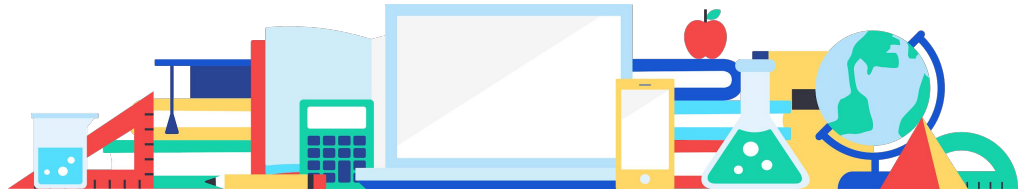
AP Computer Science A - Concepts and Vocab

September 2024



Unit 1: Primitive Types (2.5 - 5%)

Primitive Types: (2.5 - 5%)

```
public class Main {
  public static void main(String[] args) {
    System.out.print("Hi! ");
    System.out.println("Hello!");
    System.out.println("Bye!");
  }
}
```

[AP Computer Science in Java]

Why Programming? Why Java?

- ➔ **Students should know the basics of Java and higher-level programming levels**
- ➔ **Students should be able to print to the console**

Variables & Primitive Data Types

Students should know the various types in Java

- ➔ **Primitive Types**
  - ◆ Integers - whole numbers or counting numbers
  - ◆ Doubles - decimal numbers
  - ◆ Char - represent a single character
  - ◆ Boolean - hold a true or false value

- ➔ **Reference Types**
  - ◆ Types created using primitive types

Students should know how to use the final keyword

- ➔ Protects variables from alteration
- ➔ `final int val = 8`

Primitive Types: (2.5 - 5%)

Operator	Description
+	Addition operator (Ex. 1 + 1 = 2)
-	Subtraction operator (Ex. 2 - 1 = 1)
*	Multiplication operator (Ex. 2 * 2 = 4)
/	Division operator (Ex. 6 / 2 = 3)
%	Modulus operator (Ex. 10 % 3 = 1)

AP Computer Science in Java

Expressions & Assignment Statements

Students should know how to use Arithmetic expressions

- ➔ +, -, *, /, and %

Students should know the different types of division

- ➔ Integer Division
  - ◆ Dividing two integers, returns an integer value. i.e 5 / 2 = 2
- ➔ Double Division
  - ◆ Dividing two doubles, returns a double. i.e 5.0/2.0 = 2.5
- ➔ Mixed Division
  - ◆ Divides a double by an integer or vice versa. The return value is a double.

Arithmetic Exception

- ➔ An arithmetic exception is thrown when dividing by zero.

Modulus Operators

- ➔ Allows you to divide two numbers and get the remainder.

Operation Precedence

- ➔ Order of precedence still applies:
  - ◆ 1st: Parentheses
  - ◆ 2nd: Multiplication & division
  - ◆ 3rd: Addition & subtraction

Primitive Types: (2.5 - 5%)

Original	Shortcut	Description
counter = counter + 1;	counter++;	Increment a variable by 1
counter = counter - 1;	counter–;	Subtract 1 from a variable
x = x + y	x += y	Adding values to a variable
x = x - y	x -= y	Subtracting values from a variable
x = x * y;	x *= y	Multiplying values to a variable
x = x / y;	x /= y	Dividing values from a variable

[AP Computer Science in Java]

Compound Assignment Operators

Evaluating What is Stored in a Variable

- ➔ A variable can store the outcome of expressions
- ➔ Variables can also use their own existing value to change the value of the variable.

Students should know how to use arithmetic operators and shortcuts

Casting & Ranges of Variables

Students should be able to cast types into another

How Casting Works

- ➔ To turn one primitive type into another, add the type in between parentheses to cast it.

Division with Casting

- ➔ Dividing two integers will always return an integer - we can get the correct answer by casting one of the variables to a double.

Unit 2: Using Objects (5 - 7.5%

Using Objects: (5 - 7.5%)

AP Computer Science in Java

Objects: Instances of Classes

Students should know what objects and classes are are

- **Objects** are a **reference type**. This means that when we refer to them in code, we refer to where it is stored in memory.
- A **class** is like a template that defines what an object is like and what the object can do.

