AP® Computer Science A

Syllabus

July, 2020

Course Overview

This AP Computer Science A class uses the **CompuScholar Java Programming (AP)** curriculum as the primary resource. It is taught as a one-year (two-semester) sequence and covers all required topics in the "Computer Science A" Course Description published by the College Board at https://apcentral.collegeboard.org/pdf/ap-computer-science-a-course-and-exam-description.pdf.

Students need to have typical computer usage skills prior to starting this course; other introductory programming courses are not required. All programming concepts are taught from the ground up in a fun, step-by-step manner. The course includes uses a variety of multi-media content such as full-color text, animated / narrated instructional videos, and guided classroom discussions. Strong emphasis is placed on hands-on programming labs to demonstrate mastery of lesson concepts.

Text and Resources

[1] **Java Programming (AP)**, CompuScholar, Inc. 2019, ISBN 9780988707023

https://www.compuscholar.com/schools/courses/ap-java/

Teaching Strategies

The course material is designed to appeal to a variety of students, from traditional learners who thrive on written text to audio-visual students who enjoy a multi-media format. All content is delivered through an online system that allows students to work seamlessly both in the classroom and at home.

The course integrates the standard 5E instructional model: *engage* with familiar, real-world examples, *explore* with integrated multi-media lessons, *explain* with guided classroom discussions, *elaborate* with hands-on activities to apply concepts, and *evaluate* with automated lesson quizzes and chapter tests. Each lesson contains a combination of text, video, and guided classroom discussions. Students can seamlessly shift between instructional videos and lesson text to suit their learning styles. Fun, hands-on programming labs allow students to quickly see concrete results.

Labs and Programming Environment

Each of the 25+ chapters in the **Java Programming (AP)** course contains one or more hands-on programming labs where students will design or implement programs to demonstrate understanding of the lesson topics. Combined, these labs easily exceed the 20 hour minimum lab requirement. Students will get the opportunity to work on individual and group projects and will experience all phases of a project lifecycle, including requirements, design, implementation, and testing. Students are exposed to the three new College Board labs (Magpie, Picture Lab, Elevens) in addition to the curriculum labs.

The recommended Java programming IDE is "Eclipse" (www.eclipse.org). This free, open-source software is widely used throughout academic and professional environments. The **Java Programming (AP)** course contains detailed installation instructions for use in school labs or at home. Students may use alternate IDEs of desired.

Cross-Reference Tables for AP Exam Topics

The first part of this syllabus contains a Course Planner showing the week-by-week progression through the primary resource and labs.

For a detailed, point-by-point cross reference of every required AP exam topic and Java subset feature to specific lessons in the primary resource, please see the 18-page table titled "**Alignment to the College Board AP Computer Science A Standards**" appended to the end of this document.

That cross-reference table is also available online, for convenience, at the following link:

https://www.compuscholar.com/docs/java/AP_Exam_Cross_Reference.pdf

Course Planner

All readings, unless otherwise noted, are from [1] **Java Programming (AP)**. Each chapter contains multiple lesson quizzes and a chapter test in addition to the listed Lab assignments.

A school year consists of 180 days or 36 weeks. The planner covers all required exam topics prior to the administration of the AP exam in May, including a number of weeks for review and practice tests.

Week	Reading and Objectives	Labs
1	Chapter One: Understanding Computer Programming • A Survey of Computer Hardware • Introduction to Computer Software • Common Programming Languages • Computer Ethics and Security (ethics, copyrights, intellectual property, piracy, software license agreements, firewalls, anti-virus programs, passwords)	**Establish Development Environment** - Install JDK, create working directory, practice submitting projects through the online interface. Class discussion and review of a sample EULA terms and conditions.
2	Chapter Two: Getting Started with Java • The Java Platform • Writing Your First Program • Building and Running from the Command Line • Java Classes and Packages	**Show Time!** – The student's first Java program will print the current time to the console. The student will compile and run the program from the command line (without an IDE).

Week	Reading and Objectives	Labs
3	Chapter Three: The Eclipse IDE • Introducing Eclipse • Eclipse Java IDE Walk-through • Creating an Eclipse Project • Help and Reference Documentation	**Install Eclipse IDE** – If not already installed, the student will add the Eclipse IDE to their home or school computer. **Eclipse Show Time Project** – The student will recreate the same Show Time project using the Eclipse IDE to write, build, and run the program.
4	Chapter Four: Data Types and Variables • Primitive Data Types • Variables • Printing Data	**Experiment with Data Types** – The student will demonstrate declaring, initializing, and printing variables of different data types.
5-6	Chapter Five: Working With Strings • Reference Data Types • Comparing Strings • Common String Operations • Formatting and Building Strings • Converting Between Strings and Numbers	**String Theory** – The student will create multiple strings and perform a variety of operations on them, including comparison, substrings, formatting, parsing, and case conversion.
7	Chapter Six: User Input • Using Command-Line Parameters • Interactive User Input • Validating User Input	**Conversation Piece** – The student will create a program using a command-line Scanner to obtain a variety of user input, and then format that input into an output story.

