Course Syllabus

AP Computer Science A (Java)

CS A Course Overview (provided by the College Board)

AP Computer Science A is equivalent to a first-semester, college-level CS1 course in computer science. The course introduces students to computer science with fundamental topics that include problem solving, design strategies and methodologies, organization of data (data structures), approaches to processing data (algorithms), analysis of potential solutions, and the ethical and social implications of computing. The course emphasizes both object-oriented and imperative problem solving and design using the Java programming language. These techniques represent proven approaches for developing solutions that can scale up from small, simple problems to large, complex problems.

Required Texts and Other Resources

Online text: CSAwesome Runestone Course (Available for free)

Programming Environment and Equipment

- Programming Environment: Java programming language
- Online site: Students code in the Active Code windows built into the CSAwesome online course.
- Software: Teachers may choose to supplement with another Java IDE software. A list of IDEs is provided. The online IDE repl.it is recommended, and repl link are provided in some coding exercises.
- Computer Equipment: This course will be taught in a computer lab. Students will have access to computers during class.
- Internet access will be made available to students during class to access the course site as well as other relevant sites.

Course Outline

The CSAwesome course follows the 2019 AP CSA unit outline and content as provided in the AP CS A Course and Exam Description. The units that follow interweave the five AP CS A Computational Thinking Practices of Program Design and Algorithm Development, Code Logic, Code Implementation, Code Testing, and Documentation with the four Big Ideas of Modularity, Variables, Control, and Impact of Computing:

	# of 45 min class periods
❖ Unit 1. Getting Started and Primitive Types	~8-10
❖ Unit 2. Using Objects	~13-15
❖ Unit 3. Boolean Expressions and If Statements	~11-13
❖ Unit 4. Iteration (Loops)	~14-16
❖ Unit 5. Writing Classes	~12-14
❖ Unit 6. Arrays	~6-8
❖ Unit 7. ArrayList	~10-12
❖ Unit 8. 2D Arrays	~10-12
❖ Unit 9. Inheritance	~14-16
❖ Unit 10. Recursion	~3-5
❖ Unit 11. Post Test	1
❖ Unit 12. Preparing for the Exam	~5

Lab Component:

The course includes a structured-lab component in which students will complete a minimum of 20 hours of hands-on lab experiences. The curriculum has small coding assignments called Programming Challenges in each lesson, which they are encouraged to do using pair programming. In addition, students

will complete at least three of the following College Board AP Computer Science A labs, as chosen by their teachers, in the CSAwesome course to complete a minimum of 20 hours:

- MagPie Chatbot Lab (built into the curriculum)
- Picture Lab (built into the curriculum)
- Consumer Review Lab (built into the curriculum)
- Celebrity Lab
- Steganography Lab
- Data Lab

Assessments

Quizzes and Exams

Pre and Post Test built into the curriculum
Practice exams built into the curriculum
Personal Progress Checks provided by the College Board as formative assessments

Self-Check and Coding Exercises

Multiple choice questions, Parsons problems, and Active Code Exercises including a Programming Challenge are built into each lesson for each CS A topic.

Teachers can track the progress of their students on these in the Runestone Teacher Dashboard.

AP CS A Exam

Students who complete this course will be prepared to take the AP CS A Exam in May.

Unit Overviews

Unit 1: *Getting Started and Primitive Types*

This unit introduces students to the Java programming language and the use of classes, providing students with a firm foundation of concepts that will be leveraged and built upon in all future units. Students will focus on writing the main method and will start to call pre-existing methods to produce output. The use of pre-existing methods for input is not prescribed in the course; however, input is a necessary part of any computer science course so teachers will need to determine how they will address this in their classrooms. Through interactive coding challenges and exercises built into the lessons, students will start to learn about three built-in data types and learn how to create variables, store values, and interact with those variables using basic operations **(VAR)**. The ability to write expressions is essential to representing the variability of the real world in a program and will be used in all future units. Primitive data is one of two categories of variables covered in this course. Reference data will be covered in Unit 2.

Unit 2: *Using Objects*

In the first unit, students used primitive types to represent real-world data and determined how to use them in arithmetic expressions to solve problems. This unit introduces a new type of data: reference data. Reference data allows real-world objects to be represented in varying degrees specific to a programmer's purpose. Students will learn about modularity in object-oriented programming, which allows us to use abstraction to break complex programs down into individual classes and methods, through interactive coding challenges and exercises built into the lessons **(MOD)**. This unit builds on students' ability to write expressions by introducing them to Math class methods to write expressions for generating random numbers and other more complex operations. In addition, strings and the existing methods within the String class are an important topic within this unit. Knowing how to declare variables or call methods on objects is necessary throughout the course but will be very important in Units 5 and 9 when teaching students how to write their own classes and about inheritance relationships.

