AP® Computer Science A Syllabus

Course Overview

The content and objectives of AP Computer Science A course include the course objectives as described in the AP Computer Science Course Description. This course focuses on an object-oriented approach to problem-solving using Java. It includes the study of common algorithms and the use of some of Java's built-in classes and interfaces for basic data structures.

I expect all my students to take the AP Computer Science A Examination. The students and I work hard during the year to assure that every student has an opportunity to achieve a qualifying score on the exam. Students' course grades correlate strongly with their AP Examination grades.

Curricular Requirements

CTP Skills:

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

Big Ideas

MODULARITY (MOD) Incorporating elements of abstraction, by breaking problems down into interacting pieces, each with their own purpose, makes writing complex programs easier. Abstracting simplifies concepts and processes by looking at the big picture rather than being overwhelmed by the details. Modularity in object-oriented programming allows us to use abstraction to break complex programs down into individual classes and methods.

VARIABLES (VAR) Information used as a basis for reasoning, discussion, or calculation is referred to as data. Programs rely on variables to store data, on data structures to organize multiple values when program complexity increases, and on algorithms to sort, access, and manipulate this data. Variables create data abstractions, as they can represent a set of possible values or a group of related values.

CONTROL (CON) Doing things in order, making decisions, and doing the same process multiple times are represented in code by using control structures and specifying the order in which instructions are executed. Programmers need to think algorithmically in order to define and interpret processes that are used in a program.

IMPACT OF COMPUTING (IOC) Computers and computing have revolutionized our lives. To use computing safely and responsibly, we need to be aware of privacy, security, and ethical issues. As programmers, we need to understand how our programs will be used and be responsible for the consequences.

Teaching Strategies

I believe that computer science is more about problem-solving and logical thinking that it is about computers. With this in mind, I try to encourage students with a variety of interests to challenge themselves by taking the class. I strive to create a learning environment that is comfortable for all students. I encourage students to ask questions and challenge them with questions that promote discussion. I also encourage students to ask each other questions and to work cooperatively. This promotes leadership and teamwork skills, which are so vital in today’s society.

Students work on programs in class about 60% of the time. These include small programs that cover a specific topic (called **`codeIt!` Nows**), modifying larger programs, or creating a large program of their own from design through testing. Since we are using jGrasp, which is a free download, the students are able to work on these programs at home if necessary.

Labs

Writing computer programs is essential for understanding the applications of data structures and algorithms, in the JAVA programming language. I assign at least two or three long program labs per unit, which is to be completed in their respective IDE individually or at times with a partner. Many students complete the labs during class, but there are a few who may need to work on it at home or come before or after school during open lab hours to complete the work.

I have also integrated the AP Computer Science A Labs: Magpie, Elevens, and the Picture lab into this course throughout a variety of units to aid in the minimum of 20 hours of hands-on lab work. The Magpie lab aids in developing the students’ skills with conditional and nested conditional statements. The Picture lab is incorporated into the unit on ArrayLists. The Elevens labs are used throughout a variety of units towards the end of the course to stress and reinforce the object-oriented paradigm that is discussed throughout the curriculum.

The students complete all of the required activities of the AP Computer Science A labs as a part of the lab component of this course.

Course Assessments

The teacher will have formative assessments to ensure students understand the material that has been taught. They are divided into two categories. The first category is ungraded and consists of student participation, student responses to questions, observed student-student interactions and program writing completion (**`codeIt!` Nows**). Questions and programs are incorporated within each daily lesson, and students use a response device to submit answers anonymously. These results are used to spark discussion and debate between students in a social constructivist environment. These questions also provide teachers and students instant feedback to gauge the pace of instruction and the need for reteaching.

The second category is graded and consists of quizzes, labs, and long program writing (handwritten and IDE based), based on the previously discussed problem sets and material in the presentations. Summative assessments are graded and take the form of unit tests. These are all given in the same form as the AP exam: half multiple choice, half free response which includes error analysis and program construction. The multiple-choice questions are conceptual while the free response section involves solving multistep problems; often similar to questions from prior AP exams.

The intention is for identical summative assessments to be given to all the students in the course on the same day, regardless of their teacher. This is to encourage students to study together in groups, with or without a teacher, to advance their skill and understanding. Previous AP Exam questions will often be incorporated to expose students to AP type problems.

Texts and Supplemental Materials

This syllabus corresponds to a digital curriculum: The New Jersey Center for Teaching and Learning (NJCTL) Progressive Science Initiative® (PSI®): AP Computer Science A. Course resources including Presentations, Problem Sets, Quizzes, Tests, and Unit Plans can be accessed at https://njctl.org/courses/computer-science/ap-computer-science/.

NOTE: The course materials located on the NJCTL website are accepted as equivalent to a college textbook for this and other AP courses.

Required Software Downloads:

Oracle JDK: https://www.oracle.com/technetwork/java/javase/downloads/jdk12-downloads-5295953.html

jGrasp IDE: https://spider.eng.auburn.edu/user-cgi/grasp/grasp.pl?;dl=download_jgrasp.html

Course Planner

This course is taught using jGrasp. This Integrated Development Environment (IDE) is free to download. I find that seamless incorporation of an IDE frequently in a classroom is an integral and motivating aspect at the beginning of programming.

The Topic column in the tables below lists the subjects covered in the daily lectures. These topics are based on the components of the AP exam. The Resources column includes a link to SMART Notebook presentations, labs & activities used during the class, as well as assessments. Below, there is an Objectives column that lays outs the skills and abilities students will gain during the unit.