Week	Reading and Objectives	Labs
8	Chapter Seven: Basic Flow Control • Logical Expressions and Relational Operators • Using the "if" Statement • The "switch" Statement • For Loops • While Loops	**Fun Factorials** – The student will demonstrate use of a for() loop, while() loop, and do-while() loop to calculate factorials of an input number. Boundary conditions involving maximum integer sizes are explored and tested.
9	Chapter Eight: Writing Methods • Writing and Calling Methods • Method Parameters and Return Values • Calling Methods	**Checkerboard** – The student will write a program that includes a new function to print a checkerboard pattern to the screen given input row and column size parameters.
10	Chapter Nine: Debugging and Exceptions • Logic Errors, Runtime Errors and Exceptions • Catching Exceptions • Finding Runtime Errors • The Eclipse Debugger	**Bug Hunt** – The student is presented with a program that contains a number of bugs. The student will use the Eclipse debugger and troubleshooting skills to identify and resolve each issue.
11	Chapter Ten: Introduction to OOP • Object-Oriented Concepts • Defining a Class • Public, Private, Protected Classes	**Dog House** – The student will create a simple set of interacting classes.

<table>
<tr><th>Week</th><th>Reading and Objectives</th><th>Labs</th></tr>
<tr><td>12</td><td>Chapter Eleven: Objects in Java
Constructors
Object Interfaces
Static Members</td><td>Let's Go Racing! – The student will create a RaceCar object and an IRacer object. Multiple RaceCar instances will be added to a provided RaceTrack object that knows how to run races through the IRacer interface.</td></tr>
<tr><td>13</td><td>Chapter Twelve: Graphical Java Programs
Java Swing
Creating a Simple Window
Event-Driven Programming
Layout Managers</td><td>Phone Dialer – The student's first Java Swing program will show a simple phone keypad and allow users to enter a phone number for display.</td></tr>
<tr><td>14</td><td>Chapter Thirteen: Swing Input Controls
Text and Numeric Input
List Input
Option Input</td><td>Pizza Place – The student will create a pizza ordering screen to demonstrate proper use of many common UI widgets (check boxes, radio buttons, list boxes, etc).</td></tr>
<tr><td>15-16</td><td>Chapter Fourteen: Arrays and Collections
1D Arrays
2D Arrays
Java Lists
ArrayLists
Iterators and the Enhanced for() Loop</td><td>Gold Rush – The student will write algorithms to populate and traverse a data structure that combines 2D arrays and ArrayLists.</td></tr>
</table>

Week	Reading and Objectives	Labs
17-18	Chapter Fifteen: Inheritance and Polymorphism • Learn about the "Jail Break!" game. • Base Classes and Derived Classes • Using References to Base and Derived Classes • Overriding Base Methods • The "Object" Base Class • Using Base Features from Derived Classes	**Game Pieces** – The student will create three derived classes (Deputy, Henchman, Kingpin) from an abstract base, in preparation for using these classes in the mid-term project. The classes are tested to ensure they meet the requirements using a provided test class.
19-20	Chapter Sixteen: Jail Break Project For the mid-term project the student will complete a game called "Jail Break" that is based on an old Viking board game. The student will create the abstract hierarchy of pieces (AbstractGamePiece, Deputy, Henchman, Kingpin) and write other logic to complete the game. The project consists of 6 guided lab steps that involve creating new classes, modifying existing code, and integrating with provided starter objects. Each guided step contains a checkpoint for testing to ensure code meets the requirements at each step. Key concepts demonstrated include: • Encapsulation • Inheritance • Polymorphism • Modeling real-world activities • Integrating new and existing classes	**Building the Activity Starter** – Ensure the student can find and build the starter project. **Completing JailBreak.reset()** – Write logic to initialize the game board with pieces in the starting position. **Selecting Game Pieces** – Write game logic to allow selection and de-selection of game pieces. **Moving Game Pieces** – Write game logic (including virtual method overrides) to control game piece movement. **Capturing Game Pieces** – Write game logic to control game piece capturing. **Ending the Game** – Complete the end-of-game logic.

Week	Reading and Objectives	Labs
21	Chapter Seventeen: Math Functions in Java • Java Math Functions • The Binary Number System • Creating a MathFactory demonstration • Common Algorithms	**MathFactory Activity** – The student will expand the MathFactory lab to include decimal-to-binary conversion. **Algorithm Practice** – The student will create two simple algorithms based on a problem description.
22	Chapter Eighteen: File Access • Data Streams • Reading and Writing Text Data • Reading and Writing Binary Data	**Address CSV** – The student will write a program to convert a list of Address structures to a CSV file on disk, and then read that file back in again and re-populate the address list.
23	Chapter Nineteen: Sorting, Searching and Recursion • Recursion • Sorting Algorithms (Bubble, Selection, Insertion, Merge) • Searching Algorithms (Sequential, Binary)	**Recursive Binary Search** – The student will write a binary search function to locate a number in a pre-sorted array.
24	Chapter Twenty: Program Efficiency • More Common Algorithms • Algorithm Performance (Big-O) • Measuring Sorting Efficiency	**Comparison of Sorting Algorithms** – The student will implement timing and data-generation algorithms and measure the performance of 4 different sort routines with various numbers of elements.
25	Chapter Twenty-One: Vector and Bitmap Images • Screen Coordinates • Drawing Shapes • Drawing Images	**Sky Art** – The student will use recursion, vector graphics, and image graphics to generate a randomized cloudy sky scene.