Creating and Storing Objects (Instantiation)

Students should know the basics about objects

- **Instance variables, or attributes,** hold the state of the object.
- **Instantiation** and **null objects**
- The **constructor** is used to instantiate – create an instance of – an object.

Students should know how to create and use methods

- **Methods** are procedures that allow us to control and define the behavior of an object We use a parameter list to list the values that are passed. The return type passes a value back out of the method. If a method has no return type it is **void**.

Using Objects: (5 - 7.5%)

AP Computer Science in Java

String Objects: Concatenation, Literals & More

Students should be able to create, concatenate, and use strings

- A **String** is a sequence of characters.
  - A **string literal** is a string inside of double quotes, as you saw above.
  - The **String constructor** takes a string literal as an input
- **Concatenation** is joining or linking two things together to form one.
  - You can concatenate Strings with other Strings and with other primitive types like char, boolean, and double.
  - In doing so, the reference data type's **toString()** method gets called and concatenated to the existing String.
- Escape sequences can be used to include quotation marks in a string In Java

Students should know APIs, Libraries, and Documentation

- An **API** is typically used to interface two systems and make using them easier.
- **Libraries** tend to be extensions of the main programming language that can help simplify code
- Libraries can be broken down into a series of **packages**.

Using Objects: (5 - 7.5%)

Wrapper Class Methods	Use
`Integer.MIN_VALUE`	Returns the minimum possible int or Integer value
`Integer.MAX_VALUE`	Returns the maximum possible int or Integer value
`varInt.intValue()`	Converts Integer value into an int value
`varDbl.doubleValue()`	Converts Double value into a double value

[AP Computer Science in Java]

Wrapper Classes: Integer and Double

Students should understand wrapper classes

- ➔ A **wrapper class** in Java is a class that contains or “wraps” primitive data types as an object.They can take primitive data types and **convert them into an object** and **provide static methods that allow you to perform some basic number operations**, such as converting data from a string to a number.

Students should understand Autoboxing and Unboxing

- ➔ **Autoboxing** is the automatic conversion that the Java compiler makes between primitive types and their corresponding object wrapper classes.
- ➔ **Unboxing** is the automatic conversion that the Java compiler makes from the wrapper class to the primitive type.

Using Objects: (5 - 7.5%)

AP Computer Science in Java

Using the Math Class

Students should be able to perform basic math operations

Method	Use
`Math.abs(x)`	Returns the absolute value of x. This can take either an `int` or a `double`.
`Math.pow(base, exponent)`	Returns the value of base raised to the power of exponent.
`Math.sqrt(x)`	Returns the positive square root of x.

Random Numbers

- ➔ The **random method** takes no input and returns a random number from zero to one, inclusive of zero, but not inclusive of one.
- ➔ **Multiply** - By multiplying the results of Math.random(), you can create values from 0 up to (but not including) the number you multiple by.
- ➔ **Add** - Adding to the results offsets the range that you will get from the random number.
- ➔ **Casting to an integer** - Finally, if you cast the value to an integer, you will truncate the value and only have random integers..

Unit 3: Expressions and If-statements (15-17.5%)

Expressions and if-statements (15-17.5%)

AP Computer Science in Java

Boolean expressions and equality checks

Boolean operators and comparison operators

- And (&&), Not (!), Or (||)
- Comparison operators
  - Greater than (>) or equal to (>=), Less than (<) or equal to (<=)
  - Equal (==) and not equal (!=)
- A **common pitfall** for students is to confuse the *assignment operator* (=) with the equals operator (==)
- A **common point of confusion** is determining when equality comparisons should be made with == vs. the function .equals()
  - == is OK to determine functional equivalence of *primitive types*
  - When used on objects, == is checking that the references point to the same location in memory

Expressions and if-statements (15-17.5%)

KSU: Java Order of Operations (With examples)

UCSD Java Operator Precedence (Cheat sheet)

Runestone Academy Compound Boolean Expressions

Compound boolean expressions

Compound boolean expressions are boolean expressions chained together by boolean operators like AND (&&) and OR (||)