Unit 1: Fundamentals of Programming *(2-3 Weeks)*

Aligns to the Collegeboard Unit 1

Topics	**Resources, Labs, & Assessments**
● Safety and Security in Computer Use ● Introduction to Programming ● Hello, World	**Resources:**

- Basic Syntax & Output
- Escape Characters
- Comments
- Primitive Data Types
- Variables
- Syntax Errors in Programs
- Typecasting
- Input
- The String type

- Materials can be found on NJCTL website at [Unit 1]
    - Classwork-Homework Activities **(Big Idea - Impact of Computing) (Big Idea - Variables)**

Assessments:
- Formative Assessments
- codeIt! Nows
- Graded Programming Quizzes
- Unit Test

Objectives

Curricular Requirements Satisfied:

- Big Idea #1, MODULARITY (MOD) Students will design simple variables and statements including variables that are the fundamental basis for modularizing code by breaking a large task down into smaller skills. There are variable activities through the codeItNows, classwork-homework activities, quizzes and the unit test where students practice this skill **(Skill 1.A)**.
- Big Idea #2, VARIABLES (VAR) Students develop the skills and apply their knowledge of declaring, initializing, and manipulating Java variables within a program. Students will reuse variables and adjust their stored values to use variables in their most beneficial way **(Skill 1.B)**.
- Big Idea #3, CONTROL (CON) Throughout the direct instruction, group work activities, and individual practice students will develop the coding skills and documentation etiquette to useScanner objects to gather input and control how it is used and manipulated with arithmetic and logical operators **(Skill 2.A, Skill 5.A)**. Students will also design and evaluate the output of programs within a Java class and a main() method thus allowing the program to be executed **(Skill 2.B)**. Students will be able to debug basic Java programs that incorporate a main() method and variables **(Skill 4.B, Skill 5.B)**.
- Big Idea #4, IMPACT OF COMPUTING (IOC) During the classwork-homework activities, students will research and describe security practices used in storing data and maintaining privating when using a computing device.

- Describe the functionality and use of program code through comments.
- Comments are ignored by the compiler and are not executed when the program is run.
- Three types of comments in Java include /* */, which generates a block of comments, //, which generates a comment on one line, and /** */, which are Javadoc comments and are used to create API documentation.
- Call System class methods to generate output to the console.
- System.out.print and System.out. println display information on the computer monitor.
- System.out.println moves the cursor to a new line after the information has been displayed, while System.out.print does not.
- Create string literals.
- A string literal is enclosed in double quotes.
- Identify the most appropriate data type category for a particular specification.
- A type is a set of values (a domain) and a set of operations on them.
- Data types can be categorized as either primitive or reference.
- The primitive data types used in this course define the set of operations for numbers and Boolean values.
- Declare variables of the correct types to represent primitive data.
- The three primitive data types used in this course are int, double, and Boolean.
- Each variable has associated memory that is used to hold its value.

• The memory associated with a variable of a primitive type holds an actual primitive value. • When a variable is declared final, its value cannot be changed once it is initialized. • Evaluate arithmetic expressions in a program code. • A literal is the source code representation of a fixed value. • Arithmetic expressions include expressions of type int and double. • The arithmetic operators consist of +, −, *, /, and %. • An arithmetic operation that uses two int values will evaluate to an int value. • An arithmetic operation that uses a double value will evaluate to a double value. • Operators can be used to construct compound expressions. • During evaluation, operands are associated with operators according to operator precedence to determine how they are grouped. • An attempt to divide an integer by zero will result in an ArithmeticException to occur. • Evaluate what is stored in a variable as a result of an expression with an assignment statement. • The assignment operator (=) allows a program to initialize or change the value stored in a variable. The value of the expression on the right is stored in the variable on the left. • During execution, expressions are evaluated to produce a single value. • The value of an expression has a type based on the evaluation of the expression. • Evaluate arithmetic expressions that use casting. • The casting operators (int) and (double) can be used to create a temporary value converted to a different data type. • Casting a double value to an int causes the digits to the right of the decimal point to be truncated. • Some programming code causes int values to be automatically cast (widened) to double values. • Values of type double can be rounded to the nearest integer by (int)(x + 0.5) or (int)(x – 0.5) for negative numbers. • If an expression would evaluate to an int value outside of the allowed range, an integer overflow occurs. This could result in an incorrect value within the allowed range.

Unit 2: Control Statements & Loops *(2-3 Weeks)*
Aligns to the Collegeboard Units 3 & 4

Topics	**Resources, Labs, & Assessments**
● Assignment & Increment Operators ● Rational & Logical Operators ● The if-else Statement ● Nested if-else Statements ● The while Loop ● The for Loop ● Nested for & while Loops	**Resources:** ● Materials can be found on NJCTL website at Unit 2 ○ Classwork-Homework Activities **(Big Idea - Control)** **Labs:** `LowestFactor` Program **(CTP 2 Skill B)** ● In the `LowestFactor` Program, students write a program within a `main()` method to gather two integers greater than one and use arithmetic and logical operators in the program code to find and output the lowest common factor of the two numbers. `OddDiamond` Program `CashRegister` Program **Assessments:**

	● Formative Assessments ● codeIt! Nows ● Graded Programming Quizzes ● Unit Test

Objectives

Curricular Requirements Satisfied:

- Big Idea #3, CONTROL (CON) Students will manipulate the way variables and operators are sequenced and combined in an expression to determine a computed result. This can occur through the integration of iterative and selection Java structures that students will incorporate into their code to allow the computer to process each of the many possible input values **(Skill 5.C)**. Prior to compiling or executing their program code, students will trace the iterative and selective statements to determine what the code will output with or without methods other than the main() method **(Skill 2.B, Skill 2.C, Skill 3.C)**. Programs will be written from scratch and from stub code where students need to incorporate if/else, while, and for loops **(Skill 1.B)**. Iterative algorithms to count, sum, find averages, find specific values will be written and students will create and evaluate the number of times loops will run prior to the code's execution **(Skill 2.D)**. Students will compare how while loops versus for loops are written and executed to determine the best way to write iterative algorithms into their code **(Skill 4.C)**.

• Evaluate what is stored in a variable as a result of an expression with an assignment statement.
• Compound assignment operators (+=, −=, *=, /=, %=) can be used in place of the assignment operator
• The increment operator (++) and decrement operator (−−) are used to add 1 or subtract 1 from the stored value of a variable or an array element. The new value is assigned to the variable or array element.
• Evaluate Boolean expressions that use relational operators in program code.
• Primitive values and reference values can be compared using relational operators (i.e., == and !=).
• Arithmetic expression values can be compared using relational operators (i.e., , <=, >=).
• An expression involving relational operators evaluates to a Boolean value.
• Evaluate compound Boolean expressions in program code.
• Logical operators !(not), &&(and), and ||(or) are used with Boolean values. This represents the order these operators will be evaluated.
• An expression involving logical operators evaluates to a Boolean value.
• When the result of a logical expression using && or || can be determined by evaluating only the first Boolean operand, the second is not evaluated. This is known as short-circuited evaluation.
• Represent branching logical processes by using conditional statements.
• Conditional statements interrupt the sequential execution of statements.
• if statements affect the flow of control by executing different statements based on the value of a Boolean expression.
• A one-way selection (if statement) is written when there is a set of statements to execute under a certain condition. In this case, the body is executed only when the Boolean condition is true.
• A two-way selection is written when there are two sets of statements— one to be executed when the Boolean condition is true, and another set for when the Boolean condition is false. In this case, the body of the "if" is executed when the Boolean condition is true, and the body of the "else" is executed when the Boolean condition is false.
• A multi-way selection is written when there are a series of conditions with different statements for each condition. Multi-way selection is performed using if-else-if statements such that exactly one section of code is executed based on the first condition that evaluates to true.
• Represent branching logical processes by using nested conditional statements.
• Nested if statements consist of if statements within if statements.