<table>
<tr><th>Week</th><th>Reading and Objectives</th><th>Labs</th></tr>
<tr><td>26</td><td>Chapter Twenty-Two: Object Composition and Copying
Functional Decomposition
Composite Classes
Copying objects</td><td>Designing a Composite Class - In this lab the student will design a composite class based around a Computer object. The output is a diagram instead of a program.</td></tr>
<tr><td>27</td><td>Chapter Twenty-Three: Computer Networking
Basic Networking
Network Topology
Network Addressing</td><td>Animal Palace – Students will use online tools to find images and store in a shared directory and class web page.</td></tr>
<tr><td>28</td><td>Chapter Twenty-Four: Software Engineering Principles
Design Processes and Teamwork
Java Doc
Testing Your Code</td><td>Creating JavaDoc HTML – The student will add JavaDoc comments to an earlier lab project and generate HTML output using the javadoc tool.</td></tr>
<tr><td>29-30</td><td>Chapter Twenty-Six: College Board Supplemental Labs

This chapter gives students a chance to complete many activities in the College Board AP supplemental labs, including:
Magpie
Picture Lab
Elevens</td><td>Magpie Chatbot – Guided lab involving string parsing and manipulation.

Picture Lab - Guided lab using 2D arrays in the context of image processing.

Elevens - Guided lab through object-oriented design concepts with a simple card game.</td></tr>
</table>

Week	Reading and Objectives	Labs
	Chapter Twenty-Seven: GridWorld Case Study The older GridWorld case study is available for students who wish to pursue extra exercises on their own, but it is not required for the AP exam. • Introducing GridWorld • Bug Variations • GridWorld Classes and Interfaces • Interacting Objects	**Bug Variations** **GridWorld Classes and Interfaces** **Interacting Objects**
31-32	AP EXAM – PRACTICE TESTS, REVIEW, MAKE-UP WORK	Flexible time used to review and practice for the AP exam.
33	AP EXAM – EARLY MAY	
34-36	Chapter Twenty-Five: Team Project The final project can be completed after the AP exam and the timeline scaled to fit available time. Student-driven labs will cover each phase of the software lifecycle. • Project Requirements • Project Design • Project Implementation • Project Testing	**Team Project Requirements** – Student teams will define their final project requirements. **Project Design** – Student teams will design their final projects. **Team Project Implementation** – Student teams will code their final project. **Team Project Testing** – Student teams will test their final project.

CompuScholar, Inc.

Alignment to the College Board AP **Computer Science A** Standards

9th - 12th grades

AP Course Details:

Course Title:	AP Computer Science A
Grade Level:	9th - 12th grades
Standards Version:	Fall 2019
Standards Link:	ap-computer-science-a-course-and-exam-description.pdf

CompuScholar Course Details:

Course Title:	Java Programming (AP)
Course ISBN:	978-0-9887070-2-3
Course Year:	2019

Note 1: Citation(s) listed may represent a subset of the instances where objectives are met throughout the course.

Note 2: Citation(s) for a "Lesson" refer to the "Lesson Text" elements and associated "Activities" within the course, unless otherwise noted. The "Instructional Video" components are supplements designed to introduce or re-enforce the main lesson concepts, and the Lesson Text contains full details.

AP Course Description

This course teaches students the fundamentals of the Java programming language and covers all required topics defined by the College Board's AP Computer Science A course description.

AP Lab Requirements

The AP Computer Science A course must include a minimum of 20 hours of hands-on structured-lab experiences to engage students in individual or group problem solving.	**CITATION(S)**
This course easily meets and exceeds the 20-hour minimum lab requirement with hands-on lesson exercises and chapter activities. In addition, coverage and time for the new example labs is provided for teachers to use as needed.	See "Work with Me" sections within lessons and "Chapter Activities" in each chapter.
Magpie Lab (recommended starting in 2014-2015)	Chapter 26, Lesson 1
Picture Lab (recommended starting in 2014-2015)	Chapter 26, Lesson 2
Elevens Lab (recommended starting in 2014-2015)	Chapter 26, Lesson 3
GridWorld Case Study (no longer required, but available for use if desired)	Chapter 27

