with attributes and behaviors and work with primitive and object data. Students learn to work with variables and user input as they develop and utilize classes and objects to represent desserts, customers, and food trucks for the Project Mercury Pastries Food Truck business. As students work with the `Dessert` class, they build on their knowledge of inheritance to design subclasses that represent specific types of desserts. While working with variables, students learn to create and evaluate expressions using assignment and relational operators and trace code segments to determine the result of output. Throughout this unit, students continue to develop software engineering skills as they learn to use UML diagrams to represent classes and the DRY principle to refactor program code.

AP Classroom Personal Progress Checks

To assign questions from the AP Classroom Question Bank that align with this unit, create a custom quiz in AP Classroom by searching the Question Bank for the Essential Knowledge statements on the unit's Standards page. You can find instructions and video demonstrations to do this on AP Central.

The following Personal Progress Checks in AP Classroom can be assigned during the Show What You Know week:

- Personal Progress Check 1
- Personal Progress Check 5
- Personal Progress Check 9

Note: *Some Learning Objectives and Essential Knowledge statements in the suggested Personal Progress Checks are covered in later units.*

➢ **Lesson 1: Variables**

- **DESCRIPTION:** Students expand their knowledge of data types to learn about the `int` and `double` types and differentiate between primitive and reference variables. Students explore how variables are declared, initialized, and used as they write and modify code segments and identify the current value stored in a variable.
  - Predict: Creating Variables
  - Investigate: Working with Variables
  - Creating and Printing Variables
  - Check for Understanding

➢ **Lesson 2: Operators and Expressions**

- **DESCRIPTION:** Students discover how arithmetic and compound assignment operators function in Java with different data types. Students practice evaluating expressions and explore scenarios that result in errors or unexpected results.
  - Predict: Evaluating Expressions
  - Investigate: Exploring Expressions
  - Check for Understanding

MODULE 1

➢ **Lesson 3: Attributes**

- **DESCRIPTION:** In the previous unit, students created subclasses of the `Painter` class and objects of different types. Students revisit classes and objects to learn how to design classes and identify appropriate attributes for these classes. Students define instance variables to represent these attributes while exploring encapsulation and learn the difference between `public` and `private` access.
  - Predict: Accessing Attributes
  - Check for Understanding

➢ **Lesson 4: Constructors**

- **DESCRIPTION:** Students revisit constructors to identify the need for parameterized constructors and learn how to implement these in a class. In the process, students differentiate between actual and formal parameters and explore how constructor calls function in programs.
  - Predict: Parameterized Constructor
  - Writing and Using Constructors
  - Check for Understanding

➢ **Lesson 5: User Input**

- **DESCRIPTION:** Students learn how to obtain user input using the `Scanner` class. Additionally, they discover the `NullPointerException`, its cause, and strategies for debugging this error.
  - Predict: Getting User Input
  - Getting User Input
  - Check for Understanding

➢ **Lesson 6: Class Hierarchies**

- **DESCRIPTION:** Students learned about inheritance in the previous unit to create subclasses of the `Painter` class. In this lesson, students revisit inheritance to learn about using the DRY principle to refactor redundant code. Students also learn about the `Object` class and deepen their understanding of inheritance.
  - Predict: Redundant Code
  - Investigate: Superclasses and Subclasses
  - Refactoring Desserts
  - Code Review: Refactoring Desserts
  - Check for Understanding

MODULE 1

➢ **Lesson 7: Accessor Methods**

- **DESCRIPTION:** Students discover the need for accessor methods to obtain the values stored in the instance variables of a class. Students review the components of a method signature and how methods return values. Students practice writing accessor methods in the Project Mercury Pastries Food Truck classes and calling them within expressions.
  - Writing Accessor Methods
  - Check for Understanding

➢ **Lesson 8: Mutator Methods**

- **DESCRIPTION:** Students discover the need for mutator methods to change the values stored in the instance variables of a class. Students also revisit the `void` keyword and recall edge cases. Students practice writing and calling mutator methods for their classes.
  - Predict: Parameterized Constructor
  - Writing and Using Constructors
  - Check for Understanding

➢ **Lesson 9: Printing Objects**

- **DESCRIPTION:** Students have learned to obtain information about an object using its accessor methods. In this lesson, students discover the need for an easier way to get the state of an object. Students revisit the `Object` class and learn how to override its `toString()` method to display object information to the console. Students also learn to format output using escape sequences and concatenating instance variables and `String`s.
  - Predict: Printing an Object
  - Overriding Object Class Methods
  - Check for Understanding

➢ **Lesson 10: Scope and this**

- **DESCRIPTION:** Students explore the scope of variables in a program and discover that the local variable is used when it has the same name as an instance variable. Students learn to use the this keyword to refer to the current object in constructors and methods.
  - Using the this Keyword
  - Check for Understanding

MODULE 1

- **Lesson 11: Store Management Project – Day 1**
  - **DESCRIPTION:** For this project, students use the skills they have developed throughout this unit to create a store management system for a business or service. Students choose the objects for their business that represent something they are interested in or meaningful to them. After brainstorming and planning, students develop their programs to create their store management system.
    - Store Management Project

- **Lesson 12: Store Management Project – Day 2**
  - **DESCRIPTION:** Students designed classes to represent objects in a business of their choice to create a store management program. Students decide how users interact with their program and use the Scanner class to receive user input. Students reinforce software development skills and processes as they participate in a code review to give and use feedback as they develop their program.
    - Store Management Project
    - Code Review: Store Management Project

- **Lesson 13: Store Management Project – Day 3**
  - **DESCRIPTION:** Students conducted code reviews at the end of the previous lesson to give and receive feedback that assists in identifying areas of improvement and fixes for problems they encountered. At this stage of the project, students consider this feedback as they finalize development in this lesson. Students share their final projects with their peers to celebrate their successes.
    - Store Management Project
    - Unit 2 Survey

- **Lesson 14: FRQ Practice**
  - **DESCRIPTION:** Students practice developing solutions to AP CSA FRQs on a mock Classes and Objects FRQ. Students apply decomposition and annotation strategies to identify the key components of the problem and validate their solutions using Scoring Guidelines.