Unit 3: *Boolean Expressions and If Statements*

Algorithms are composed of three building blocks: sequencing, selection, and iteration. This unit focuses on selection, which is represented in a program by using conditional statements. Students will learn that conditional control structures give the program the ability to decide and respond appropriately and are a critical aspect of any nontrivial computer program **(CON)**. Through interactive coding challenges and exercises as well as the MagPie Chatbot Lab, students learn that selection and iteration work together to solve problems. In addition to learning the syntax and proper use of conditional statements, students will build on the introduction of Boolean variables by writing Boolean expressions with relational and logical operators. The third building block of all algorithms is iteration, which will be covered in Unit 4.

Unit 4: *Iteration (Loops)*

This unit focuses on iteration using while and for loops. Students will have learned that boolean expressions are useful when a program needs to perform different operations under different conditions. In this unit, they will learn that boolean expressions are also one of the main components in iteration. This unit introduces several standard algorithms that use iteration. Knowledge of standard algorithms makes solving similar problems easier, as algorithms can be modified or combined to suit new situations. Iteration is used when traversing data structures such as arrays, ArrayLists, and 2D arrays. Students will be able to determine the number of times that a code segment will execute by doing a run-time analysis and using a code tracing table to keep track of the variables and their values throughout each iteration of a loop and completing the Consumer Review Lab **(Skill 2.D)**. In addition, students will learn that iteration is a necessary component of several standard algorithms, including searching and sorting, which will be covered in later units.

Unit 5: *Writing Classes*

This unit will pull together information from all previous units to create new, user-defined reference data types in the form of classes. The ability to accurately model real-world entities in a computer program is a large part of what makes computer science so powerful. This unit focuses on identifying appropriate behaviors and attributes of real-world entities and organizing these into classes. Students will build on what they learn in this unit to represent relationships between classes through hierarchies, which appear in Unit 9. The creation of computer programs can have extensive impacts on societies, economies, and cultures. The legal and ethical concerns that come with programs and the responsibilities of programmers are also addressed in this unit. By the end of this unit, students will also understand the importance of documentation when writing program code. Through programming challenges and interactive activities, students will learn about commenting and conditions. Specifically, students will be able to describe pre and post conditions that are necessary for a program to work as intended **(Skill 5.D).**

Unit 6: *Arrays*

This unit focuses on data structures, which are used to represent collections of related data using a single variable rather than multiple variables. Using a data structure along with iterative statements with appropriate bounds will allow for similar treatment to be applied more easily to all values in the collection. Just as there are useful standard algorithms when dealing with primitive data, there are standard algorithms to use with data structures. In this unit, students apply standard algorithms to arrays as well as identify errors in program code found in programming challenges and interactive activities throughout the unit (**Skill 4.B)**. Additional standard algorithms, such as standard searching and sorting algorithms, will be covered in the next unit.

Unit 7: *ArrayList*

As students learned in Unit 6, data structures are helpful when storing multiple related data values. Arrays have a static size, which causes limitations related to the number of elements stored, and it can be challenging to reorder elements stored in arrays. The ArrayList object has a dynamic size, and the class contains methods for insertion and deletion of elements, making reordering and shifting items easier. Deciding which data structure to select becomes increasingly important as the size of the data set grows, such as when using a large real-world data set. In this unit, students will also learn about privacy concerns related to storing large amounts of personal data and about what can happen if such information is compromised **(IOC)**. Through POGIL group work and interactive activities, students will gain an understanding of how to use computing safely and responsibly which requires being aware of privacy, security, and ethical issues.

Unit 8: *2D Arrays*

In Unit 6, students learned how 1D arrays store large amounts of related data. These same concepts will be implemented with two-dimensional (2D) arrays in this unit. A 2D array is most suitable to represent a table. Each table element is accessed using the variable name and row and column indices. Unlike 1D arrays, 2D arrays require nested iterative statements to traverse and access all elements. The easiest way to accomplish this is in row-major order, but it is important to cover additional traversal patterns, such as back and forth or column-major. After completing programming challenges and interactive exercises, such as the Picture Lab, students will be able to write program code to create, traverse, and manipulate elements in 2D array objects **(Skill 3.E)**.

Unit 9: *Inheritance*

Creating objects, calling methods on the objects created, and being able to define a new data type by creating a class are essential understandings before moving into this unit. One of the strongest advantages of Java is the ability to categorize classes into hierarchies through inheritance. Certain existing classes can be extended to include new behaviors and attributes without altering existing code. These newly created

classes are called subclasses. In this unit, students will strengthen their ability to determine an appropriate program design to solve a problem or accomplish a task **(Skill 1.A)**. Students will learn how to recognize common attributes and behaviors that can be used in a superclass and will then create a hierarchy by writing subclasses to extend a superclass. Recognizing and utilizing existing hierarchies will help students create more readable and maintainable programs.

Unit 10: *Recursion*

Sometimes a problem can be solved by solving smaller or simpler versions of the same problem rather than attempting an iterative solution. This is called recursion, and it is a powerful math and computer science idea. In this unit, students will revisit how control is passed when methods are called, which is necessary knowledge when working with recursion. Tracing skills introduced in Unit 2 are helpful for determining the purpose or output of a recursive method. In this unit, students will learn how to write simple recursive methods and determine the purpose or output of a recursive method by tracing.