AP Topic Outline

UNIT 1: Primitive Types	CITATION(S)
Topic 1.1: Why Programming? Why Java?	
MOD-1.A.1 - System.out.print and System.out.println display information on the computer monitor.	Chapter 2, Lesson 2 Chapter 4, Lesson 3
MOD-1.A.2 - System.out.println moves the cursor to a new line after the information has been displayed, while System.out.print does not.	Chapter 4, Lesson 3
TOPIC 1.2: Variables and Data Types	
VAR-1.B.1 - A type is a set of values (a domain) and a set of operations on them.	Chapter 4, Lesson 1
VAR-1.B.2 - Data types can be categorized as either primitive or reference.	Chapter 4, Lesson 1 Chapter 5, Lesson 2
VAR-1.B.3 - The primitive data types used in this course define the set of operations for numbers and Boolean values.	Chapter 4, Lesson 1 Chapter 4, Lesson 2
VAR-1.C.1 - The three primitive data types used in this course are int, double, and boolean.	Chapter 4, Lesson 1 Chapter 4, Lesson 2
VAR-1.C.2 - Each variable has associated memory that is used to hold its value.	Chapter 4, Lesson 1 Chapter 4, Lesson 2
VAR-1.C.3 - The memory associated with a variable of a primitive type holds an actual primitive value.	Chapter 4, Lesson 1 Chapter 4, Lesson 2
VAR-1.C.4 - When a variable is declared final, its value cannot be changed once it is initialized.	Chapter 4, Lesson 2
TOPIC 1.3: Expressions and Assignment Statements	
CON-1.A.1 - A literal is the source code representation of a fixed value	Chapter 4, Lesson 2
CON-1.A.2 - Arithmetic expressions include expressions of type int and double.	Chapter 4, Lesson 2
CON-1.A.3 - The arithmetic operators consist of +, −, *, /, and %	Chapter 4, Lesson 2
CON-1.A.4 - An arithmetic operation that uses two int values will evaluate to an int value.	Chapter 4, Lesson 2
CON-1.A.5 - An arithmetic operation that uses a double value will evaluate to a double value.	Chapter 4, Lesson 2
CON-1.A.6 - Operators can be used to construct compound expressions.	Chapter 4, Lesson 2 Chapter 7, Lesson 1
CON-1.A.7 - During evaluation, operands are associated with operators according to operator precedence to determine how they are grouped.	Chapter 7, Lesson 1
CON-1.A.8 - An attempt to divide an integer by zero will result in an ArithmeticException to occur.	Chapter 9, Lesson 1
CON-1.B.1 - 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.	Chapter 4, Lesson 2

CON-1.B.2 - During execution, expressions are evaluated to produce a single value.	Chapter 4, Lesson 2 Chapter 7, Lesson 1
CON-1.B.3 - The value of an expression has a type based on the evaluation of the expression.	Chapter 4, Lesson 2 Chapter 7, Lesson 1
TOPIC 1.4: Compound Assignment Operators	
CON-1.B.4 - Compound assignment operators (+=, −=, *=, /=, %=) can be used in place of the assignment operator.	Chapter 4, Lesson 2
CON-1.B.5 - 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.	Chapter 4, Lesson 2
TOPIC 1.5: Casting and Ranges of Variables	
CON-1.C.1 - The casting operators (int) and (double) can be used to create a temporary value converted to a different data type.	Chapter 4, Lesson 2
CON-1.C.2 - Casting a double value to an int causes the digits to the right of the decimal point to be truncated.	Chapter 4, Lesson 2
CON-1.C.3 - Some programming code causes int values to be automatically cast (widened) to double values.	Chapter 4, Lesson 2
CON-1.C.4 - Values of type double can be rounded to the nearest integer by (int)(x + 0.5) or (int)(x – 0.5) for negative numbers.	Chapter 4, Lesson 2
CON-1.C.5 - Integer values in Java are represented by values of type int, which are stored using a finite amount (4 bytes) of memory. Therefore, an int value must be in the range from Integer.MIN_VALUE to Integer.MAX_VALUE inclusive.	Chapter 4, Lesson 2
CON-1.C.6 - 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.	Chapter 17, Lesson 2

UNIT 2: Using Objects	**CITATION(S)**
TOPIC 2.1: Objects: Instances of Classes	
MOD-1.B.1 - An object is a specific instance of a class with defined attributes.	Chapter 10, Lessons 1-2
MOD-1.B.2 A class is the formal implementation, or blueprint, of the attributes and behaviors of an object.	Chapter 10, Lessons 1-2
TOPIC 2.2: Creating and Storing Objects (Instantiation)	
MOD-1.C.1 - A signature consists of the constructor name and the parameter list.	Chapter 11, Lesson 1
MOD-1.C.2 - 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.	Chapter 11, Lesson 1
MOD-1.C.3 - A parameter is a value that is passed into a constructor. These are often referred to as actual parameters.	Chapter 11, Lesson 1
MOD-1.C.4 - Constructors are said to be overloaded when there are multiple constructors with the same name but a different signature.	Chapter 11, Lesson 1