• Represent iterative processes using a while loop. • Iteration statements change the flow of control by repeating a set of statements zero or more times until a condition is met. • In loops, the Boolean expression is evaluated before each iteration of the loop body, including the first. When the expression evaluates to true, the loop body is executed. This continues until the expression evaluates to false, whereupon the iteration ceases. • A loop is an infinite loop when the Boolean expression always evaluates to true. • If the Boolean expression evaluates to false initially, the loop body is not executed at all. • Executing a return statement inside an iteration statement will halt the loop and exit the method or constructor. • For algorithms in the context of a particular specification that does not require the use of traversals: Identify standard algorithms, Modify standard algorithms, Develop an algorithm. • There are standard algorithms to: Identify if an integer is or is not evenly divisible by another integer, Identify the individual digits in an integer, Determine the frequency with which a specific criterion is met • There are standard algorithms to: Determine a minimum or maximum value, Compute a sum, average, or mode. • Represent iterative processes using a for loop. • There are three parts in a for loop header: the initialization, the Boolean expression, and the increment. The increment statement can also be a decrement statement. • In a for loop, the initialization statement is only executed once before the first Boolean expression evaluation. The variable being initialized is referred to as a loop control variable. • In each iteration of a for loop, the increment statement is executed after the entire loop body is executed and before the Boolean expression is evaluated again. • A for loop can be rewritten into an equivalent while loop and vice versa. • “Off by one” errors occur when the iteration statement loops one time too many or one time too few. • For algorithms in the context of a particular specification that involves String objects: Identify standard algorithms, Modify standard algorithms, Develop an algorithm. • There are standard algorithms that utilize String traversals to: Find if one or more substrings has a particular property, Determine the number of substrings that meet specific criteria, Create a new string with the characters reversed • Represent nested iterative processes. • Nested iteration statements are iteration statements that appear in the body of another iteration statement. • When a loop is nested inside another loop, the inner loop must complete all its iterations before the outer loop can continue. • Compute statement execution counts and informal run-time comparison of iterative statements. • A statement execution count indicates the number of times a statement is executed by the program.

Unit 3: Strings *(2 Weeks)*

Aligns to the Collegeboard Unit 2

Topics	**Resources, Labs, & Assessments**
● String Objects ● Immutability of Strings ● Using Strings ● Numeric Values Within a String ● Programming with Objects in Mind	**Resources:** ● Materials can be found on NJCTL website at Unit 3

	○ Classwork-Homework Activities **(Big Idea - Modularity)** **Labs:** `LetterChange` Program `WordCount` Program `DefaultPswd` Program `CaesarCipher` Program **Assessments:** ● Formative Assessments ● codeIt! Nows ● Graded Programming Quizzes ● Unit Test

Objectives

Curricular Requirements Satisfied:

- Big Idea #2, VARIABLES (VAR) Students will use variables to store input values as well as to count and sum values in various algorithms. Primitive types will be parsed from String objects and students will store the parsed values in variables to be manipulated with arithmetic, selective, and iterative algorithms **(Skill 2.A)**. Students will locate, debug, and document errors in their code as they work to manipulate Strings **(Skill 4.B, Skill 5.B)** and compare ways to determine the most efficient ways to extract information from and use Strings **(Skill 4.C)**.
- Big Idea #3, CONTROL (CON) Using String objects, students will traverse and manipulate the strings to locate characters, indices, and substrings within the Java strings. Selective and iterative algorithms will be written and documented to manipulate the String objects **(Skill 5.A)**. Students will write unique methods practicing creating arguments and passing parameters into methods to get a desired output from the method **(Skill 1.B)**. With the uniquely created methods, students will practice invoking the methods correctly from the main() method using dot notation and objects **(Skill 1.C, Skill 2.C, Skill 3.A)**.

• Define variables of the correct types to represent reference data.
• The keyword null is a special value used to indicate that a reference is not associated with any object.
• The memory associated with a variable of a reference type holds an object reference value or, if there is no object, null. This value is the memory address of the referenced object.
• For String class: a. Create String objects. b. Call String methods.
• String objects can be created by using string literals or by calling the String class constructor.
• String objects are immutable, meaning that String methods do not change the String object.
• String objects can be concatenated using the + or += operator, resulting in a new String object.
• Primitive values can be concatenated with a String object. This causes implicit conversion of the values to String objects.
• Escape sequences start with a \ and have a special meaning in Java. Escape sequences used in this course include \”, \\, and \n.
• Application program interfaces (APIs) and libraries simplify complex programming tasks.
• Documentation for APIs and libraries are essential to understanding the attributes and behaviors of an object of a class.
• Classes in the APIs and libraries are grouped into packages.
• The String class is part of the java.lang package. Classes in the java.lang package are available by default.
• A String object has index values from 0 to length – 1. Attempting to access indices outside this range will result in an IndexOutOfBoundsException.

• A String object can be concatenated with an object reference, which implicitly calls the referenced object's toString method. • The following String methods and constructors—including what they do and when they are used—are part of the Java Quick Reference: a. String(String str) — Constructs a new String object that represents the same sequence of characters as str b. int length() — Returns the number of characters in a String object c. String substring(int from, int to) — Returns substring beginning at index from andending at index (to – 1) d. String substring(int from) — Returns substring(from, length()) e. int indexOf(String str) — Returns the index of the first occurrence of str; returns -1 if not found f. boolean equals(String other) — Returns true if this is equal to other; returns false otherwise g. int compareTo(String other) — Returns a value < 0 if this is less than other; returns zero if this is equal to other; returns a value > 0 if this is greater than other • A string identical to the single element substring at position index can be created by calling substring(index, index + 1)

