# UNIT 6

# Array

**10–15%**
AP EXAM WEIGHTING

**~6–8**
CLASS PERIODS

Remember to go to **AP Classroom** to assign students the online **Personal Progress Check** for this unit.

Whether assigned as homework or completed in class, the **Personal Progress Check** provides each student with immediate feedback related to this unit's topics and skills.

## Personal Progress Check 6

**Multiple-choice: ~15 questions**
**Free-response: 2 questions**

- Array and `ArrayList` (Array only)
- Array and `ArrayList` (Array only): partial

# Array

## Developing Understanding

This unit focuses on data structures, which are used to represent collections of related data using a single variable rather than multiple variables. Using a data structure along with iterative statements with appropriate bounds will allow for similar treatment to be applied more easily to all values in the collection. Just as there are useful standard algorithms when dealing with primitive data, there are standard algorithms to use with data structures. In this unit, we apply standard algorithms to arrays; however, these same algorithms are used with ArrayLists and 2D arrays as well. Additional standard algorithms, such as standard searching and sorting algorithms, will be covered in the next unit.

### BIG IDEA 1
**Variables** VAR

- How can programs leverage volcano data to make predictions about the date of the next eruption?

### BIG IDEA 2
**Control** CON

- How can knowing standard algorithms be useful when solving new problems?

## Building Computational Thinking Practices

3.D  4.B

Students should be able to implement program code to create, traverse, and manipulate elements in a 1D array. Traversing elements of a 1D array can be accomplished in multiple ways. Programmers need to make decisions about which loop structure is most effective given the problem they are trying to solve. Some loop structures, such as the enhanced `for` loop, only allow programmers to examine the data stored in a 1D array structure, while other loop structures allow the data to be manipulated.

Students should be able to identify and correct errors related to traversing and manipulating 1D array structures. A common run-time error that programmers experience is an out-of-bounds error, which occurs when the program tries to access an element that is beyond the range of elements in a collection. Students should double-check the values of the index being used on at least the initial and final loop iterations to ensure that they aren't out of bounds.

## Preparing for the AP Exam

A specific iterative structure is commonly used to traverse an array: starting at the beginning and moving toward the last element in the array. When students are asked to determine the result of program code that contains arrays, they often use this same iterative structure. Knowing this iterative structure can be helpful when students use tracing to determine what an algorithm is doing. Students can follow this traversal for the first few iterations and apply that pattern to the remaining elements in the array.

When preparing for the free-response questions, students should be come familiar with how existing algorithms work, rather than just memorizing the program code. Have students write out an algorithm on paper and test it using manipulatives. This allows students to experience the algorithm on a deeper level than if they simply program it. A strong understanding of how existing algorithms work allows programmers to make modifications to those algorithms to accomplish similar tasks.

# UNIT AT A GLANCE

| Enduring Understanding | Topic | Suggested Skills | Class Periods<br>~**6–8** CLASS PERIODS |
|---|---|---|---|
| VAR-2 | **6.1 Array Creation and Access** | **1.C** Determine code that would be used to interact with completed program code.<br><br>**3.D** Write program code to create, traverse, and manipulate elements in 1D array or `ArrayList` objects. | |
| | **6.2 Traversing Arrays** | **2.B** Determine the result or output based on statement execution order in a code segment without method calls (other than output).<br><br>**3.D** Write program code to create, traverse, and manipulate elements in 1D array or `ArrayList` objects.<br><br>**4.B** Identify errors in program code. | |
| | **6.3 Enhanced `for` Loop for Arrays** | **3.D** Write program code to create, traverse, and manipulate elements in 1D array or `ArrayList` objects.<br><br>**4.C** Determine if two or more code segments yield equivalent results. | |
| CON-2 | **6.4 Developing Algorithms Using Arrays** | **1.B** Determine code that would be used to complete code segments.<br><br>**3.D** Write program code to create, traverse, and manipulate elements in 1D array or `ArrayList` objects.<br><br>**5.D** Describe the initial conditions that must be met for a program segment to work as intended or described. | |

**AP** Go to **AP Classroom** to assign the **Personal Progress Check** for Unit 6. Review the results in class to identify and address any student misunderstandings.