- Example: “n is greater than 0 and less than 10”
  - In code, this is: `n > 0 && n < 10`
- **Order of operations applies!** See UCSD Java Operator Precedence cheat sheet

If-statements and Control Flow

Key points

- If statements can stand on their own, but Else-if and Else statements require an If statement to come first
- If and Else-if statements are always accompanied by a condition, but Else-statements are not

Common pitfall: Two if-statements vs If-Else-if

- Having two if-statements side by side is *not necessarily equivalent* to an If-Else-if statement
- Two independent if-statements are each evaluated in turn, thus the code within each could be executed in turn
- Conversely, a single construction composed of if-else if will only execute *up to 1* body of code

Expressions and if-statements (15-17.5%)

[W3 Schools: Java If ... Else]

[GeeksForGeeks: Decision Making in Java]

Anatomy of if statements

Anatomy of an If-statement in Java

```
If (condition) {
        // Body - This code is only executed if “condition” evaluates to true
}
```

Anatomy of if-else statements

```
if (condition) {
        // Code that executes if “condition” evaluates to true
} else {
        // Code that executes if “condition” evaluates to false
}
```

Anatomy of if-else if statements

```
if (condition 1) {
        // A: Code that evaluates if “condition 1” evaluates to true
} else if (condition 2) {
        // B: Code that evaluates if “condition 2” evaluates to true
}
```

Note: Students should be reminded that it is possible that neither A nor B is executed, if condition 1 and condition 2 are both false

Unit 4: Iteration (17.5 - 22.5%)

Iteration (17.5-22.5%)

Loops in Java
What is Big O Notation Explained

Iteration

Students need to be able to iterate with different kinds of loops

- ➔ Iteration is the process of repeating a set of instructions until a condition is met.
- ➔ In Java, this is accomplished using a control flow known as a loop.
- ➔ Loops can be nested

Students should be aware of common pitfalls

- ➔ Infinite loops (loops that never terminate and lead to errors) and off-by-one errors (often due to forgetting about **zero-indexing**) are common

Students should be able to assess high-level algorithm performance

- ➔ **“Big O Notation”** is a form of informal code analysis

Anatomy of while Loops

```
while (condition) {
    // The code here repeats while condition is true.
}
```

Anatomy of for Loops

```
for (initialization; condition; increment/decrement) {
    // The code here repeats while condition is true.
}
```

Iteration (17.5-22.5%)

How will this skill be tested?

Key points

- In the exam, students will need to know
  - Boolean expressions/conditions
  - How to terminate a loop early
  - How to determine the code segment that will cause the iterative program to have the intended behavior
  - Informal code analysis: How to determine how many times a loop will execute, compare the runtime of different iterative programs, Big O Notation

AP CS A Iteration Unit, p. 78

Unit 5: Writing Classes (5 - 7.5%)

Writing Classes (5 - 7.5%)

AP Computer Science in Java

Access, state, and methods

Students need to understand data encapsulation and access

- **The `private` and `public` keywords** determine whether a class, property, or method are accessible outside of their defining context
- Classes should typically be declared public since they need to be accessed by other classes. Likewise, a constructor in the class should always be public since it needs to be called by external classes.
- **Data encapsulation** is a technique in which the implementation of details of the class are kept hidden from the user. By using the private level access, instance variables are encapsulated in the class.
- **Getters** are methods that access object state

Object State

- The state of an object represents the attributes and their values at a given time and is defined by instance variables belonging to the object.
  - These instance variables create a “has-a” relationship.
- **Setters** are methods that mutate object state

Constructors

- Constructors set the initial state of an object. Java uses a no-argument constructor by default when no class constructor is provided

Writing Classes (5 - 7.5%)

AP Computer Science in Java
Comments in Java

Documentation and Methods

Students should practice documenting their code

- ➔ Students are encouraged to practice documenting their code through comments. Comments may be written as single-line (**//**) or multi-line (**/* */**) or Javadoc (**/** */**)
- ➔ Students may also document their methods with **preconditions** and **postconditions** to describe the expected starting state and expected outcome of some code