Unit 4: Arrays *(2 Weeks)*

Aligns to the Collegeboard Units 6 & 8

Topics	**Resources, Labs, & Assessments**
• What is an Array • Array Elements & Loops • Scope & Programming • Two-Dimensional Arrays	**Resources:** • Materials can be found on NJCTL website at Unit 4 **Labs:** `LowestToGreatest` Program `NameLength` Program `SortRandomNumbers` Program **(CTP 3 Skill D)** • In the `SortRandomNumbers` Program, students will create four methods each with a one dimensional array as an argument. The methods will be designed to calculate the sum, average, and find the minimum and maximum value of the values in the array. AP Lab #1: MagPie Lab **(CTP 4 Skill A)** • In the MagPie Lab during Activity 4, students are tasked with tracing the `findKeyword()` method with various inputs and recording stored values in variables. **Assessments:** • Formative Assessments • codeIt! Nows • Graded Programming Quizzes • Unit Test

Objectives

Curricular Requirements Satisfied:

- Big Idea #2, VARIABLES (VAR) To manage the large amounts of data or complex relationships in data, students will write code that groups the data together into a single data structure without creating individual variables for each value **(Skill 3.D)**. Students will create various Java types of arrays to hold integers, doubles, Strings, characters, and booleans throughout the classwork, formative, and summative assessments **(Skill 1.B)**. Students will evaluate and document how code executes within a main() method by tracing each code statement **(Skill 2.B, Skill 5.A)**.
- Big Idea #3, CONTROL (CON) Throughout the direct instruction, group work activities, and individual practice students will integrate selective and iterative control structures to traverse one dimensional and two dimensional arrays **(Skill 3.E)**. Through the traversal of arrays, students will gather values stored at individual index values, collect sub arrays, and sort or search through the values in an array **(Skill 2.D)**. By working with different types of iterative algorithms such as while, for, or enhanced for loops, students will be able to compare the best and most efficient algorithm for a given purpose **(Skill 4.C)**. Students would create the appropriate methods and calls to methods to run the code that interacts with the newly introduced arrays **(Skill 1.C)**. Students will use various method calls with different test cases to debug program code and to determine how algorithms involved in traversing arrays, execute **(Skill 4.A, Skill 4.B, (Skill 5.B)**. Students will also code and document through comments, initial conditions for their program to include that must be met in order for the code to execute correctly **(Skill 5.D)**.

- Represent collections of related primitive or object reference data using one dimensional (1D) array objects.
- The use of array objects allows multiple related items to be represented using a single variable.
- The size of an array is established at the time of creation and cannot be changed.
- Arrays can store either primitive data or object reference data.
- 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:

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

- Initializer lists can be used to create and initialize arrays.
- Square brackets ([ ]) are used to access and modify an element in a 1D array using an index.
- 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.
- Traverse the elements in a 1D array.
- Iteration statements can be used to access all the elements in an array. This is called traversing the array.
- Traversing an array with an indexed for loop or while loop requires elements to be accessed using their indices.
- 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.
- Traverse the elements in a 1D array object using an enhanced for loop.
- An enhanced for loop header includes a variable, referred to as the enhanced for loop variable.
- For each iteration of the enhanced for loop, the enhanced for loop variable is assigned a copy of an element without using its index.

• Assigning a new value to the enhanced for loop variable does not change the value stored in the array.
• 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.
• Represent collections of related primitive or object reference data using two-dimensional (2D) array objects.
• 2D arrays are stored as arrays of arrays. Therefore, the way 2D arrays are created and indexed is similar to 1D array objects.
a. EXCLUSION STATEMENT—2D array objects that are not rectangular are outside the scope of the course and AP Exam.
• For the purposes of the exam, when accessing the element at arr[first][second], the first index is used for rows, the second index is used for columns.
• The initializer list used to create and initialize a 2D array consists of initializer lists that represent 1D arrays.
• The square brackets [row][col] are used to access and modify an element in a 2D array.
• “Row-major order” refers to an ordering of 2D array elements where traversal occurs across each row, while “column-major order” traversal occurs down each column.
• For 2D array objects: Traverse using nested for loops and Traverse using nested enhanced for loops.
• Nested iteration statements are used to traverse and access all elements in a 2D array. Since 2D arrays are stored as arrays of arrays, the way 2D arrays are traversed using for loops and enhanced for loops is similar to 1D array objects.
• Nested iteration statements can be written to traverse the 2D array in “row-major order” or “column-major order.”
• The outer loop of a nested enhanced for loop used to traverse a 2D array traverses the rows. Therefore, the enhanced for loop variable must be the type of each row, which is a 1D array. The inner loop traverses a single row. Therefore, the inner enhanced for loop variable must be the same type as the elements stored in the 1D array
• 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.
• There are standard algorithms that utilize array traversals to:
a. Determine a minimum or maximum value
b. Compute a sum, average, or mode
c. Determine if at least one element has a particular property
d. Determine if all elements have a particular property
e. Access all consecutive pairs of elements
f. Determine the presence or absence of duplicate elements
g. Determine the number of elements meeting specific criteria
• There are standard array algorithms that utilize traversals to:
a. Shift or rotate elements left or right
b. Reverse the order of the elements
• For algorithms in the context of a particular specification that requires the use of 2D array traversals Identify standard algorithms, Modify standard algorithms, Develop an algorithm.
• When applying sequential/linear search algorithms to 2D arrays, each row must be accessed then sequential/linear search applied to each row of a 2D array.
• All standard 1D array algorithms can be applied to 2D array objects.

Unit 5: Classes *(3 weeks)*
Aligns to the Collegeboard Units 2 & 5

Topics	**Resources, Labs, & Assessments**

- Design and Structure of Classes
- Passing Parameters & Variables
- Public, Private & Static
- Encapsulation
- Methods & Constructors

Resources:

- Materials can be found on NJCTL website at Unit 5

Labs:

`CarTrip` Program **(CTP 1 Skill A)**

- In the `CarTrip` Program, students are asked to design a CarTrip class with all of the necessary fields, constructors, and methods to store and determine the population and favorite sports team in the city.