MODULE 1

➢ **Lesson 15: Unit 2 Assessment**

- **DESCRIPTION:** Students complete a multiple-choice assessment that covers the unit topics.
  - Unit 2 Assessment

➢ **Lesson 16: Console**

➢ **Lesson 17: The Neighborhood**

- With Paint Bucket
- No Paint Bucket

MODULE 1

UNIT 3

ARRAYS AND ALGORITHMS

DESCRIPTION

This unit introduces students to data structures to store primitive values and object references. Students use one-dimensional (1D) arrays to represent lists of data while expanding their knowledge of loops and conditionals to analyze and process data in a 1D array. Students learn to use `for` loops to traverse arrays and discover that an algorithm involving loops can be implemented with either a `for` loop or a `while` loop. Throughout the unit, students develop and modify standard algorithms to find and manipulate elements in a 1D array while also discovering the concept of polymorphism when traversing arrays of objects. As part of the planning process, students identify the preconditions and postconditions that an algorithm must satisfy. Additionally, students learn to use the `static` and `final` keywords to create static variables to represent values shared with all classes and constants to represent values that cannot be changed.

AP Classroom Personal Progress Checks

To assign questions from the AP Classroom Question Bank that align with this unit, create a custom quiz in AP Classroom by searching the Question Bank for the Essential Knowledge statements on the unit's Standards page. You can find instructions and video demonstrations to do this on AP Central.

The following Personal Progress Checks in AP Classroom can be assigned during the Show What You Know week:

- Personal Progress Check 1
- Personal Progress Check 5
- Personal Progress Check 9

Note: *Some Learning Objectives and Essential Knowledge statements in the suggested Personal Progress Checks are covered in later units.*

➢ **Lesson 1: One-Dimensional (1D) Arrays**

- **DESCRIPTION:** Students have used variables to store literal values and references to objects. Software engineers often need to store multiple values of the same type in a program. In this lesson, students learn how to use one-dimensional (1D) arrays to store multiple values that can be referenced with one variable. Students practice creating arrays using the `new` keyword and with initializer lists and obtaining the size of the array.
  - Investigate: 1D Arrays
  - Using 1D Arrays
  - Check for Understanding

➢ **Lesson 2: Array Elements**

- **DESCRIPTION:** Students explore how to modify and obtain elements stored in a one-dimensional (1D) array. Students learn about common errors when working with 1D arrays and identify strategies for debugging these errors. Students practice modifying and printing elements in a 1D array individually and using a `while` loop.
  - Predict: Modifying Values
  - Investigate: 1D Array Elements
  - Modifying Values
  - Check for Understanding

MODULE 1

➢ **Lesson 3: For Loops**

- **DESCRIPTION:** Students learn about the structure and functionality of a `for` loop and use a for loop to traverse a one-dimensional (1D) array. Before using `for` loops with 1D arrays, students practice identifying the number of times a loop will execute and using trace tables to track the changing values of variables. Additionally, students differentiate between `while` loops and `for` loops and convert between both types of loops. Students practice writing `for` loops to traverse 1D arrays to access elements.
  - Using for Loops
  - Code Review: Using for Loops
  - Check for Understanding

➢ **Lesson 4: Array Algorithms**

- **DESCRIPTION:** Students apply their knowledge of traversing 1D arrays to develop algorithms to find elements that have specific properties. Students learn how to identify the preconditions and postconditions for an algorithm. Additionally, students differentiate between executing a `return` statement inside a loop and at the end of a method.
  - Predict: Finding Elements
  - Investigate: Exploring Algorithms
  - Writing Algorithms with Arrays
  - Code Review: Writing Algorithms with Arrays
  - Check for Understanding

➢ **Lesson 5: Enhanced For Loops**

- **DESCRIPTION:** Students learn about enhanced `for` loops to traverse a 1D array. As students explore the functionality of enhanced `for` loops, they identify differences, advantages, and disadvantages of using an enhanced `for` loop to traverse an array. Students use an enhanced `for` loop to calculate the sum or average of the elements in a 1D array.
  - Predict: Enhanced for Loops
  - Investigate: Enhanced for Loops
  - Using Enhanced for Loops
  - Check for Understanding

➢ **Lesson 6: Static Variables and Constants**

- **DESCRIPTION:** Students learn about static variables and constants and identify scenarios where these types of variables would be useful. Students differentiate between static variables and instance variables as they explore their functionality. Students practice using static variables and constants in program code, including implementing an algorithm to determine the number of elements meeting specific criteria.
  - Predict: Static Variables
  - Investigate: Constants
  - Using Static Variables and Constants
  - Check for Understanding

MODULE 1

➢ **Lesson 7: Polymorphism**

- **DESCRIPTION:** Students explore arrays of objects and learn how to refer to object data when accessing elements in an array. Students revisit inheritance and discover using polymorphism to work with 1D arrays of a superclass type. Students review overriding methods and traverse an array of objects to use subclass methods.
  - Predict: Arrays of Objects
  - Investigate: Polymorphism
  - Using Polymorphism
  - Check for Understanding

➢ **Lesson 8: More Array Algorithms**

- **DESCRIPTION:** Students explore the `File` class and use the `Scanner` class to read data from a file into a 1D array. In the process, students revisit the `ArrayIndexOutOfBoundsException` to identify debugging strategies to resolve off-by-one errors. Using data from a file, students develop algorithms to find the minimum and maximum values in a 1D array. Students plan their algorithms to identify the steps to solve the problem and write pseudocode to outline these steps.
  - Predict: Reading a File
  - Investigate: Accessing Pairs
  - Finding Minimum and Maximum Values
  - Check for Understanding

➢ **Lesson 9: Reversing Arrays**

- **DESCRIPTION:** Students develop an algorithm to reverse elements in a 1D array. Students plan their algorithms using manipulatives to identify the steps to solve the problem and write pseudocode to outline these steps.
  - Predict: In the Wrong Order
  - Reversing a 1D Array
  - Code Review: Reversing a 1D Array
  - Check for Understanding

➢ **Lesson 10: Finding Duplicates**

- **DESCRIPTION:** Students learn about nested loops and practice tracing these to explore their functionality. Using nested loops, students develop an algorithm to determine whether there are duplicate elements in a 1D array. Students plan their algorithms using manipulatives to identify the steps to solve the problem and write pseudocode before implementing them in Java.
  - Predict: Nested Loops
  - Finding Duplicates
  - Code Review: Finding Duplicates
  - Check for Understanding

MODULE 1

- **Lesson 11: Data for Social Good Project – Day 1**
  - **DESCRIPTION:** For this project, students use the skills they have developed throughout this unit to store and analyze data from a text file for a topic that they are interested in or meaningful to them. Students choose a user scenario and data they want to analyze to identify a solution to address a user's needs. After brainstorming and planning, students develop their programs.
    - Data for Social Good Project

- **Lesson 12: Data for Social Good Project – Day 2**
  - **DESCRIPTION:** Students identified the data for their topic of interest and read this data into a 1D array. Students continue to develop their projects using their data to implement standard algorithms to analyze the data. Students reinforce software development skills and processes as they participate in a code review to give and use feedback as they develop their program.
    - Data for Social Good Project
    - Code Review: Data for Social Good Project

- **Lesson 13: Data for Social Good Project – Day 3**
  - **DESCRIPTION:** Students conducted code reviews at the end of the previous lesson to give and receive feedback that assists in identifying areas of improvement and fixes for problems they encountered. At this stage of the project, students consider this feedback as they finalize development in this lesson. Students share their final projects with their peers to celebrate their successes.
    - Data for Social Good Project
    - Unit 3 Survey

- **Lesson 14: FRQ Practice**
  - **DESCRIPTION:** Students practice developing solutions to AP CSA FRQs on a mock Arrays FRQ. Students apply decomposition and annotation strategies to identify the key components of the problem and validate their solutions using Scoring Guidelines.