MOD-1.C.5 - The actual parameters passed to a constructor must be compatible with the types identified in the formal parameter list.	Chapter 11, Lesson 1
MOD-1.C.6 - Parameters are passed using call by value. Call by value initializes the formal parameters with copies of the actual parameters.	Chapter 8, Lesson 3
MOD-1.D.1 - Every object is created using the keyword new followed by a call to one of the class's constructors.	Chapter 10, Lesson 2 Chapter 11, Lesson 1
MOD-1.D.2 - A class contains constructors that are invoked to create objects. They have the same name as the class.	Chapter 11, Lesson 1
MOD-1.D.3 - Existing classes and class libraries can be utilized as appropriate to create objects.	Chapter 5 Chapter 17, Lesson 1
MOD-1.D.4 - Parameters allow values to be passed to the constructor to establish the initial state of the object.	Chapter 11, Lesson 1
VAR-1.D.1 - The keyword null is a special value used to indicate that a reference is not associated with any object.	Chapter 5, Lesson 1
VAR-1.D.2 - 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.	Chapter 5, Lesson 1
TOPIC 2.3: Calling a Void Method	
MOD-1.E.1 - An object's behavior refers to what the object can do (or what can be done to it) and is defined by methods.	Chapter 10, Lessons 1-2
MOD-1.E.2 - 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.	Chapter 10, Lessons 1-2
MOD-1.E.3 - A method signature for a method without parameters consists of the method name and an empty parameter list.	Chapter 8, Lesson 1
MOD-1.E.4 - 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	Chapter 8, Lesson 1
MOD-1.E.5 - Non-static methods are called through objects of the class.	Chapter 10, Lesson 2
MOD-1.E.6 - The dot operator is used along with the object name to call non-static methods.	Chapter 10, Lesson 2
MOD-1.E.7 - Void methods do not have return values and are therefore not called as part of an expression.	Chapter 8, Lesson 1
MOD-1.E.8 - Using a null reference to call a method or access an instance variable causes a NullPointerException to be thrown.	Chapter 9, Lesson 1
TOPIC 2.4: Calling a Void Method with Parameters	
MOD-1.F.1 - A method signature for a method with parameters consists of the method name and the ordered list of parameter types.	Chapter 8, Lessons 2-3
MOD-1.F.2 - Values provided in the parameter list need to correspond to the order and type in the method signature.	Chapter 8, Lessons 2-3

MOD-1.F.3 - Methods are said to be overloaded when there are multiple methods with the same name but a different signature.	Chapter 8, Lessons 2-3
TOPIC 2.5: Calling a Non-void Method	
MOD-1.G.1 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.	Chapter 8, Lessons 2-3
TOPIC 2.6: String Objects: Concatenation, Literals, and More	
VAR-1.E.1 - String objects can be created by using string literals or by calling the String class constructor.	Chapter 5, Lesson 1
VAR-1.E.2 - String objects are immutable, meaning that String methods do not change the String object.	Chapter 5, Lessons 1-2
VAR-1.E.3 - String objects can be concatenated using the + or += operator, resulting in a new String object.	Chapter 5, Lesson 4
VAR-1.E.4 - Primitive values can be concatenated with a String object. This causes implicit conversion of the values to String objects.	Chapter 5, Lessons 4-5
VAR-1.E.5 - Escape sequences start with a \ and have a special meaning in Java. Escape sequences used in this course include \”, \\, and \n.	Chapter 4, Lesson 3
TOPIC 2.7: String Methods	
VAR-1.E.6 - Application program interfaces (APIs) and libraries simplify complex programming tasks	Chapter 2, Lesson 4 Chapter 17, Lesson 1
VAR-1.E.7 - Documentation for APIs and libraries are essential to understanding the attributes and behaviors of an object of a class.	Chapter 24, Lesson 2
VAR-1.E.8 - Classes in the APIs and libraries are grouped into packages.	Chapter 2, Lesson 4
VAR-1.E.9 - The String class is part of the java.lang package. Classes in the java.lang package are available by default.	Chapter 5, Lesson 1
VAR-1.E.10 - A String object has index values from 0 to length– 1. Attempting to access indices outside this range will result in an IndexOutOfBoundsException.	Chapter 5, Lesson 3
VAR-1.E.11 - A String object can be concatenated with an object reference, which implicitly calls the referenced object’s toString method.	Chapter 5, Lesson 4
VAR-1.E.12 - The following String methods and constructors—including what they do and when they are used—are part of the Java Quick Reference:	See Below
String(String str) — Constructs a new String object that represents the same sequence of characters as str	Chapter 5, Lesson 1
int length() — Returns the number of characters in a String object	Chapter 5, Lesson 3
String substring(int from, int to) — Returns the substring beginning at index from and ending at index to - 1	Chapter 5, Lesson 3
String substring(int from)— Returns substring(from, length())	Chapter 5, Lesson 3
int indexOf(String str) — Returns the index of the first occurrence of str; returns -1 if not found	Chapter 5, Lesson 3
boolean equals(String other)— Returns true if this is equal to other; returns false otherwise	Chapter 5, Lesson 3