`CityDriver` Program

`Coin` Program

Assessments:

- Formative Assessments
- codeIt! Nows
- Graded Programming Quizzes
- Unit Test

Objectives

Curricular Requirements Satisfied:

- Big Idea #1, MODULARITY (MOD) Students will use code to represent a physical object or nonphysical concept, real or imagined, by defining a class based on the attributes and/or behaviors of the object or concept **(Skill 1.A, Skill 5.A).** The objects created by students in their programs will represent different types and students will practice creating uniquely typed objects versus the standard typed objects created prior to this unit **(Skill 3.B)**. Multiclass programs will be designed or modified by students based on related and modularized attributes and behaviors throughout all activities and forms of assessment **(Skill 1.B)**. The toString() method will be used often by students, to demonstrate how some methods are so frequently used that they can override existing toString() methods to write solutions more quickly **(Skill 2.C)**.
- Big Idea #2, VARIABLES (VAR) Students will incorporate variables into their classes as global fields to expand the previous capabilities of their classes. Arguments in methods will be used as variables in methods to hold passed information. Students will apply these new pieces of class information into their own programs **(Skill 1.C)** and determine errors in method calls or creation, field instantiation, and constructor creation **(Skill 4.B, Skill 5.D)**.
- Big Idea #3, CONTROL (CON) Students will write object and driver classes to expand their program's capabilities and organize or modularize their code **(Skill 3.A)**. Within students' object and driver classes, selection and iteration algorithms including various types of arithmetic and logical operators will be written within methods and constructors to perform the functions needed **(Skill 2.A)**. Students will document through comments the parts of the classes and how they function **(Skill 5.A, Skill 5.B)**.

- Explain the relationship between a class and an object.
- An object is a specific instance of a class with defined attributes.
- A class is the formal implementation, or blueprint, of the attributes and behaviors of an object.
- Identify, using its signature, the correct constructor being called.
- A signature consists of the constructor name and the parameter list.
- The parameter list, in the header of a constructor, lists the types of the values that are passed and their variable names. These are often referred to as formal parameters.

- A parameter is a value that is passed into a constructor. These are often referred to as actual parameters.
- Constructors are said to be overloaded when there are multiple constructors with the same name but a different signature.
- The actual parameters passed to a constructor must be compatible with the types identified in the formal parameter list.
- Parameters are passed using call by value. Call by value initializes the formal parameters with copies of the actual parameters.
- For creating objects: a. Create objects by calling constructors without parameters. b. Create objects by calling constructors with parameters.
- Every object is created using the keyword new followed by a call to one of the class's constructors.
- A class contains constructors that are invoked to create objects. They have the same name as the class.
- Existing classes and class libraries can be utilized as appropriate to create objects.
- Parameters allow values to be passed to the constructor to establish the initial state of the object.
- Call non-static void methods without parameters.
- An object's behavior refers to what the object can do (or what can be done to it) and is defined by methods.
- Procedural abstraction allows a programmer to use a method by knowing what the method does even if they do not know how the method was written.
- A method signature for a method without parameters consists of the method name and an empty parameter list.
- A method or constructor call interrupts the sequential execution of statements, causing the program to first execute the statements in the method or constructor before continuing. Once the last statement in the method or constructor has executed or a return statement is executed, flow of control is returned to the point immediately following where the method or constructor was called.
- Non-static methods are called through objects of the class.
- The dot operator is used along with the object name to call non-static methods.
- Void methods do not have return values and are therefore not called as part of an expression.
- Using a null reference to call a method or access an instance variable causes a NullPointerException to be thrown.
- Call non-static void methods with parameters.
- A method signature for a method with parameters consists of the method name and the ordered list of parameter types.
- Values provided in the parameter list need to correspond to the order and type in the method signature.
- Methods are said to be overloaded when there are multiple methods with the same name but a different signature.
- Call non-static non-void methods with or without parameters.
- Non-void methods return a value that is the same type as the return type in the signature. To use the return value when calling a non-void method, it must be stored in a variable or used as part of an expression.
- Call static methods.
- Static methods are called using the dot operator along with the class name unless they are defined in the enclosing class.
- Designate access and visibility constraints to classes, data, constructors, and methods.
- The keywords public and private affect the access of classes, data, constructors, and methods.
- The keyword private restricts access to the declaring class, while the keyword public allows access from classes outside the declaring class.
- Classes are designated public.
- Access to attributes should be kept internal to the class. Therefore, instance variables are designated as private.
- Constructors are designated public.

- Access to behaviors can be internal or external to the class. Therefore, methods can be designated as either public or private
- Define instance variables for the attributes to be initialized through the constructors of a class.
- An object’s state refers to its attributes and their values at a given time and is defined by instance variables belonging to the object. This creates a “has-a” relationship between the object and its instance variables.
- Constructors are used to set the initial state of an object, which should include initial values for all instance variables.
- Constructor parameters are local variables to the constructor and provide data to initialize instance variables.
- 4 When a mutable object is a constructor parameter, the instance variable should be initialized with a copy of the referenced object. In this way, the instance variable is not an alias of the original object, and methods are prevented from modifying the state of the original object.
- When no constructor is written, Java provides a no-argument constructor, and the instance variables are set to default values.
- Describe the functionality and use of program code through comments.
- Comments are ignored by the compiler and are not executed when the program is run.
- Three types of comments in Java include /* */, which generates a block of comments, //, which generates a comment on one line, and /** */, which are Javadoc comments and are used to create API documentation.
- A precondition is a condition that must be true just prior to the execution of a section of program code in order for the method to behave as expected. There is no expectation that the method will check to ensure preconditions are satisfied.
- A postcondition is a condition that must always be true after the execution of a section of program code. Postconditions describe the outcome of the execution in terms of what is being returned or the state of an object.
- Programmers write method code to satisfy the postconditions when preconditions are met.
- Define behaviors of an object through non-void methods without parameters written in a class.
- An accessor method allows other objects to obtain the value of instance variables or static variables.
- A non-void method returns a single value. Its header includes the return type in place of the keyword void.
- In non-void methods, a return expression compatible with the return type is evaluated, and a copy of that value is returned. This is referred to as “return by value.”
- When the return expression is a reference to an object, a copy of that reference is returned, not a copy of the object.
- The return keyword is used to return the flow of control to the point immediately following where the method or constructor was called.
- The toString method is an overridden method that is included in classes to provide a description of a specific object. It generally includes what values are stored in the instance data of the object.
- If System.out.print or System.out. println is passed an object, that object’s toString method is called, and the returned string is printed.
- Define behaviors of an object through void methods with or without parameters written in a class.
- A void method does not return a value. Its header contains the keyword void before the method name.
- A mutator (modifier) method is often a void method that changes the values of instance variables or static variables.
- Define behaviors of an object through non-void methods with parameters written in a class.
- Methods can only access the private data and methods of a parameter that is a reference to an object when the parameter is the same type as the method’s enclosing class.
- Non-void methods with parameters receive values through parameters, use those values, and return a computed value of the specified type.

