fiveable 💻 AP Computer Science A – 2023 Cheat Sheet | 👉🏽 See all AP CSA study guides | @thinkfiveable

🐒Unit 1
Primitive Types

- **byte**: a signed 8-bit integer. Its range is from -128 to 127.
- **short**: a signed 16-bit integer. Its range is from -32,768 to 32,767.
- **int**: a signed 32-bit integer. Its range is from -2,147,483,648 to 2,147,483,647.
- **long**: a signed 64-bit integer. Its range is from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.
- **float**: a single-precision 32-bit floating point **number**. Its range is from 1.4E-45 to 3.4028235E38.
- **double**: a double-precision 64-bit floating point number. Its range is from 4.9E-324 to 1.7976931348623157E308.
- **boolean**: a boolean type, which can have either a true or false value.
- **char**: a single 16-bit Unicode character. Its range is from '\u0000' (or 0) to '\uffff' (or 65,535).

📱Unit 2
Using Objects

- **Create objects:** Know how to create objects using the new keyword, initialize object fields using constructors and setters, and use the this keyword to refer to the current object.
- **Classes and objects to solve problems:** Practice creating classes and objects to model real-world concepts and use them to implement solutions to problems.
- **Inheritance and polymorphism:** Understand how to use inheritance to inherit fields and methods from a parent class, and how to use polymorphism to allow objects to be treated as instances of their parent class or any of their implemented interfaces.
- **Interfaces:** Understand how to create interfaces and use them in your code to define a set of methods that a class must implement.
- **Access modifiers:** Understand the different access modifiers (private, protected, and public) and how they control the visibility and accessibility of fields and methods in a class.
- **ArrayLists and other collections:** Understand how to use ArrayLists and other collections to store and manipulate groups of objects, including methods such as add(), remove(), and size().
- **Object methods**: Understand how to use object methods such as equals(), hashCode(), and toString() to compare objects, generate hash codes, and convert objects to strings.
- **Static methods and variables:** Understand how to use static methods and variables to define methods and fields that are associated with a class rather than with objects of the class.

✅Unit 3
Boolean Expressions & if Statements

- **Boolean expressions** evaluate to either true or false, and are used in if statements to determine whether to execute a block of code. Make sure you understand how to create boolean expressions using comparison operators *(such as ==, !=, <, >, <=, and >=) and logical operators (such as &&, ||, and !).*
- **If statements**: if statements are used to control the flow of execution in a program. Make sure you understand how to use if statements to execute different blocks of code based on different conditions.
- **Else and else if statements:** else and else if statements can be used to execute different blocks of code when the condition in the if statement is false.
- **Nest if statements:** You can nest if statements inside other if statements to test for multiple conditions.
- **Short-circuit evaluation**: When using logical operators such as && and ||, the right-hand side of the expression may not be evaluated if the left-hand side determines the outcome of the expression. This is known as short-circuit evaluation.
- **Ternary operator:** The ternary operator (?:) can be used as a shorthand for if-else statements when assigning a value to a variable.
- **Switch statements:** switch statements can be used as an alternative to if-else statements when testing for multiple conditions on a single variable.
- **Boolean variables**: boolean variables can be used to simplify boolean expressions and make code easier to read and understand.

🕹️Unit 4
Iteration

- **Understand the different types of loops:** 3 types of loops in Java: while loops, do-while loops, and for loops.
- **While loops and do while loops:** while loops - used when you want to execute a block of code repeatedly while a certain condition is true. Do-while loops - similar to while loops, but guarantee that the loop body will execute at least once before checking the loop condition.
- **For loops:** Are used when you want to execute a block of code a fixed number of times, or when iterating over a collection of objects.
- **Continue and break statements:** continue is used to skip over a single iteration of a loop, while break is used to exit the loop entirely.
- **Nested loops:** Can be used to iterate over multiple dimensions of an array, or to generate all possible combinations of a set of variables.
- **The scope of loop variables**: Loop variables declared in a for loop are only visible within the loop body, while variables declared outside the loop can be accessed from anywhere in the program.
- **Efficient algorithms:**When solving problems with loops, be sure to use efficient algorithms to minimize the number of loop iterations required.

⚙️Unit 5
Writing Classes

- **The concept of classes and objects:** A class is a template or blueprint for creating objects, which are instances of a class. Make sure you understand the difference between a class and an object, and how to create objects in Java.
- **Define class attributes and methods:** Class attributes are variables that define the state of an object, while class methods are functions that define the behavior of an object. Make sure you understand how to define class attributes and methods, and their data types and return types.
- **Use constructors:** Constructors are special methods that are used to initialize the state of an object when it is created. Make sure you understand how to define constructors and their parameters.
- **Use access modifiers:** Access modifiers such as public, private, and protected are used to control the visibility of class attributes and methods. Make sure you understand how to use access modifiers to control access to class members.
- **Use inheritance:** Inheritance allows you to create new classes based on existing classes. This helps you to reuse code and to create a hierarchy of classes. Make sure you understand how to create subclasses and how to override methods inherited from a superclass.