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	Chapter 5, Lesson 3
VAR-1.E.13 - A string identical to the single element substring at position index can be created by calling substring(index, index + 1).	Chapter 5, Lesson 3
TOPIC 2.8: Wrapper Classes: Integer and Double	
VAR-1.F.1 - The Integer class and Double class are part of the java.lang package.	Chapter 4, Lesson 2 Chapter 5, Lesson 3
VAR-1.F.2 - The following Integer methods and constructors — including what they do and when they are used—are part of the Java Quick Reference:	Chapter 4, Lesson 2
Integer(int value) — Constructs a new Integer object that represents the specified int value	Chapter 4, Lesson 2
Integer.MIN_VALUE — The minimum value represented by an int or Integer	Chapter 4, Lesson 2
Integer.MAX_VALUE — The maximum value represented by an int or Integer	Chapter 4, Lesson 2
int intValue() — Returns the value of this Integer as an int	Chapter 4, Lesson 2
VAR-1.F.3 - The following Double methods and constructors — including what they do and when they are used—are part of the Java Quick Reference:	Chapter 4, Lesson 2
Double(double value) —Constructs a new Double object that represents the specified double value	Chapter 4, Lesson 2
double doubleValue() — Returns the value of this Double as a double	Chapter 4, Lesson 2
VAR-1.F.4 - 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.	Chapter 4, Lesson 2
VAR-1.F.5 - The Java compiler applies autoboxing when a primitive value is: * Passed as a parameter to a method that expects an object of the corresponding wrapper class. * Assigned to a variable of the corresponding wrapper class.	Chapter 4, Lesson 2
VAR-1.F.6 - 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.	Chapter 4, Lesson 2
VAR-1.F.7 - The Java compiler applies unboxing when a wrapper class object is: * Passed as a parameter to a method that expects a value of the corresponding primitive type. * Assigned to a variable of the corresponding primitive type.	Chapter 4, Lesson 2
TOPIC 2.9: Using the Math Class	
MOD-1.H.1 - Static methods are called using the dot operator along with the class name unless they are defined in the enclosing class.	Chapter 11, Lesson 3 Chapter 17, Lesson 1
CON-1.D.1 - The Math class is part of the java.langpackage.	Chapter 17, Lesson 1

CON-1.D.2 - The Math class contains only static methods.	Chapter 17, Lesson 1
CON-1.D.3 - The following static Math methods—including what they do and when they are used—are part of the Java Quick Reference:	Chapter 17, Lesson 1
int abs(int x) — Returns the absolute value of an int value	Chapter 17, Lesson 1
double abs(double x) — Returns the absolute value of a double value	Chapter 17, Lesson 1
double pow(double base, double exponent) — Returns the value of the first parameter raised to the power of the second parameter	Chapter 17, Lesson 1
double sqrt(double x) — Returns the positive square root of a double value	Chapter 17, Lesson 1
double random() — Returns a double value greater than or equal to 0.0 and less than 1.0	Chapter 17, Lesson 1
CON-1.D.4 - The values returned from Math.random can be manipulated to produce a random int or double in a defined range.	Chapter 17, Lesson 1

UNIT 3: Boolean Expressions and if Statements	**CITATION(S)**
TOPIC 3.1: Boolean Expressions	
CON-1.E.1 - Primitive values and reference values can be compared using relational operators (i.e., == and !=).	Chapter 7, Lesson 1
CON-1.E.2 - Arithmetic expression values can be compared using relational operators (i.e., <, >, <=, >=).	Chapter 7, Lesson 1
CON-1.E.3 - An expression involving relational operators evaluates to a Boolean value.	Chapter 7, Lesson 1
TOPIC 3.2: if Statements and Control Flow	
CON-2.A.1 - Conditional statements interrupt the sequential execution of statements.	Chapter 7, Lesson 2
CON-2.A.2 - if statements affect the flow of control by executing different statements based on the value of a Boolean expression.	Chapter 7, Lesson 2
CON-2.A.3 - 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.	Chapter 7, Lesson 2
TOPIC 3.3: if-else Statements	
CON-2.A.4 - 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.	Chapter 7, Lesson 2
TOPIC 3.4: elseif Statements	
CON-2.A.5 - 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.	Chapter 7, Lesson 2

TOPIC 3.5: Compound Boolean Expressions	
CON-2.B.1 - Nested if statements consist of if statements within if statements.	Chapter 7, Lesson 2
CON-1.F.1 - Logical operators !(not), &&(and), and ||(or) are used with Boolean values. This represents the order these operators will be evaluated.	Chapter 7, Lesson 1
CON-1.F.2 - An expression involving logical operators evaluates to a Boolean value.	Chapter 7, Lesson 1
CON-1.F.3 - 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.	Chapter 7, Lesson 1
TOPIC 3.6: Equivalent Boolean Expressions	
CON-1.G.1 - De Morgan's Laws can be applied to Boolean expressions.	Chapter 7, Lesson 1
CON-1.G.2 - Truth tables can be used to prove Boolean identities.	Chapter 7, Lesson 1
CON-1.G.3 - Equivalent Boolean expressions will evaluate to the same value in all cases.	Chapter 7, Lesson 1
TOPIC 3.7: Comparing Objects	
CON-1.H.1 - Two object references are considered aliases when they both reference the same object.	Chapter 5, Lesson 2
CON-1.H.2 - Object reference values can be compared, using == and !=, to identify aliases	Chapter 15, Lesson 5
CON-1.H.3 - A reference value can be compared with null, using == or !=, to determine if the reference actually references an object.	Chapter 7, Lesson 1
CON-1.H.4 - Often classes have their own equals method, which can be used to determine whether two objects of the class are equivalent.	Chapter 15, Lesson 5