MODULE 1

➢ **Lesson 15: Unit 3 Assessment**

- **DESCRIPTION:** Students complete a multiple-choice assessment that covers the unit topics.
  - Unit 3 Assessment

➢ **Lesson 16: Console**

➢ **Lesson 17: The Neighborhood**

- With Paint Bucket
- No Paint Bucket

UNIT 4

CONDITIONS AND LOGIC

DESCRIPTION

This unit revisits the use of packages to access existing functionalities in their programs, such as The Neighborhood, to expand their knowledge of APIs and libraries and explore The Theater. With The Theater, students learn to improve the decisions made in conditionals and loops with relational and logical operators to evaluate primitive values and object references. While working with conditionals and logical operators, students expand their knowledge of the static keyword to write static methods and explore their functionality. Additionally, students use casting operators and Math class methods to evaluate expressions and perform calculations, including incorporating random in program decisions and behaviors. While working with compound Boolean expressions and logical operators, students develop an understanding of De Morgan's Laws and learn how to evaluate truth tables.

AP Classroom Personal Progress Checks

To assign questions from the AP Classroom Question Bank that align with this unit, create a custom quiz in AP Classroom by searching the Question Bank for the Essential Knowledge statements on the unit's Standards page. You can find instructions and video demonstrations to do this on AP Central.

The following Personal Progress Checks in AP Classroom can be assigned during the Show What You Know week:

- Personal Progress Check 3

Note: *Some Learning Objectives and Essential Knowledge statements in the suggested Personal Progress Checks are covered in later units.*

➢ **Lesson 1: The Theater**

- **DESCRIPTION:** Students are introduced to The Theater and explore its components and functionality. Students review the documentation for The Theater and an existing program to learn about its classes and methods, then practice using them to play sounds, create drawings, and show images.
  - Investigate: The Theater
  - The Theater Classes and Methods
  - Exploring The Theater
  - Check for Understanding

➢ **Lesson 2: Object Aliases and Equality**

- **DESCRIPTION:** Students discover the difference between comparing primitive values and object references. Students learn that two object references are considered aliases when they both reference the same object and how to identify aliases or `null` values using the `==` and `!=` operators. Students then revisit the `Object` class and override the `equals()` method to determine whether two objects of the class are equivalent.
  - Investigate: Comparing Objects
  - Investigate: Object Equality
  - Overriding the equals() Method
  - Check for Understanding

MODULE 1

➢ **Lesson 3: Nested if Statements**

- **DESCRIPTION:** Students have used `if` and `if-else` statements to execute blocks of code based on a condition. In this lesson, students explore the functionality of nested `if` statements and learn to use these to test a combination of conditions. Students use nested `if` statements to write an algorithm to compare pairs of values in a 1D array.
  - Investigate: Nested if Statements
  - Using Nested if Statements
  - Check for Understanding

➢ **Lesson 4: Logical Operators**

- **DESCRIPTION:** Students revisit logical operators to explore the use of the && (and) and || (or) operators. Students use truth tables to evaluate Boolean expressions and learn about short-circuited evaluation.
  - Investigate: Logical Operators
  - Check for Understanding

➢ **Lesson 5: De Morgan's Laws**

- **DESCRIPTION:** Students learn about De Morgan's Laws and how they are applied to Boolean expressions. Students explore how equivalent Boolean expressions evaluate to the same value in all cases. Students then use truth tables to evaluate compound Boolean expressions and prove De Morgan's Laws.
  - Check for Understanding

➢ **Lesson 6: Static Methods**

- **DESCRIPTION:** Students have written methods that require an object to be created before the method can be called. In this lesson, students revisit the `static` keyword and learn to write and call static methods. Students explore their functionality and identify scenarios for their use. Students then practice writing and calling static methods to solve problems.
  - Investigate: Static Methods
  - Using Static Methods
  - Code Review: Using Static Methods
  - Check for Understanding

MODULE 1

➢ **Lesson 7: The Math Class**

- **DESCRIPTION:** Students explore the `Math` class to learn about its structure and available methods. As students review the documentation for the `Math` class, they revisit the `final` keyword and learn how it can be used to prevent inheritance. Students practice using methods in the `Math` class to solve problems.
  - Investigate: The Math Class
  - Using Math Class Methods
  - Check for Understanding

➢ **Lesson 8: Casting and Rounding**

- **DESCRIPTION:** Students learn how casting operators can be used to temporarily convert a value to a different type. Students explore the effect of using the `(int)` and `(double)` casting operators on a value and expression. Students also use casting operators to round a `double` value to the nearest integer.
  - Investigate: Casting
  - Investigate: Division and Printing
  - Investigate: Rounding
  - Check for Understanding

➢ **Lesson 9: Random**

- **DESCRIPTION:** Students revisit the `Math` class and learn to use the `Math.random()` method to generate random numbers. Students explore how to manipulate the values returned from `Math.random()` to produce a random `int` or `double` in a specific range. Students then practice generating and using random numbers.
  - Predict: Random Numbers
  - Investigate: Generating Random Numbers
  - Using Random Numbers
  - Code Review: Using Random Numbers
  - Check for Understanding

➢ **Lesson 10: Multi-Selection Statements**

- **DESCRIPTION:** Students expand their knowledge of selection statements to evaluate and implement multi-selection statements. Students explore the functionality of multi-selection statements and learn to use these to test a combination of conditions. Students use multi-selection statements to implement algorithms to solve problems.
  - Investigate: Multi-Selection Statements
  - Using Multi-Selection Statements
  - Finding Patterns in Data
  - Check for Understanding