Students must be able to write methods

- ➔ Methods may have a return type or be **void**
- ➔ Methods may be defined with *parameters;* when passed to a method call, these are referred to as *arguments*
- ➔ When a primitive is given as the argument, the method is provided a copy. When the argument is an object reference, the actual object reference is passed.
  - ◆ **Best practice:** Students should be careful not to mutate the argument unless specified in the method's description
- ➔ A **static method or static variable** is associated with the class but not any class instance

Writing Classes (5 - 7.5%)

‘This’ and Scope

Students will need to understand the ‘this’ keyword

- ➔ ‘this’ is a reference to the current object– the object whose method or constructor is being called
- ➔ For non-static methods and constructors only

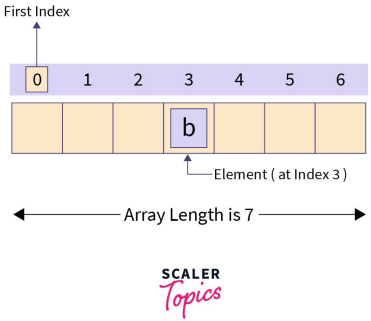
Students should be mindful of scope

- ➔ Students will likely have been introduced to the concept of scope already, but may need a reminder when moving onto Java classes
- ➔ Students should understand that:
  - ◆ Class methods are scoped to the class and unknown outside of the context of the class
  - ◆ Instance variables belong to the class instance / object
  - ◆ Method parameters are scoped to the method

AP Computer Science in Java
Comments in Java

Unit 6: Arrays (10-15%)

Arrays



Source: Scaler

CodeHS: Introduction to Arrays

Array basics

Students should understand the structure of an array

- ➔ Each *element* in the array is stored at an *index* of the array
- ➔ Arrays are 0-indexed, meaning the first position is considered index 0
- ➔ Arrays have a property, ***length***, which is the number of elements the array may contain

Students must be able to traverse arrays

- ➔ For-loop, while loop, for-each loop, or do-while loop
  - ◆ Note: do-while loops are not on the AP exam
- ➔ Students may start with in-order traversal starting at index 0 to get familiar

Arrays

Common Pitfalls

A common pitfall for students is attempting to access an index of an array that does not exist.

- ➔ This will cause an **ArrayIndexOutOfBoundException** to be thrown
- ➔ This is usually due to “off-by-one” errors
  - ◆ Example: Student tries to access element at myArr[myArr.length]
    - Since arrays are zero-indexed, the last element is actually stored at myArr[myArr.length - 1]

Array length is accessed via the *property* length, not a method length()

- ➔ Correct: int arrLength = myArr.length;
- ➔ Incorrect: int arrLength = myArr.length();

Arrays

Developing algorithms using arrays

Students should be able to identify which types of problems commonly use arrays

- ➔ These are typically problems that are concerned with a collection of values

Students may practice common array problems

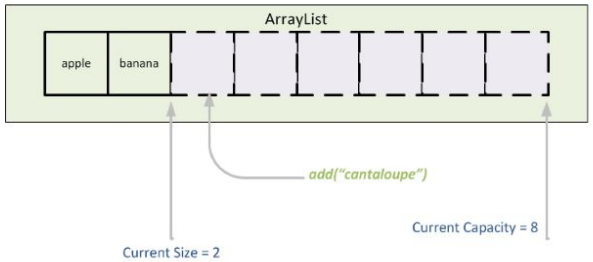
- ➔ What is the sum/average of all values in this array?
- ➔ How many times does this value occur in the array?
- ➔ How can we reverse an array?
- ➔ How can we shift an array left or right?

Common algorithms can be found here

Developing Algorithms Using Arrays - Fivable

Unit 7: ArrayList (2.5-7.5%)

ArrayList (2.5-7.5%)



From Emmanuel Abiola

AP Computer Science in Java, ArrayLists

Comparison to Arrays

Unlike Arrays, ArrayLists can be extended

- An array has only the space for the number of elements it is declared with
- **ArrayLists can dynamically increase their size** as more elements are added
- ArrayLists have both capacity and size
  - **Capacity:** The underlying number of cells allocated for the ArrayList
  - **Size:** The actual number of elements in the cells

Traversal

- Just like Arrays, ArrayLists can be traversed with for loops, enhanced for loops, and while loops. See examples here.