⌚Unit 6
Array

- **The difference between an array and an ArrayList**: An array is a fixed-size collection of elements of the same type, while an ArrayList is a dynamic-size collection of elements that can be of different types. Make sure you understand the differences between these two data structures and how to use them in Java.
- **Declare and initialize arrays:** Arrays are declared with a specific size and type. Make sure you understand how to declare and initialize arrays in Java, and how to access and modify their elements.
- **Use loops with arrays:** Loops are often used with arrays to iterate over their elements. Make sure you understand how to use for and while loops with arrays in Java.
- **Use ArrayLists:** ArrayLists are a more flexible alternative to arrays, allowing you to add, remove, and modify elements dynamically. Make sure you understand how to declare and initialize ArrayLists, and how to use the various methods to manipulate them.
- **Use enhanced for loops:** Enhanced for loops are a convenient way to iterate over the elements of an array or an ArrayList. Make sure you understand how to use enhanced for loops in Java.
- **Use multidimensional arrays**: Multidimensional arrays are arrays of arrays. Make sure you understand how to declare, initialize, and access elements of multidimensional arrays in Java.

💾Unit 7
ArrayList

- **Different types of searching and sorting algorithms**: There are various searching and sorting algorithms such as linear search, binary search, selection sort, insertion sort, bubble sort, and merge sort. Make sure you understand the concept behind each algorithm, and how to implement them in Java.
- **Know when to use each algorithm:** Each algorithm has its own advantages and disadvantages, and is more suited to certain situations. For example, binary search is more efficient than linear search for large datasets. Make sure you understand when to use each algorithm.
- **Analyze the time and space complexity of algorithms**: The time and space complexity of an algorithm determines its efficiency in terms of time and memory usage. Make sure you understand how to analyze the time and space complexity of algorithms, and how to compare them.
- **Practice implementing algorithms**: Practice implementing searching and sorting algorithms in Java, and make sure you understand how to write efficient and correct code.
- **Understand recursion**: Some searching and sorting algorithms use recursion, which is a programming technique where a function calls itself. Make sure you understand how recursion works, and how to use it to implement algorithms such as binary search and merge sort.
- **Use test cases**: Use test cases to test your code and ensure that it works correctly. Make sure you understand how to write test cases that cover all possible scenarios.

💻Unit 8
2D Array

- **Recursion**: Recursion is a programming technique where a function calls itself. Make sure you understand how recursion works, and how it can be used to solve problems.
- **Identify base cases and recursive cases:** Recursive functions typically have two parts: base cases and recursive cases. Base cases are the terminating conditions that stop the recursion, while recursive cases are the conditions that call the function recursively. Make sure you understand how to identify base cases and recursive cases in problems.
- **Visualize the call stack**: When a recursive function is called, each call is added to a call stack. Make sure you understand how the call stack works, and how to visualize it to better understand recursive functions.
- **Understand recursion with arrays and ArrayLists**: Recursion can be used to solve problems with arrays and ArrayLists. Make sure you understand how to use recursion to search and sort arrays and ArrayLists.
- **Practice solving problems with recursion:** Practice using recursion to solve problems, and make sure you understand how to write efficient and correct recursive functions.
- **Avoid infinite recursion**: Infinite recursion occurs when a function calls itself indefinitely. Make sure you understand how to avoid infinite recursion, and how to debug recursive functions if they do not terminate.

👨‍👧‍👦Unit 9
Inheritance

- **Understand the concept of two-dimensional arrays:** Two-dimensional arrays are arrays that have both rows and columns. Make sure you understand how to declare and initialize two-dimensional arrays in Java.
- **Access elements in a two-dimensional array:** To access elements in a two-dimensional array, you need to specify the row and column index. Make sure you understand how to access elements in a two-dimensional array using nested loops.
- **Manipulate two-dimensional arrays:** You can manipulate two-dimensional arrays by changing the values of individual elements, swapping rows or columns, and transposing the array. Make sure you understand how to manipulate two-dimensional arrays to solve problems.
- **Practice solving problems with two-dimensional arrays:** Practice using two-dimensional arrays to solve problems, such as searching and sorting, and matrix operations.
- **Understand the relationship between two-dimensional arrays and nested loops:** Two-dimensional arrays are often used in conjunction with nested loops. Make sure you understand how to use nested loops to iterate through a two-dimensional array.
- **Understand irregular two-dimensional arrays**: An irregular two-dimensional array is an array where each row can have a different number of elements. Make sure you understand how to declare and manipulate irregular two-dimensional arrays.

💯FRQ Tips

Read the instructions carefully: Make sure to read the instructions and requirements for each question carefully. Follow the prompts and answer all parts of the question **//** **Manage your time**: You will have 1 hour and 30 minutes to complete 4 FRQs **//** **Use proper syntax**: Make sure to use proper syntax and format your code correctly. This includes using appropriate naming conventions, indentation, and commenting your code as needed **//** **Show your work:** Include comments in your code to explain why you made certain decisions and how your code works **//** **Use good programming practices:** Use good programming practices such as modularity and abstraction to break down complex problems into smaller, more manageable parts **//** **Understand the scoring rubric:** Familiarize yourself with the scoring rubric for each question. The rubric outlines the specific requirements for each question and the points allocated for each part. Review and revise: Review your answers and revise them as needed. Check for any errors or mistakes and make sure that your code is well-organized and readable **//** **Don't leave anything blank:** Even if you are unsure of the answer to a question, make an attempt to answer it. Partial credit can be awarded for correct answers or reasonable attempts