MODULE 1

- **Lesson 11: Abstract Data Art Project – Day 1**
  - **DESCRIPTION:** For this project, students use the skills they have developed throughout this unit to analyze data from a text file for a topic they are interested in or meaningful to them. Students use the data they chose in a previous lesson to identify the patterns and message they want to convey visually using The Theater. After brainstorming and planning, students develop their programs.
    - Abstract Data Art Project

- **Lesson 12: Abstract Data Art Project – Day 2**
  - **DESCRIPTION:** Students planned their visuals and algorithms to convey a story about their chosen dataset. Students continue to develop their projects to implement algorithms to analyze the data and portray the findings and results visually. Students reinforce software development skills and processes as they participate in a code review to give and use feedback as they develop their program.
    - Abstract Data Art Project
    - Code Review: Abstract Data Art Project

- **Lesson 13: Abstract Data Art Project – Day 3**
  - **DESCRIPTION:** Students conducted code reviews at the end of the previous lesson to give and receive feedback that assists in identifying areas of improvement and fixes for problems they encountered. At this stage of the project, students consider this feedback as they finalize development in this lesson. Students share their final projects with their peers to celebrate their successes.
    - Abstract Data Art Project
    - Unit 4 Survey

- **Lesson 14: FRQ Practice**
  - **DESCRIPTION:** Students practice developing solutions to AP CSA FRQs on a mock Arrays FRQ. Students apply decomposition and annotation strategies to identify the key components of the problem and validate their solutions using Scoring Guidelines.

MODULE 1

➢ **Lesson 15: Unit 4 Assessment**

- **DESCRIPTION:** Students complete a multiple-choice assessment that covers the unit topics.
  - Unit 4 Assessment

➢ **Lesson 16: Sandbox: Console**

➢ **Lesson 17: Sandbox: The Neighborhood**

- With Paint Bucket
- No Paint Bucket

➢ **Lesson 18: Sandbox: The Theater**

.

UNIT 5

TWO-DIMENSIONAL ARRAYS

DESCRIPTION

This unit expands on data structures introduced in Unit 3 to create tables of data using two-dimensional (2D) arrays. Students identify similarities and differences between 1D and 2D arrays when creating, accessing, and traversing 2D arrays and apply standard algorithms to find and manipulate elements. As students analyze problems involving 2D arrays, they revisit these standard algorithms to determine what code needs to be added or modified and to interact with completed program code. Students apply these concepts to manipulate pixels and in The Theater to create image filters in addition to working with primitive values and various object references. Additionally, students use the programming knowledge and skills they have acquired to consider the impacts of programs on society, economies, and culture.

AP Classroom Personal Progress Checks

To assign questions from the AP Classroom Question Bank that align with this unit, create a custom quiz in AP Classroom by searching the Question Bank for the Essential Knowledge statements on the unit's Standards page. You can find instructions and video demonstrations to do this on AP Central.

The following Personal Progress Checks in AP Classroom can be assigned during the Show What You Know week:

- Personal Progress Check 8

Note: *Some Learning Objectives and Essential Knowledge statements in the suggested Personal Progress Checks are covered in later units.*

➢ **Lesson 1: Two-Dimensional (2D) Arrays**

- **DESCRIPTION:** Students explore the structure of two-dimensional (2D) arrays and identify similarities and differences between 1D and 2D arrays. Students learn to access and modify individual elements by referring to their row and column index values. Students then practice declaring and initializing 2D arrays using the `new` keyword and initializer lists and work with the elements stored in the arrays.
  - Investigate: Two-Dimensional (2D) Arrays
  - Using 2D Arrays
  - Check for Understanding

➢ **Lesson 2: Row-Major Traversal**

- Investigate: Nested Loops and 2D Arrays
- Traversing 2D Arrays
- Check for Understanding

MODULE 1

➢ **Lesson 3: Column-Major Traversal**

- **DESCRIPTION:** Students learn to use nested loops to traverse a 2D array in column-major order and differentiate between column-major and row-major traversals. Students identify scenarios that are best for traversing in row-major and column-major order and develop an algorithm to traverse a 2D array in column-major order to find the sum of all values in each column of the array. Students then practice applying standard 1D array algorithms to 2D arrays to find information and perform calculations.
  - Predict: Column-Major Traversal
  - Traversing 2D Arrays
  - Check for Understanding

➢ **Lesson 4: Reverse a 2D Array**

- **DESCRIPTION:** Students revisit the algorithm they developed to reverse a 1D array and plan an algorithm to reverse a 2D array using pseudocode and manipulatives. Students then implement their algorithm in Java and practice applying their algorithm to various scenarios.
  - Reversing 2D Arrays
  - Code Review: Reversing 2D Arrays
  - Check for Understanding

➢ **Lesson 5: Enhanced For Loops**

- **DESCRIPTION:** Students revisit enhanced `for` loops to explore how these are used with 2D arrays. In the process, students discover that objects can be modified using an enhanced `for` loop because the loop variable is an alias of the object reference in the array. Students explore how enhanced for loops can be used with 2D arrays and identify similarities and differences between using enhanced `for` loops with 1D arrays and 2D arrays. Students then practice using enhanced `for` loops with 2D arrays.
  - Predict: Enhanced For Loops and Objects
  - Investigate: Enhanced For Loops and 2D Arrays
  - Using Enhanced For Loops with 2D Arrays
  - Code Review: Using Enhanced For Loops with 2D Arrays
  - Check for Understanding

➢ **Lesson 6: Images in The Theater**

- **DESCRIPTION:** Students apply their knowledge of 2D arrays and object-oriented programming to create image filters using The Theater. Students learn how images are made up of pixels and that pixels consist of red, green, and blue color values and explore the `Image` and `Pixel` classes. Students write pseudocode to create an image filter that sets a specific color value to zero and then implement their algorithm in Java.
  - Investigate: Modifying Pixels
  - Creating Image Filters
  - Check for Understanding

MODULE 1

➢ **Lesson 7: Impacts of Programs**

- **DESCRIPTION:** Students explore the beneficial and harmful impacts of programs by analyzing and discussing real-world applications of image manipulation, such as deepfakes. Students discover that deepfakes are created using large amounts of data and reflect on how data is collected and the ethical and social implications of data collection.
  - Check for Understanding

➢ **Lesson 8: Modifying Images**

- **DESCRIPTION:** Students practice working with 2D arrays and modifying standard algorithms to create new image filters using The Theater. Students explore an example of a new image filter and then write pseudocode to create a grayscale image filter. Students implement their algorithm in Java and create additional image filters by modifying `Pixel` objects stored in a 2D array.
  - Investigate: Changing Colors in an Image
  - Creating Image Filters
  - Code Review: Creating Image Filters
  - Check for Understanding