UNIT 4: Iteration	**CITATION(S)**
TOPIC 4.1: while Loops	
CON-2.C.1 - Iteration statements change the flow of control by repeating a set of statements zero or more times until a condition is met.	Chapter 7, Lesson 5
CON-2.C.2 - 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.	Chapter 7, Lesson 5
CON-2.C.3 - A loop is an infinite loop when the Boolean expression always evaluates to true.	Chapter 7, Lesson 5
CON-2.C.4 - If the Boolean expression evaluates to false initially, the loop body is not executed at all.	Chapter 7, Lesson 5
CON-2.C.5 - Executing a return statement inside an iteration statement will halt the loop and exit the method or constructor.	Chapter 7, Lesson 5 Chapter 8, Lesson 2

CON-2.D.1- 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	Chapter 17, Lesson 4 Chapter 20, Lesson 1
CON-2.D.2 - There are standard algorithms to: * Determine a minimum or maximum value * Compute a sum, average, or mode	Chapter 17, Lesson 4 Chapter 20, Lesson 1
TOPIC 4.2: for Loops	
CON-2.E.1 - 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.	Chapter 7, Lesson 4
CON-2.E.2 - 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.	Chapter 7, Lesson 4
CON-2.E.3 - 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.	Chapter 7, Lesson 4
CON-2.E.4 - A for loop can be rewritten into an equivalent while loop and vice versa.	Chapter 7, Lessons 4-5
CON-2.E.5 - “Off by one” errors occur when the iteration statement loops one time too many or one time too few.	Chapter 7, Lesson 4
TOPIC 4.3: Developing Algorithms Using Strings	
CON-2.F.1 - 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	Chapter 17, Lesson 4 Chapter 20, Lesson 1
TOPIC 4.4: Nested Iteration	
CON-2.G.1 - Nested iteration statements are iteration statements that appear in the body of another iteration statement.	Chapter 7, Lessons 4-5 Chapter 8 Activity Chapter 14, Lesson 2
CON-2.G.2 - When a loop is nested inside another loop, the inner loop must complete all its iterations before the outer loop can continue.	Chapter 7, Lessons 4-5 Chapter 8 Activity Chapter 14, Lesson 2
TOPIC 4.5: Informal Code Analysis	
CON-2.H.1 - A statement execution count indicates the number of times a statement is executed by the program.	Chapter 20, Lessons 2-3

UNIT 5: Writing Classes	**CITATION(S)**
TOPIC 5.1: Anatomy of a Class	
MOD-2.A.1 - The keywords public and private affect the access of classes, data, constructors, and methods.	Chapter 10, Lesson 3
MOD-2.A.2 - The keyword private restricts access to the declaring class, while the keyword public allows access from classes outside the declaring	Chapter 10, Lesson 3

MOD-2.A.3 - Classes are designated public.	Chapter 10, Lesson 3
MOD-2.A.4 - Access to attributes should be kept internal to the class. Therefore, instance variables are designated as private.	Chapter 10, Lesson 3
MOD-2.A.5 - Constructors are designated public.	Chapter 11, Lesson 1
MOD-2.A.6 - Access to behaviors can be internal or external to the class. Therefore, methods can be designated as either public or private.	Chapter 10, Lesson 3
MOD-3.A.1 - Data encapsulation is a technique in which the implementation details of a class are kept hidden from the user.	Chapter 10, Lesson 3
MOD-3.A.2 - 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.	Chapter 10, Lesson 3
MOD-3.A.3 - Instance variables are encapsulated by using the private access modifier.	Chapter 10, Lesson 3
MOD-3.A.4 - The provided accessor and mutator methods in a class allow client code to use and modify data.	Chapter 10, Lesson 3
TOPIC 5.2: Constructors	
MOD-2.B.1 - 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.	Chapter 10, Lesson 2
MOD-2.B.2 - Constructors are used to set the initial state of an object, which should include initial values for all instance variables.	Chapter 11, Lesson 1
MOD-2.B.3 - Constructor parameters are local variables to the constructor and provide data to initialize instance variables.	Chapter 11, Lesson 1
MOD-2.B.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.	Chapter 11, Lesson 1
MOD-2.B.5 - When no constructor is written, Java provides a no-argument constructor, and the instance variables are set to default values.	Chapter 11, Lesson 1
TOPIC 5.3: Documentation with Comments	
MOD-2.C.1- Comments are ignored by the compiler and are not executed when the program is run.	Chapter 2, Lesson 2
MOD-2.C.2 - 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.	Chapter 2, Lesson 2 Chapter 24, Lesson 2
MOD-2.C.3 - 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.	Chapter 24, Lesson 3
MOD-2.C.4 - 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.	Chapter 24, Lesson 3