ArrayList (2.5-7.5%)

AP Computer Science in Java, ArrayLists vs. Arrays

ArrayList vs. Array

Basic differences in usage

	Array	ArrayList
Get length	arr.length;	arrList.size();
Get value at index	arr[index];	arrList.get(index);
Set value at index	arr[index] = val;	arrList.set(index, val);
Instantiation	int[] myArr = new int[5];	ArrayList<Integer> myArrList = new ArrayList<Integer>(5)
Removing elements	Set index to null int[index] = null;	arrList.remove(index);
Adding elements	This is just setting elements	arrList.add(index); // to end of list

ArrayList (2.5-7.5%)

AP Computer Science in Java

Common pitfalls and skills

IndexOutOfBoundsException

- ➔ Like arrays, if you attempt to access an element outside of the range [0, number of elements - 1], an exception will be thrown

Adding or removing elements while traversing the ArrayList

- ➔ This requires special methods, otherwise a **ConcurrentModificationException** will be thrown
- ➔ This is because when elements are removed in an ArrayList, the elements that follow are shifted down
- ➔ You need to do index-based removal (i.e. use a regular for loop instead of an enhanced for loop/for-each loop)
- ➔ See more info here

Assessing performance

- ➔ Students may start to work with larger datasets, making array traversal and storage a concern. Students should practice determining how many times an array is accessed or a loop is run and should be able to determine the Big-O complexity of an algorithm

ArrayList (2.5-7.5%)

Searching and sorting

Students should be familiar with common search and sort algorithms

- **Students should NOT memorize these algorithms;** they will need to understand them conceptually and may want to visualize them
- **Selection sort:** Repeatedly search the unsorted portion of the array for the smallest value, and move it into place. See more info.
  - Time complexity: O(n^2)
- **Insertion sort:** Separate set into a sorted and unsorted portion, and place each element from the unsorted portion into the sorted portion one at a time. See more info.
  - Time complexity: O(n^2)
- **Linear/sequential search:** Walk through the array or ArrayList in order until you find the target element, or until you have reached the end
  - Time complexity: O(n)

Sort algorithm visualizations

GeeksForGeeks, Selection Sort

GeeksForGeeks, InsertionSort

ArrayList (2.5-7.5%)

Ethical issues around data collection

The AP exam is looking to see that a student can:

- ➔ Explain:
  - ◆ Privacy risks involved in collecting and storing personal data
- ➔ Understand:
- ➔ That programmers have a responsibility to try to protect personal data
  - That programs can both assist with and compromise personal security.
- ➔ It may help to point the student to helpful articles and video resources. One place to start is with CSAwesome's *Ethics of Data Collection and Data Privacy*.

CSAwesome, Ethics of Data Collection and Data Privacy

CodeHS

Unit 8: 2D Arrays (7.5-10%)

2D Arrays (7.5-10%)

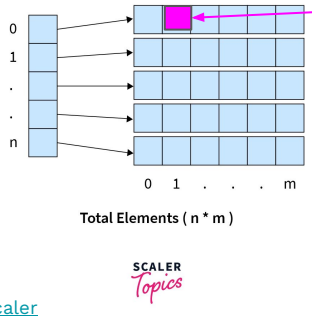


Image from Scaler

AP Computer Science in Java

The structure of a 2D array

Students should understand structure, access, and traversal of 2D arrays

- ➔ An array where the elements are themselves arrays
- ➔ Elements are accessed in **row-major order** (row, then column)
- ➔ Example: my2DArr[0][1] access the element in the *0th row, 1st column*

Creation and access

- ➔ Example: int[][] my2DArr = new int[numRows][numCols];
  - ◆ In the diagram on the left, numRows=n and numCols=m
- ➔ Example: my2DArr[n-1][m-1]
  - ◆ In the diagram on the left, this accesses the final (bottom-right) element of the 2D array

Tip

- ➔ Conceptually, it helps to envision a 2D array as a matrix with rows/columns
- ➔ But in reality, the data structure is a simple array where each element *points to* another array

2D Arrays (7.5-10%)