➢ **Lesson 9: Finding Duplicates**

- **DESCRIPTION:** Students revisit the algorithm they developed to find duplicate values in a 1D array and plan an algorithm to find duplicates in a 2D array using pseudocode and manipulatives. Students then implement their algorithm in Java and practice applying their algorithm to various scenarios.
  - Finding Duplicates in a 2D Array
  - Code Review: Finding Duplicates in a 2D Array
  - Check for Understanding

➢ **Lesson 10: Shifting a 2D Array**

- **DESCRIPTION:** Students practice working with 2D arrays and modifying standard algorithms to plan and implement an algorithm to shift a 2D array. Students explore how the elements in a 1D array can be shifted and then write pseudocode to modify the algorithm to be applied to a 2D array.
  - Investigate: Shifting Elements
  - Shifting a 2D Array
  - Code Review: Shifting a 2D Array
  - Check for Understanding

MODULE 1

- **Lesson 11: Personal Narrative Project – Day 1**
  - **DESCRIPTION:** For this project, students use the skills they have developed throughout this unit to create new image filters to use in a collage or animation that portrays personal experiences and/or interests. After brainstorming and planning, students develop their programs.
    - Personal Narrative Project

- **Lesson 12: Personal Narrative Project – Day 2**
  - **DESCRIPTION:** Students planned their visuals and algorithms to create a collage or animation about their personal experiences and/or interests. Students continue to develop their projects to implement algorithms to create a new image filter and their collage or animation. Students reinforce software development skills and processes as they participate in a code review to give and use feedback as they develop their program.
    - Personal Narrative Project
    - Code Review: Personal Narrative Project

- **Lesson 13: Personal Narrative Project – Day 3**
  - **DESCRIPTION:** Students conducted code reviews at the end of the previous lesson to give and receive feedback that assists in identifying areas of improvement and fixes for problems they encountered. At this stage of the project, students consider this feedback as they finalize development in this lesson. Students share their final projects with their peers to celebrate their successes.
    - Personal Narrative Project
    - Unit 5 Survey

- **Lesson 14: FRQ Practice**
  - **DESCRIPTION:** Students practice developing solutions to AP CSA FRQs on a mock 2D Arrays FRQ. Students apply decomposition and annotation strategies to identify the key components of the problem and validate their solutions using Scoring Guidelines.

MODULE 1

➢ **Lesson 15: Unit 5 Assessment**

- **DESCRIPTION:** Students complete a multiple-choice assessment that covers the unit topics.
  - Unit 5 Assessment

➢ **Lesson 16: Console**

➢ **Lesson 17: The Neighborhood**

- With Paint Bucket
- No Paint Bucket

➢ **Lesson 18: The Theater**

UNIT 6

ARRAYLISTS AND STRING METHODS

DESCRIPTION

This unit continues to expand on data structures to introduce students to creating lists using the ArrayList class. In the process, students learn about the `Integer` and `Double` classes and use their methods to parse data from text files and explore the limits of integer values. Students differentiate between when to use each type of data structure while learning about the structure and functionality of an `ArrayList`. Students apply standard algorithms to find and manipulate data in an `ArrayList` of numerical and object data. Throughout the unit, students learn to use the String class to analyze and process text obtained from a user and from file input while learning about basic natural language processing techniques and applications. Additionally, students further develop software engineering skills by writing Javadoc comments to create API documentation for their programs.

AP Classroom Personal Progress Checks

To assign questions from the AP Classroom Question Bank that align with this unit, create a custom quiz in AP Classroom by searching the Question Bank for the Essential Knowledge statements on the unit's Standards page. You can find instructions and video demonstrations to do this on AP Central.

The following Personal Progress Checks in AP Classroom can be assigned during the Show What You Know week:

- Personal Progress Check 2
- Personal Progress Check 7

Note: *Some Learning Objectives and Essential Knowledge statements in the suggested Personal Progress Checks are covered in later units.*

➢ **Lesson 1: Project Planning**

- **DESCRIPTION:** Students are introduced to the Natural Language Processing Project and evaluate requirements and examples to identify questions and key features. Before beginning work on the project, students expand their understanding of program structure and comments by identifying best practices and exploring the use of Javadocs to generate program documentation.
  - Natural Language Processing Project Examples
  - Documenting Program Code
  - Check for Understanding

➢ **Lesson 2: Integer and Double Objects**

- **DESCRIPTION:** Students construct `Integer` and `Double` objects to represent `int` and `double` primitive values. Using the methods in the `Integer` and `Double` classes, students return the values these objects store as primitive values. Additionally, students explore the minimum and maximum values for integers to avoid overflow errors.
  - Investigate: Integer and Double Objects
  - Using Wrapper Classes
  - Check for Understanding

MODULE 1

➢ **Lesson 3: ArrayLists**

- **DESCRIPTION:** Students explore the `ArrayList` class and learn how to declare and initialize an `ArrayList` and add elements. Students identify scenarios where an `ArrayList` is more suitable and differentiate between a dynamic data structure and a static data structure. Students then practice creating an `ArrayList` and adding values to the list.
  - Investigate: ArrayLists
  - Using ArrayLists
  - Check for Understanding

➢ **Lesson 4: Manipulating Elements**

- **DESCRIPTION:** Students work with the `get()` and `set()` methods to retrieve and modify elements in an `ArrayList` and learn about the `String` method `length()` to return the number of characters in a `String` object. They apply the standard algorithms they developed with 1D and 2D arrays to obtain and modify elements in an `ArrayList`.
  - Investigate: Working with ArrayList Data
  - Working with ArrayLists
  - Check for Understanding

➢ **Lesson 5: Lists of Objects**

- **DESCRIPTION:** Students explore generic types in Java and learn why an `ArrayList` can only store objects. Students then practice applying the algorithms they learned with 1D and 2D arrays to `ArrayList` to find information and perform calculations with `ArrayList` data.
  - Investigate: ArrayList of Objects
  - ArrayList Algorithms
  - Code Review: ArrayList Algorithms
  - Check for Understanding

➢ **Lesson 6: Removing Elements**

- **DESCRIPTION:** Students explore the functionality of the `remove()` method in the `ArrayList` class and identify potential errors when attempting to remove items while traversing the list. Students learn about stop word removal and practice removing elements from an `ArrayList`.
  - Investigate: Removing Values
  - Predict: Using an Enhanced For Loop
  - Investigate: Removing with an Enhanced For Loop
  - Removing Values from an ArrayList
  - Check for Understanding