- It is good programming practice to not modify mutable objects that are passed as parameters unless required in the specification.
- When an actual parameter is a primitive value, the formal parameter is initialized with a copy of that value. Changes to the formal parameter have no effect on the corresponding actual parameter.
- When an actual parameter is a reference to an object, the formal parameter is initialized with a copy of that reference, not a copy of the object. If the reference is to a mutable object, the method or constructor can use this reference to alter the state of the object.
- Passing a reference parameter results in the formal parameter and the actual parameter being aliases. They both refer to the same object.
- Define behaviors of a class through static methods.
- Static methods are associated with the class, not objects of the class.
- Static methods include the keyword static in the header before the method name.
- Static methods cannot access or change the values of instance variables.
- Static methods can access or change the values of static variables.
- Static methods do not have a this reference and are unable to use the class's instance variables or call non-static methods.
- Define the static variables that belong to the class.
- Static variables belong to the class, with all objects of a class sharing a single static variable.
- Static variables can be designated as either public or private and are designated with the static keyword before the variable type.
- Static variables are used with the class name and the dot operator, since they are associated with a class, not objects of a class.
- Designate private visibility of instance variables to encapsulate the attributes of an object.
- Data encapsulation is a technique in which the implementation details of a class are kept hidden from the user.
- When designing a class, programmers make decisions about what data to make accessible and modifiable from an external class. Data can be either accessible or modifiable, or it can be both or neither.
- Instance variables are encapsulated by using the private access modifier.
- The provided accessor and mutator methods in a class allow client code to use and modify data.
- For wrapper classes: a. Create Integer objects. b. Call Integer methods. c. Create Double objects. d. Call Double methods.
- The Integer class and Double class are part of the java.lang package.
- The following Integer methods and constructors—including what they do and when they are used—are part of the Java Quick Reference:

a. Integer(int value) — Constructs a new Integer object that represents the specified int value
b. Integer.MIN_VALUE — The minimum value represented by an int or Integer
c. Integer.MAX_VALUE — The maximum value represented by an int or Integer
d. int intValue() — Returns the value of this Integer as an int

- The following Double methods and constructors—including what they do and when they are used—are part of the Java Quick Reference:

a. Double (double value) – Constructs a new Double object that represents the specified double value
b. double doubleValue() — Returns the value of this Double as a double

- Autoboxing is the automatic conversion that the Java compiler makes between primitive types and their corresponding object wrapper classes. This includes converting an int to an Integer and a double to a Double.
- The Java compiler applies autoboxing when a primitive value is:

a. Passed as a parameter to a method that expects an object of the corresponding wrapper class.
b. Assigned to a variable of the corresponding wrapper class.

- Unboxing is the automatic conversion that the Java compiler makes from the wrapper class to the primitive type. This includes converting an Integer to an int and a Double to a double.
- The Java compiler applies unboxing when a wrapper class object is:

a. Passed as a parameter to a method that expects a value of the corresponding primitive type. b. Assigned to a variable of the corresponding primitive type. • Explain where variables can be used in the program code. • Local variables can be declared in the body of constructors and methods. These variables may only be used within the constructor or method and cannot be declared to be public or private. • When there is a local variable with the same name as an instance variable, the variable name will refer to the local variable instead of the instance variable. • Formal parameters and variables declared in a method or constructor can only be used within that method or constructor. • Through method decomposition, a programmer breaks down a large problem into smaller subproblems by creating methods to solve each individual subproblem. • Evaluate object reference expressions that use the keyword this. • Within a non-static method or a constructor, the keyword this is a reference to the current object—the object whose method or constructor is being called. • The keyword this can be used to pass the current object as an actual parameter in a method call. • Evaluate expressions that use the Math class methods. • The Math class is part of the java.lang package. • The Math class contains only static methods. • The following static Math methods—including what they do and when they are used—are part of the Java Quick Reference: a. int abs(int x) — Returns the absolute value of an int value b. double abs(double x) — Returns the absolute value of a double value c. double pow(double base, double exponent) — Returns the value of the first parameter raised to the power of the second parameter d. double sqrt(double x) — Returns the positive square root of a double value e. double random() — Returns a double value greater than or equal to 0.0 and less than 1.0 f. The values returned from Math.random can be manipulated to produce a random int or double in a defined range. • Explain the ethical and social implications of computing systems. • System reliability is limited. Programmers should make an effort to maximize system reliability. • Legal issues and intellectual property concerns arise when creating programs. • The creation of programs has impacts on society, economies, and culture. These impacts can be beneficial and/or harmful.

Unit 6: ArrayLists *(2.5 weeks)*

Aligns to the Collegeboard Unit 7

Topics	**Resources, Labs, & Assessments**
● Changing Sizes of Arrays ● ArrayLists ● ArrayLists & Wrapper Classes ● Iteration ● ArrayList Methods	**Resources:** ● Materials can be found on NJCTL website at Unit 6 **Labs:** `DeckofCards` Program **Assessments:** ● Formative Assessments ● codeIt! Nows

	● Graded Programming Quizzes ● Unit Test

Objectives

Curricular Requirements Satisfied:

- Big Idea #2, VARIABLES (VAR) Students will learn and use ArrayLists to manage large amounts of data or complex relationships in data **(Skill 3.D)**. The ArrayLists will group the data together into a single data structure without creating individual variables for each value, although students will develop Java code to work with the traversal and manipulation of ArrayLists. Through the use of variables, students will be able to store the number of times an algorithm traverses an ArrayList **(Skill 2.D)**, run test cases to ensure proper function **(Skill 4.A)**, and modify the code to change the final outcome. **(Skill 5.C)**.
- Big Idea #3, CONTROL (CON) Students will write iterative and selective algorithms to traverse ArrayLists and gather individual or small subsets of data from the ArrayList **(Skill 1.B)**. Through written algorithms, students' code will be able to manipulate ArrayLists to store and output the desired results. Students will also write methods that pass ArrayLists as arguments and practice calling the methods in a `main()` to determine how the code executes **(Skill 2.C)**.