Traversal

Row-major traversal example

```
int rows = 4;
int cols = 3;
for (int r = 0; r < rows; r++) {
        for (int c = 0; c < cols; c++) {
                System.out.println(my2DArr[r][c]);
        }
}
```

Column-major traversal

Same thing, but iterating on columns in the outer level and rows in the inner level

GeeksForGeeks, Multidimensional Arrays in Java

2D Arrays (7.5-10%)

Tips for helping students with 2D arrays

What's on the exam:

- ➔ The AP exam is only concerned with 1D and 2D arrays. While more dimensions are possible, they will not be on the exam.
- ➔ The AP exam is only concerned with rectangular 2D arrays

Students should leverage what they learned from 1D arrays

- ➔ The standard algorithms used to search and sort 1D arrays still apply for 2D arrays

https://codehs.gitbooks.io/apjava/content/Data-Structures/2d-arrays.html

Unit 9: Subclasses and Inheritance (5 - 10%)

Subclasses and Inheritance
5 - 10%

Students should understand the hierarchy and polymorphism of classes

- ➔ Allows for a **hierarchy** of classes where common code is moved into superclasses
  - ◆ E.g. Hound $\xrightarrow{super}$ Dog $\xrightarrow{super}$ Animal
- ➔ Subclassing is done through the **extends** keyword
- ➔ **Overriding** allows you to override definitions from the superclass, i.e. a new method implementation
- ➔ Parent classes can have multiple children (subclasses), but each class can only have one parent
- ➔ The **super()** method invokes the superclass, and this is commonly seen in subclass constructors
- ➔ **Object** is the superclass of all other classes in Java and exposes functions toString() and equals(Object o)

Polymorphism

- ➔ **Polymorphism** is the ability for a single object to take many forms. This manifests as:
  - ◆ Method overriding
  - ◆ Up- or down-casting
  - ◆ Dynamic binding (i.e. Animal animal = new Fox(); Fox castedFox = (Fox) animal;) and static binding

CodeHS - Inheritance

Polymorphism - CollegeBoard (Detailed)

CodeHS - Polymorphism (Concise)

Unit 10: Recursion
5 - 7%

Recursion
5-7.5%

What's on the exam:

- ➔ Students **will not** have to **write** recursive code on the exam, but they will need to be able to answer multiple choice questions about recursive code

Students should know the key elements of a recursive algorithm:

- ➔ The basic structure of a recursive method includes a **recursive call** and a **base case**
- ➔ Students may practice rewriting familiar code and algorithms (such as search and sort algorithms) recursively. Some good examples are:
  - ◆ Binary search
  - ◆ Merge sort
- ➔ Students should practice tracing through the recursive method calls all the way until the algorithm terminates
- ➔ Students may use online algorithm visualizations to better understand how recursive algorithms like mergesort work

CodeHS - Recursion

Algorithm Visualizations

Exam

Exam

Overview AP Exams

What is AP?

Advanced Placement is a specific type of course created by the nonprofit organization College Board. Students take these courses to prepare for college level coursework.

What is an AP exam & why is it important?

Students are able to take a national exam in April or May to demonstrate their mastery of course content. **These exams are important because, depending on the student's score they can be granted college credit for the course!** Usually public colleges and universities require a 3 out of 5, while other colleges may require a 4 or 5. Some high school teachers also incorporate a student's AP score into their final grade.

Exam

College Board AP Computer Science A Exam Format OVerview

AP Computer Science Principles Exam

Format

- 3 hour exam in May
- Section 1 (90 minutes): 40 multiple choice questions (50% of exam score)
  - Mostly individual questions, with one or two sets of multiple questions (typically two to three questions per set).
  - Computational Thinking Practices 1, 2, 4, and 5 are assessed.
- Section 2 (90 minutes): 4 questions, free response (50% of exam score)
  - All free-response questions assess Computational Thinking Practice 3: Code Implementation, with the following focus:
  - Question 1: Methods and Control Structures (9 points)
  - Question 2: Class (9 points).
  - Question 3: Array/ArrayList (9 points).
  - Question 4: 2D Array (9 points).

Scoring

- The exam is a scaled score that is converted into 1-5 final score
- There is no penalty for wrong answers for multiple choice questions, so students should not leave blank any questions.