MODULE 1

➢ **Lesson 8: Comparing Strings**

- **DESCRIPTION:** Students explore the functionality of the `compareTo()` method then write an algorithm to sort a list of `String`s in alphabetical order. Students implement their algorithms in Java and practice using the `compareTo()` method to determine if one `String` comes before another and to sort a list of `String`s in alphabetical order.
  - Investigate: ArrayLists
  - Using ArrayLists
  - Check for Understanding

➢ **Lesson 9: String Algorithms**

- **DESCRIPTION:** Students combine their knowledge of standard algorithms and `String`s to write an algorithm to check if a `String` is a palindrome. Students implement their algorithm in Java, then apply standard algorithms to traverse `String`s find if one or more substrings has a particular property or to determine the number of substrings that meet specific criteria.
  - String Algorithms
  - Code Review: String Algorithms
  - Check for Understanding

➢ **Lesson 10: ArrayList Algorithms**

- **DESCRIPTION:** Students explore a new standard algorithm to obtain individual digits from an integer and learn how sentiment values are used in natural language processing. Students write an algorithm to find the sentiment value of a word and apply their algorithm to various scenarios to find the overall sentiment value of text.
  - Investigate: Obtaining Individual Digits
  - Finding the Sentiment Value
  - Code Review: Finding the Sentiment Value
  - Check for Understanding

➢ **Lesson 11: Natural Language Processing Project – Day 1**

- **DESCRIPTION:** For this project, students use the skills they have developed throughout this unit to create a program that uses natural language processing techniques to analyze, manipulate, and/or generate text. After brainstorming and planning, students develop their programs.
  - Natural Language Processing Project

MODULE 1

- **Lesson 12: Natural Language Processing Project – Day 2**
  - **DESCRIPTION:** Students planned their classes and algorithms to create a program that analyzes, manipulates, and/or generates text using natural language processing techniques. Students continue to develop their projects to implement algorithms to analyze text and interact with a user. Students reinforce software development skills and processes as they participate in a code review to give and use feedback as they develop their program.
    - Natural Language Processing Project
    - Code Review: Natural Language Processing Project

- **Lesson 13: Natural Language Processing Project – Day 3**
  - **DESCRIPTION:** Students conducted code reviews at the end of the previous lesson to give and receive feedback that assists in identifying areas of improvement and fixes for problems they encountered. At this stage of the project, students consider this feedback as they finalize development in this lesson. Students share their final projects with their peers to celebrate their successes.
    - Natural Language Processing Project
    - Unit 6 Survey

- **Lesson 14: FRQ Practice**
  - **DESCRIPTION:** Students practice developing solutions to AP CSA FRQs on a mock Arrays FRQ. Students apply decomposition and annotation strategies to identify the key components of the problem and validate their solutions using Scoring Guidelines.

- **Lesson 15: Unit 6 Assessment**
  - **DESCRIPTION:** Students complete a multiple-choice assessment that covers the unit topics.
    - Unit 6 Assessment

- **Lesson 16: Console**
- **Lesson 17: The Neighborhood**
  - With Paint Bucket
  - No Paint Bucket
- **Lesson 18: The Theater**

UNIT 7

METHOD DECOMPOSITION AND RECURSION

DESCRIPTION

This unit allows students to practice software design and development using the skills they have learned throughout the curriculum while planning and developing a creative coding project to convey a personal interest or story using The Theater. Students use decomposition strategies and object-oriented principles to plan and implement their ideas while ensuring their projects meet specified requirements. In the process, students learn to write `private`, overloaded, and overridden methods and use the `super` keyword in a subclass method to call a superclass method while exploring the functionality of methods and their parameters. Throughout the unit, students practice tracing and writing recursive methods and comparing these methods to iterative solutions. With the knowledge and skills acquired throughout the year, students consider the need for maximizing system reliability as they explore bugs and issues in existing programs.

AP Classroom Personal Progress Checks

To assign questions from the AP Classroom Question Bank that align with this unit, create a custom quiz in AP Classroom by searching the Question Bank for the Essential Knowledge statements on the unit's Standards page. You can find instructions and video demonstrations to do this on AP Central.

- **Lesson 1: Project Planning**
  - **DESCRIPTION:** Students are introduced to the Creative Coding with The Theater Project and evaluate requirements and examples to identify questions and key features. Students learn about project backlogs and benchmarks and set up their Project Planning Board by identifying some of the tasks they need to complete for the project.
    - Creative Coding with The Theater Example Projects
    - Check for Understanding

- **Lesson 2: Method Parameters**
  - **DESCRIPTION:** Students revisit passing primitives and object references as parameters to identify similarities and differences between each. Students explore the impact of using an object reference as a parameter on the original object and identify scenarios when it is not ideal to use an object reference as a parameter. Students then learn about user stories and write user stories for their projects to identify required program features.
    - Investigate: Object References as Parameters
    - Using Object References as Parameters
    - Check for Understanding

MODULE 1

➢ **Lesson 3: Overloading Methods**

- **DESCRIPTION:** Students revisit overloading and explore overloading methods to determine their impacts on the functionality of a program. Students practice writing overloaded methods to solve problems using the console, The Neighborhood, or The Theater before beginning their initial plans for their Creative Coding with The Theater Project.
  - Predict: Overloaded Methods
  - Writing Overloaded Methods
  - Check for Understanding

➢ **Lesson 4: Private Methods**

- **DESCRIPTION:** Students revisit access modifiers to explore the accessibility of methods and constructors in a program. Students identify scenarios for using a `private` method and constructor and practice writing and using `private` methods using the console or The Theater. Students then finalize their planning for their Creative Coding with The Theater Project and participate in a peer review to receive feedback on their work.
  - Investigate: Private Methods
  - Writing Private Methods
  - Check for Understanding

➢ **Lesson 5: Overriding Methods**

- **DESCRIPTION:** Students have overridden methods from the `Object` class and worked with class hierarchies that implemented overridden methods. Students explore writing and calling overridden methods and how the `super` keyword can be used to call the superclass version of a method. Students identify scenarios for overriding methods and practice writing and calling overridden methods using the console, The Neighborhood, or The Theater.
  - Investigate: Overriding Methods
  - Overriding Methods
  - Code Review: Overriding Methods
  - Check for Understanding

➢ **Lesson 6: Intellectual Property**

- **DESCRIPTION:** Students explore the ethical and social implications of intellectual property by analyzing and discussing real-world applications of algorithmic decision-making. Students learn why algorithms should be held accountable and regulated through audits and reflect on whether they would want their code to be audited.
  - Check for Understanding