# SAMPLE INSTRUCTIONAL ACTIVITIES

The sample activities on this page are optional and are offered to provide possible ways to incorporate instructional approaches into the classroom. They were developed in partnership with teachers from the AP community to share ways that they approach teaching some of the topics in this unit. Please refer to the Instructional Approaches section beginning on p. 159 for more examples of activities and strategies.

| Activity | Topic | Sample Activity |
|---|---|---|
| 1 | 6.1 | **Diagramming**<br>Provide students with several prompts to create and access elements in an array. After they have determined the code for each prompt, have students draw a memory diagram that shows references and the arrays they point to. Have students update the diagram with each statement to demonstrate how changing the contents through one array reference effects all the array references for this array. |
| 2 | 6.2 | **Error analysis**<br>Provide students with several error-ridden code segments containing array traversals along with the expected output of each segment. Ask them to identify any errors that they see on paper and to suggest fixes to provide the expected output. Have them type up their solutions in an IDE to verify their work. |
| 3 | 6.3 | **Think-pair-share**<br>Ask students to consider two program code segments that are meant to yield the same result: one using a traditional `for` loop and one using a `for each` loop. Have them take a few minutes to think independently about whether the two segments accomplish the same result and, if not, what changes could be made in order for that to happen. Then, ask students to work with their partners to come up with situations where it would make sense to use one type of loop over the other before sharing with the whole class. |
| 4 | 6.4 | **Pair programming**<br>Have students use pair programming to solve an array-based free-response question. Have one student be the driver for Part A while the other navigates, then have them switch for Part B. Once they are done, have partners switch solutions with another group and work through the scoring guidelines to "grade" that solution. Spend time as a class discussing the different approaches students used. |

# TOPIC 6.1
# Array Creation and Access

## Required Course Content

### ENDURING UNDERSTANDING

**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.

### LEARNING OBJECTIVE

**VAR-2.A**

Represent collections of related primitive or object reference data using one-dimensional (1D) array objects.

### ESSENTIAL KNOWLEDGE

**VAR-2.A.1**

The use of array objects allows multiple related items to be represented using a single variable.

**VAR-2.A.2**

The size of an array is established at the time of creation and cannot be changed.

**VAR-2.A.3**

Arrays can store either primitive data or object reference data.

**VAR-2.A.4**

When an array is created using the keyword `new`, all of its elements are initialized with a specific value based on the type of elements:

- Elements of type `int` are initialized to `0`
- Elements of type `double` are initialized to `0.0`
- Elements of type `boolean` are initialized to `false`
- Elements of a reference type are initialized to the reference value `null`. No objects are automatically created

**VAR-2.A.5**

Initializer lists can be used to create and initialize arrays.

## LEARNING OBJECTIVE

**VAR-2.A**

Represent collections of related primitive or object reference data using one-dimensional (1D) array objects.

## ESSENTIAL KNOWLEDGE

**VAR-2.A.6**

Square brackets (`[ ]`) are used to access and modify an element in a 1D array using an index.

**VAR-2.A.7**

The valid index values for an array are 0 through one less than the number of elements in the array, inclusive. Using an index value outside of this range will result in an `ArrayIndexOutOfBoundsException` being thrown.

## TOPIC 6.2
# Traversing Arrays

## Required Course Content

### ENDURING UNDERSTANDING

**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.

### LEARNING OBJECTIVE

**VAR-2.B**
Traverse the elements in a 1D array.

### ESSENTIAL KNOWLEDGE

**VAR-2.B.1**
Iteration statements can be used to access all the elements in an array. This is called traversing the array.

**VAR-2.B.2**
Traversing an array with an indexed `for` loop or `while` loop requires elements to be accessed using their indices.

**VAR-2.B.3**
Since the indices for an array start at 0 and end at the number of elements − 1, "off by one" errors are easy to make when traversing an array, resulting in an `ArrayIndexOutOfBoundsException` being thrown.

## TOPIC 6.3
# Enhanced for Loop for Arrays

## Required Course Content

### ENDURING UNDERSTANDING

**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.

### LEARNING OBJECTIVE

**VAR-2.C**
Traverse the elements in a 1D array object using an enhanced `for` loop.

### ESSENTIAL KNOWLEDGE

**VAR-2.C.1**
An enhanced `for` loop header includes a variable, referred to as the enhanced `for` loop variable.

**VAR-2.C.2**
For each iteration of the enhanced `for` loop, the enhanced `for` loop variable is assigned a copy of an element without using its index.

**VAR-2.C.3**
Assigning a new value to the enhanced `for` loop variable does not change the value stored in the array.

**VAR-2.C.4**
Program code written using an enhanced `for` loop to traverse and access elements in an array can be rewritten using an indexed `for` loop or a `while` loop.

TOPIC 6.4

# Developing Algorithms Using Arrays

## Required Course Content

### ENDURING UNDERSTANDING

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.

### LEARNING OBJECTIVE

CON-2.I

For algorithms in the context of a particular specification that requires the use of array traversals:
- Identify standard algorithms.
- Modify standard algorithms.
- Develop an algorithm.

### ESSENTIAL KNOWLEDGE

CON-2.I.1

There are standard algorithms that utilize array traversals to:
- Determine a minimum or maximum value
- Compute a sum, average, or mode
- Determine if at least one element has a particular property
- Determine if all elements have a particular property
- Access all consecutive pairs of elements
- Determine the presence or absence of duplicate elements
- Determine the number of elements meeting specific criteria

CON-2.I.2

There are standard array algorithms that utilize traversals to:
- Shift or rotate elements left or right
- Reverse the order of the elements