• Represent collections of related object reference data using ArrayList objects.
• An ArrayList object is mutable and contains object references.
• The ArrayList constructor ArrayList() constructs an empty list.
• Java allows the generic type ArrayList, where the generic type E specifies the type of the elements.
• When ArrayList is specified, the types of the reference parameters and return type when using the methods are type E.
• ArrayList is preferred over ArrayList because it allows the compiler to find errors that would otherwise be found at run-time.
• The ArrayList class is part of the java. util package. An import statement can be used to make this class available for use in the program.
• The following ArrayList methods— including what they do and when they are used—are part of the Java Quick Reference:
a. int size() - Returns the number of elements in the list
b. boolean add(E obj) - Appends obj to end of list; returns true
c. void add(int index, E obj) - Inserts obj at position index (0 <= index <= size), moving elements at position index and higher to the right (adds 1 to their indices) and adds 1 to size
d. E get(int index) - Returns the element at position index in the list
e. E set(int index, E obj) — Replaces the element at position index with obj;returns the element formerly at position index
f. E remove(int index) — Removes element from position index, moving elements at position index + 1 and higher to the left (subtracts 1 from their indices) and subtracts 1 from size; returns the element formerly at position index
• For ArrayList objects: Traverse using a for or while loop and Traverse using an enhanced for loop.
• Iteration statements can be used to access all the elements in an ArrayList. This is called traversing the ArrayList.
• Deleting elements during a traversal of an ArrayList requires using special techniques to avoid skipping elements.
• Since the indices for an ArrayList start at 0 and end at the number of elements − 1, accessing an index value outside of this range will result in an ArrayIndexOutOfBoundsException being thrown.

• Changing the size of an ArrayList while traversing it using an enhanced for loop can result in a ConcurrentModificationException being thrown. Therefore, when using an enhanced for loop to traverse an ArrayList, you should not add or remove elements. • For algorithms in the context of a particular specification that requires the use of ArrayList traversals: Identify standard algorithms, Modify standard algorithms, Develop an algorithm. • There are standard ArrayList algorithms that utilize traversals to: Insert elements, Delete elements, and apply the same standard algorithms that are used with 1D arrays. • Some algorithms require multiple String, array, or ArrayList objects to be traversed simultaneously • Apply sequential/linear search algorithms to search for specific information in array or ArrayList objects. • There are standard algorithms for searching. • Sequential/linear search algorithms check each element in order until the desired value is found or all elements in the array or ArrayList have been checked.

Unit 7: Inheritance & Polymorphism *(3 weeks)*
Aligns to the Collegeboard Unit 9

Topics	**Resources, Labs, & Assessments**
● Class Hierarchy ● Superclass & Subclass ● Overriding & Overloading ● Inheritance ● Expending a Superclass ● Polymorphism ● Type Compatibility	**Resources:** ● Materials can be found on NJCTL website at Unit 7 **Labs:** `WarGame` Program **(Big Idea - Modularity)** ● The `WarGame` program will be designed by students focusing on the concept of modularity. Students will need to create methods to enact the actions that can occur in the card game War. Students will have a driver class that will run the War game and play through the possible actions. **Assessments:** ● Formative Assessments ● codeIt! Nows ● Graded Programming Quizzes ● Unit Test

Objectives
Curricular Requirements Satisfied: ● Big Idea #1, MODULARITY (MOD) Students will create multiple classes containing common attributes and behaviors and will use the `extends` keyword to denote an inheritance program **(Skill 1.A, Skill 3.A)**. Students' classes will contain and inherit shared attributes and behaviors to form a hierarchy. Students will design and create hierarchical programs and determine the difference between inheritance programs and association programs. After creating their inheritance programs, students will design and document code to implement classes in an inheritance to create and use objects referencing classes in an

inheritance **(Skill 1.C, Skill 5.A)**. By creating objects within an inheritance class, students will apply type casting of objects **(Skill 3.B)** and explain why certain code will or will not work **(Skill 5.D)**.

- Create an inheritance relationship from a subclass to the superclass.
- A class hierarchy can be developed by putting common attributes and behaviors of related classes into a single class called a superclass.
- Classes that extend a superclass, called subclasses, can draw upon the existing attributes and behaviors of the superclass without repeating these in the code.
- Extending a subclass from a superclass creates an "is-a" relationship from the subclass to the superclass.
- The keyword extends is used to establish an inheritance relationship between a subclass and a superclass. A class can extend only one superclass.
- Constructors are not inherited.
- The superclass constructor can be called from the first line of a subclass constructor by using the keyword super and passing appropriate parameters.
- The actual parameters passed in the call to the superclass constructor provide values that the constructor can use to initialize the object's instance variables.
- When a subclass's constructor does not explicitly call a superclass's constructor using super, Java inserts a call to the superclass's no-argument constructor.
- Regardless of whether the superclass constructor is called implicitly or explicitly, the process of calling su-perclass constructors continues until the Object constructor is called. At this point, all of the constructors within the hierarchy execute beginning with the Object constructor.
- Method overriding occurs when a public method in a subclass has the same method signature as a public method in the superclass.
- Any method that is called must be defined within its own class or its superclass.
- A subclass is usually designed to have modified (overridden) or additional methods or instance variables.
- A subclass will inherit all public methods from the superclass; these methods remain public in the subclass.
- The keyword super can be used to call a superclass's constructors and methods.
- The superclass method can be called in a subclass by using the keyword super with the method name and passing appropriate parameters.
- Define reference variables of a superclass to be assigned to an object of a subclass in the same hierarchy.
- When a class S "is-a" class T, T is referred to as a superclass, and S is referred to as a subclass.
- If S is a subclass of T, then assigning an object of type S to a reference of type T facilitates polymorphism.
- If S is a subclass of T, then a reference of type T can be used to refer to an object of type T or S.
- Declaring references of type T, when S is a subclass of T, is useful in the following declarations:

Formal method parameters and arrays — T[ ] var ArrayList var

- Call methods in an inheritance relationship.
- Utilize the Object class through inheritance.
- At compile time, methods in or inherited by the declared type determine the correctness of a non-static method call.
- At run-time, the method in the actual object type is executed for a non-static method call.
- Call Object class methods through inheritance.
- The Object class is the superclass of all other classes in Java.
- The Object class is part of the java.lang package
- The following Object class methods and constructors—including what they do and when they are used—are part of the Java Quick Reference:

a. boolean equals(Object other)
b. String toString()

• Subclasses of Object often override the equals and toString methods with classspecific implementations.