MODULE 1

➢ **Lesson 7: Recursion**

- **DESCRIPTION:** Students explore recursive methods through an unplugged activity and practice tracing code with recursive methods. Students then work through the tasks on their Project Planning Board to begin the development of their Creative Coding with The Theater Project.
  - Creative Coding with The Theater Project
  - Check for Understanding

➢ **Lesson 8: Recursion vs Iteration**

- **DESCRIPTION:** Students explore how iterative solutions can be implemented using recursion and identify the similarities and differences in both algorithms. Students practice converting iterative algorithms to recursion then continue development on their Creative Coding with The Theater Project.
  - Investigate: Recursion vs Iteration
  - Converting Iterative Algorithms to Recursion
  - Creative Coding with The Theater Project
  - Check for Understanding

➢ **Lesson 9: Recursive Algorithms**

- **DESCRIPTION:** Students explore how the standard algorithms they have worked with can be implemented using recursion. Students practice tracing recursive solutions involving `String`s, arrays, and `ArrayList`s using the console or The Theater. Students then participate in a peer review to give and receive feedback on the progress they have made on their Creative Coding with The Theater Project.
  - Investigate: Recursive Algorithms
  - Tracing Recursive Methods
  - Code Review: Creative Coding with The Theater Project
  - Check for Understanding

➢ **Lesson 10: Maximizing System Reliability**

- **DESCRIPTION:** Students explore the issues and impacts resulting from open source software bugs. Students discover strategies for maximizing system reliability.
  - Check for Understanding

MODULE 1

- **Lesson 11: Creative Coding with The Theater – Day 1**
  - **DESCRIPTION:** Students continue to develop their Creative Coding Project to create a program that portrays a personal interest or solves a problem using The Theater. Students first self-assess their work to evaluate their progress in completing project requirements, then continue developing their projects. Students then conduct acceptance testing to determine which of their user stories are complete and use the self-assessment and peer feedback to prioritize their remaining tasks.
    - Creative Coding with The Theater Project
    - Code Review: Creative Coding with The Theater Project

- **Lesson 12: Creative Coding with The Theater – Day 2**
  - **DESCRIPTION:** Students conducted code reviews at the end of the previous lesson to give and receive feedback that assists in identifying areas of improvement and fixes for problems they encountered. At this stage of the project, students consider this feedback as they finalize development in this lesson. Students share their final projects with their peers to celebrate their successes.
    - Creative Coding with The Theater Project
    - Unit 7 Survey

- **Lesson 14: FRQ Practice**
  - **DESCRIPTION:** Students practice developing solutions to AP CSA FRQs on a mock Arrays FRQ. Students apply decomposition and annotation strategies to identify the key components of the problem and validate their solutions using Scoring Guidelines.

- **Lesson 15: Unit 7 Assessment**
  - **DESCRIPTION:** Students complete a multiple-choice assessment that covers the unit topics.
    - Unit 7 Assessment

- **Lesson 16: Console**
- **Lesson 17: The Neighborhood**
  - With Paint Bucket
  - No Paint Bucket
- **Lesson 18: The Theater**

UNIT 8

SEARCHING AND SORTING

DESCRIPTION

This unit expands on algorithms students have learned to introduce common approaches to searching and sorting 1D and 2D arrays and ArrayLists. In the process, students analyze and compare the efficiencies of these algorithms using statement execution counts and further develop problem-solving skills to decompose complex problems. Throughout the unit, students apply their programming and software engineering skills to plan and develop a creative coding project using the console that incorporates object-oriented design, data structures, and algorithmic thinking. With the knowledge and skills acquired throughout the year, students consider the privacy and security of programs and users.

AP Classroom Personal Progress Checks

To assign questions from the AP Classroom Question Bank that align with this unit, create a custom quiz in AP Classroom by searching the Question Bank for the Essential Knowledge statements on the unit's Standards page. You can find instructions and video demonstrations to do this on AP Central.

The following Personal Progress Checks in AP Classroom can be assigned during the Show What You Know week:

- Personal Progress Check 10

Note: *Some Learning Objectives and Essential Knowledge statements in the suggested Personal Progress Checks are covered in later units.*

➢ **Lesson 1: Project Planning**

- **DESCRIPTION:** Students are introduced to the Creative Coding with the Console Project and evaluate requirements and examples to identify questions and key features. Students learn about project backlogs and benchmarks and set up their Project Planning Board by identifying some of the tasks they need to complete for the project.
  - Creative Coding with the Console Example Projects
  - Check for Understanding

➢ **Lesson 2: Searching**

- **DESCRIPTION:** Students have used searching in standard algorithms to find a target element in a list. In this lesson, students explore the linear search algorithm and how it can be applied to different data structures. Students learn how to use execution counts to determine the best, average, and worst cases and practice evaluating algorithms for their efficiency. Students then write user stories for their projects to identify required program features.
  - Investigate: Linear/Sequential Search
  - Check for Understanding

MODULE 1

- **Lesson 3: Binary Search**
  - **DESCRIPTION:** Students discover a better searching algorithm that can perform more efficiently if the list is already sorted. Students trace code for a binary search to evaluate its efficiency and compare it to the linear search. Students apply the binary search algorithm to solve problems in a variety of scenarios. Students then begin their initial plans for their Creative Coding with the Console Project.
    - Investigate: Binary Search
    - Check for Understanding

- **Lesson 4: Recursive Binary Search**
  - **DESCRIPTION:** Students revisit recursion and explore how the binary search algorithm can be implemented recursively. Students then practice implementing the binary search algorithm iteratively or recursively before finalizing their planning for their Creative Coding with the Console Project and participating in a peer review to receive feedback on their work.
    - Investigate: Recursive Binary Search
    - Implementing the Binary Search
    - Check for Understanding

- **Lesson 5: Selection Sort**
  - **DESCRIPTION:** Students explore the selection sort algorithm and practice tracing the algorithm to evaluate its efficiency. Students identify the benefits and limitations of the selection sort algorithm, then practice implementing it in Java to sort elements.
    - Investigate: Selection Sort
    - Using the Selection Sort
    - Check for Understanding

- **Lesson 6: Insertion Sort**
  - **DESCRIPTION:** Students explore the insertion sort algorithm and compare its functionality and efficiency with the selection sort algorithm. Students analyze the efficiency of each algorithm using execution counts and identify the benefits and limitations of the selection and insertion sort algorithms.
    - Investigate: Insertion Sort vs Selection Sort
    - Check for Understanding

MODULE 1

➢ **Lesson 7: Merge Sort**

- **DESCRIPTION:** Students explore the merge sort algorithm to organize elements in a list using a divide-and-conquer approach. Students identify the components of the algorithm and learn about helper functions. Students then trace the algorithm to understand its functionality and determine its efficiency using execution counts. Students then work through the tasks on their Project Planning Board to begin the development of their Creative Coding with the Console Project.
  - Predict: Merge Sort
  - Creative Coding with the Console Project
  - Check for Understanding

➢ **Lesson 8: Multiple Lists**

- **DESCRIPTION:** Students explore traversing multiple lists at the same time by creating algorithms and tracing code involving multiple lists. Students identify scenarios where multiple lists might be needed and consider how multiple lists might be used and traversed in games. Students then continue working through the tasks on their Project Planning Board to develop their Creative Coding with the Console Project.
  - Predict: Traversing Multiple Lists
  - Traversing Multiple Lists
  - Creative Coding with the Console Project
  - Check for Understanding

➢ **Lesson 9: Searching and Sorting**

- **DESCRIPTION:** Students have explored searching and sorting algorithms, analyzed their efficiencies using execution counts, and implemented algorithms to work with multiple lists. Students plan and implement solutions using searching and sorting algorithms to solve problems involving one or more lists. Students then participate in a peer review to give and receive feedback on the progress they have made on their Creative Coding with the Console Project.
  - Searching and Sorting Algorithms
  - Code Review: Creative Coding with the Console Project
  - Check for Understanding

➢ **Lesson 10: Privacy and Security**

- **DESCRIPTION:** Students recall how their data is collected then think through how they would use that data as software engineers. They read case studies on privacy and security breaches and debate about the tradeoffs between the utility of data collection and safeguarding privacy and security.
  - Check for Understanding

MODULE 1

- **Lesson 11: Creative Coding with the Console – Day 1**
  - **DESCRIPTION:** Students continue to develop their Creative Coding Project to create a program that portrays a personal interest or solves a problem using the console. Students first self-assess their work to evaluate their progress in completing project requirements, then continue developing their projects. Students then conduct acceptance testing to determine which of their user stories are complete and use the self-assessment and peer feedback to prioritize their remaining tasks.
    - Creative Coding with the Console Project
    - Code Review: Creative Coding with the Console Project

- **Lesson 12: Creative Coding with the Console – Day 2**
  - **DESCRIPTION:** In the previous lesson, students conducted self-assessments and wrote acceptance criteria for their projects. Students obtain peer feedback in this lesson to identify remaining tasks to complete and continue developing their Creative Coding with the Console Project.
    - Creative Coding with the Console Project
    - Code Review: Creative Coding with the Console Project

- **Lesson 13: Creative Coding with the Console – Day 3**
  - **DESCRIPTION:** Students conducted code reviews at the end of the previous lesson to give and receive feedback that assists in identifying areas of improvement and fixes for problems they encountered. At this stage of the project, students consider this feedback as they finalize development in this lesson. Students share their final projects with their peers to celebrate their successes.
    - Creative Coding with the Console Project
    - Unit 8 Survey

- **Lesson 14: FRQ Practice**
  - **DESCRIPTION:** Students practice developing solutions to AP CSA FRQs on a mock 2D Arrays FRQ. Students apply decomposition and annotation strategies to identify the key components of the problem and validate their solutions using Scoring Guidelines.

MODULE 1

➢ **Lesson 15: Unit 8 Assessment**

- **DESCRIPTION:** Students complete a multiple-choice assessment that covers the unit topics.
  - Unit 8 Assessment

➢ **Lesson 16: Console**

➢ **Lesson 17: The Neighborhood**

- With Paint Bucket
- No Paint Bucket

➢ **Lesson 18: The Theater**

MODULE 1

- **CSA Post-Survey**

UNIT 9

REVIEW AND PRACTICE

DESCRIPTION

This unit prepares students for the AP CSA Exam by reviewing key concepts, practicing multiple-choice and free response questions, and strengthening test-taking strategies. Students identify strengths and areas of improvement to create individualized study plans to focus their practice and self-assess their progress.

- **Lesson 1: The AP CSA Exam**
  - **DESCRIPTION:** Students learn about the structure and expectations of the AP CSA Exam. In the process, students identify and answer questions about the exam. Students explore test-taking strategies and study resources that they can use to prepare for the exam.
- **Lesson 2: MCQ Pre-Assessment**
  - **DESCRIPTION:** Students complete a Multiple-Choice Pre-Assessment on AP Classroom.

MODULE 1

- **Lesson 3: MCQ Study Plan**
  - **DESCRIPTION:** Students review their results from the multiple-choice pre-assessment to create a study plan.
- **Lesson 5: MCQ Practice – Day 2**
  - **DESCRIPTION:** Students reflect on their study plan and identify their highest priority areas for improvement. Students use a choice board of study strategies to practice for the AP CSA Exam.
- **Lesson 6: FRQ Pre-Assessment**
  - **DESCRIPTION:** Students complete an FRQ Pre-Assessment.
- **Lesson 7: FRQ Study Plan**
  - **DESCRIPTION:** Students review their results from the FRQ pre-assessment to create a study plan.

MODULE 1

- **Lesson 8: FRQ Practice – Day 1**
  - **DESCRIPTION:** Students reflect on their study plan and identify their highest priority areas for improvement. Students use a choice board of study strategies to practice for the AP CSA Exam.
- **Lesson 9: FRQ Practice – Day 2**
  - **DESCRIPTION:** Students identify strategies for approaching FRQs on the AP CSA Exam.
- **Lesson 10: Mock MCQ Exam – Day 1**
  - **DESCRIPTION:** Students complete the first half of a mock AP CSA multiple-choice exam.
- **Lesson 11: Mock MCQ Exam – Day 2**
  - **DESCRIPTION:** Students complete the second half of a mock AP CSA multiple-choice exam.

MODULE 1

- **Lesson 12: Mock MCQ Exam – Day 3**
  - **DESCRIPTION:** Students review their results on the MCQ Mock Exam and update their study plans.
- **Lesson 13: Mock FRQ Exam – Day 1**
  - **DESCRIPTION:** Students complete the first half of a mock AP CSA free response exam.
- **Lesson 14: Mock FRQ Exam – Day 2**
  - **DESCRIPTION:** Students complete the second half of a mock AP CSA free response exam.
- **Lesson 15: Mock FRQ Exam – Day 3**
  - **DESCRIPTION:** Students review their results on the FRQ Mock Exam and update their study plans.

MODULE 1

➢ **Lesson 16: Console**

➢ **Lesson 18: The Theater**

➢ **Lesson 17: The Neighborhood**

- With Paint Bucket
- No Paint Bucket