Unit 8: Recursion ***(2 weeks)***
Aligns to the Collegeboard Unit 10

Topics	**Resources, Labs, & Assessments**
● Recursion ● The Base Case ● Tracing Recursion ● Two-Directional Recursion ● Recursion w/ Objects ● Recursion Beyond AP	**Resources:** ● Materials can be found on NJCTL website at Unit 8 **Labs:** `Lucas Tower` Program **Assessments:** ● Formative Assessments ● codeIt! Nows ● Graded Programming Quizzes ● Unit Test **(CTP 5 Skill A on the free response question)**
Objectives	
Curricular Requirements Satisfied: ● Big Idea #3, CONTROL (CON) Students will write recursive algorithms designed to recursively step through a program to obtain an output. Students will compare their recursive algorithm to previously created iterative algorithms, determine how each functions, how many times an algorithm will execute **(Skill 2.D)**, and how quickly each can perform a given task. Students will write and fill in recursive statements included within recursive methods **(Skill 1.B)**. While working with recursive statements, students will trace code and method calls to determine how their program will execute **(Skill 2.C)** and explain how they work **(Skill 5.A)**. • Determine the result of executing recursive methods. • A recursive method is a method that calls itself. • Recursive methods contain at least one base case, which halts the recursion, and at least one recursive call. • Each recursive call has its own set of local variables, including the formal parameters. • Parameter values capture the progress of a recursive process, much like loop control variable values cap-ture the progress of a loop. • Any recursive solution can be replicated through the use of an iterative approach. a. EXCLUSION STATEMENT— Writing recursive program code is outside the scope of the course and AP Exam. • Recursion can be used to traverse String, array, and ArrayList objects.	

Unit 9: Searching & Sorting ***(2 weeks)***
Aligns to the Collegeboard Units 7 & 10

Topics	**Resources, Labs, & Assessments**
● Sequential Search ● Binary Search ● Selection Sort ● Insertion Sort ● MergeSort ● QuickSort ● Objects & ArrayLists	**Resources:** ● Materials can be found on NJCTL website at Unit 9 **Labs:** `WordSearchAndReorder` Program **Assessments:** ● Formative Assessments ● codeIt! Nows ● Graded Programming Quizzes ● Unit Test

Objectives

Curricular Requirements Satisfied:

- Big Idea #3, CONTROL (CON) Students will write iterative, selective, and recursive algorithms to traverse various types of data stored in arrays or ArrayLists **(Skill 3.D)**. The purpose of traversing the data sets will be to search or sort the information into the desired output. Efficiency of each searching and sorting algorithm will be evaluated by students in determining how many iterations or recursive steps are taken by the algorithm **(Skill 2.D)**. Students will modify initial inputs or data sets to determine how the searching or sorting algorithm changes **(Skill 5.C)**.

• Apply sequential/linear search algorithms to search for specific information in array or ArrayList objects.
• There are standard algorithms for searching.
• Sequential/linear search algorithms check each element in order until the desired value is found or all ele-ments in the array or ArrayList have been checked.
• Apply selection sort and insertion sort algorithms to sort the elements of array or ArrayList objects.
• Selection sort and insertion sort are iterative sorting algorithms that can be used to sort elements in an array or ArrayList.
• Compute statement execution counts and informal run-time comparison of sorting algorithms.
• Informal run-time comparisons of program code segments can be made using statement execution counts.
• For algorithms in the context of a particular specification that requires the use of 2D array traversals Identify standard algorithms, Modify standard algorithms, Develop an algorithm.
• When applying sequential/linear search algorithms to 2D arrays, each row must be accessed then sequen-tial/linear search applied to each row of a 2D array.
• All standard 1D array algorithms can be applied to 2D array objects.
• Apply recursive search algorithms to information in String, 1D array, or ArrayList objects.
• Data must be in sorted order to use the binary search algorithm.
• The binary search algorithm starts at the middle of a sorted array or ArrayList and eliminates half of the array or ArrayList in each iteration until the desired value is found or all elements have been eliminated.
• Binary search can be more efficient than sequential/linear search.
a. EXCLUSION STATEMENT— Search algorithms other than sequential/linear and binary search are outside the scope of the course and AP Exam.
• The binary search algorithm can be written either iteratively or recursively
• Apply recursive algorithms to sort elements of array or ArrayList objects.
• Merge sort is a recursive sorting algorithm that can be used to sort elements in an array or ArrayList.

Unit 10: Ethical & Social Implications of Programming ***(1 week)***

Topics	**Resources, Labs, & Assessments**
● Legal Issues with Technology, Information Sharing, and Security ● Social Issues with Technology, Information Sharing, and Security ● Ethical Issues with Technology, Information Sharing, and Security ● Social & Ethical Issues Project Description	**Resources:** ● Materials can be found on NJCTL website at Unit 10 ● College Board released past exams ● Barron's AP Computer Science A ● The Princeton Review: Cracking the AP Computer Science A Exam **Labs:** AP Labs on CollegeBoard site **Assessments:** ● Ethical & Social Issues Project

Objectives
Curricular Requirements Satisfied: ● Big Idea #4, IMPACT OF COMPUTING (IOC) - Students will research and develop a project to describe how data is collected from a computing system or site and present a project on their findings. Through their research students will note the ethical and social implications involved in using computing systems and whether certain systems are more or less reliable than others **(Skill 4.C, Skill 5.A)**. A. Explain the ethical and social implications of computing systems. 1. System reliability is limited. Programmers should make an effort to maximize system reliability. 2. Legal issues and intellectual property concerns arise when creating programs. 3. The creation of programs has impacts on society, economies, and culture. These impacts can be beneficial and/or harmful. B. Explain the risks to privacy from collecting and storing personal data on computer systems. 1. When using the computer, personal privacy is at risk. Programmers should attempt to safeguard personal privacy. 2. Computer use and the creation of programs have an impact on personal security. These impacts can be beneficial and/or harmful.

Student Evaluation

Course grades are based on approximate distributions as shown below. Classwork, homework, and codeIt! Nows, while checked for completion, is not graded.

Category	**Frequency**	**Approx. Point Value**
Long Programs (Labs)	1-2 per week	40-60 points
AP Labs	3, spread throughout course	80-100 points
Graded Programming Quizzes	1-2 per week	10-30 points

Unit Tests	1 roughly every 2-3 weeks (end of each unit)	100 points
Final Exam	1, at the end of the course	100 points

Teacher Resources

CollegeBoard Resources at https://apcentral.collegeboard.org/courses/ap-computer-science-a/course

Litvin, Maria, and Gary Litvin (2015). *Java Methods: An Introduction to Object-oriented Programming, 3rd ed*. Andover, MA: Skylight Publishing. ISBN: 0965485331

Teukolsky M.S., Roselyn (2018). *Barron's AP Computer Science A with Online Tests, 8th ed*. Hauppauge, NY: Barrons Educational Series. ISBN: 1438009194

The Princeton Review (2019). *Cracking the AP Computer Science A Exam, 2019 ed*. New York, NY: The Princeton Review. ISBN: 1524758019