MOD-2.C.5 - Programmers write method code to satisfy the postconditions when preconditions are met	Chapter 24, Lesson 3
TOPIC 5.4: Accessor Methods	
MOD-2.D.1 - An accessor method allows other objects to obtain the value of instance variables or static variables.	Chapter 10, Lesson 3
MOD-2.D.2 - A non-void method returns a single value. Its header includes the return type in place of the keyword void.	Chapter 8, Lesson 2
MOD-2.D.3 - 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.”	Chapter 8, Lesson 2
MOD-2.D.4 - When the return expression is a reference to an object, a copy of that reference is returned, not a copy of the object.	Chapter 8, Lesson 2
MOD-2.D.5 - The return keyword is used to return the flow of control to the point immediately following where the method or constructor was called.	Chapter 8, Lesson 2
MOD-2.D.6 - 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.	Chapter 15, Lesson 5
MOD-2.D.7 - 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.	Chapter 15, Lesson 5
TOPIC 5.5: Mutator Methods	
MOD-2.E.1 - A void method does not return a value. Its header contains the keyword void before the method name.	Chapter 8, Lesson 1 Chapter 10, Lesson 3
MOD-2.E.2 - A mutator (modifier) method is often a void method that changes the values of instance variables or static variables.	Chapter 10, Lesson 3
TOPIC 5.6: Writing Methods	
MOD-2.F.1 - 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.	Chapter 10, Lesson 3
MOD-2.F.2 - Non-void methods with parameters receive values through parameters, use those values, and return a computed value of the specified type.	Chapter 8, Lessons 2-3
MOD-2.F.3 - It is good programming practice to not modify mutable objects that are passed as parameters unless required in the specification.	Chapter 8, Lesson 3
MOD-2.F.4- 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.	Chapter 8, Lesson 3
MOD-2.F.5 - 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.	Chapter 8, Lesson 3
MOD-2.F.6 - Passing a reference parameter results in the formal parameter and the actual parameter being aliases. They both refer to the same object.	Chapter 8, Lesson 3
TOPIC 5.7: Static Variables and Methods	
MOD-2.G.1 - Static methods are associated with the class, not objects of the class.	Chapter 11, Lesson 3

MOD-2.G.2 - Static methods include the keyword static in the header before the method name	Chapter 11, Lesson 3
MOD-2.G.3 - Static methods cannot access or change the values of instance variables.	Chapter 11, Lesson 3
MOD-2.G.4 - Static methods can access or change the values of static variables.	Chapter 11, Lesson 3
MOD-2.G.5 - Static methods do not have a this reference and are unable to use the class’s instance variables or call non-static methods.	Chapter 11, Lesson 3
MOD-2.H.1 - Static variables belong to the class, with all objects of a class sharing a single static variable.	Chapter 11, Lesson 3
MOD-2.H.2 - Static variables can be designated as either public or private and are designated with the static keyword before the variable type.	Chapter 11, Lesson 3
MOD-2.H.3 - Static variables are used with the class name and the dot operator, since they are associated with a class, not objects of a class.	Chapter 11, Lesson 3
TOPIC 5.8: Scope and Access	
VAR-1.G.1 - 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.	Chapter 10, Lesson 2
VAR-1.G.2 - 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.	Chapter 10, Lesson 2
VAR-1.G.3 - Formal parameters and variables declared in a method or constructor can only be used within that method or constructor.	Chapter 10, Lesson 2
VAR-1.G.4 - Through method decomposition, a programmer breaks down a large problem into smaller subproblems by creating methods to solve each individual subproblem.	Chapter 22, Lesson 1
TOPIC 5.9: this Keyword	
VAR-1.H.1 - 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.	Chapter 10, Lesson 2
VAR-1.H.2 - The keyword this can be used to pass the current object as an actual parameter in a method call.	Chapter 10, Lesson 2
TOPIC 5.10: Ethical and Social Implications of Computing Systems	
IOC-1.A.1 - System reliability is limited. Programmers should make an effort to maximize system reliability.	Chapter 1, Lesson 4 Chapter 9, Lesson 3 Chapter 24, Lesson 3
IOC-1.A.2 - Legal issues and intellectual property concerns arise when creating programs.	Chapter 1, Lesson 4-5
IOC-1.A.3 - The creation of programs has impacts on society, economies, and culture. These impacts can be beneficial and/or harmful.	Chapter 1, Lesson 4-5

UNIT 6: Array	**CITATION(S)**
TOPIC 6.1: Array Creation and Access	
VAR-2.A.1 - The use of array objects allows multiple related items to be represented using a single variable.	Chapter 14, Lesson 1

VAR-2.A.2 - The size of an array is established at the time of creation and cannot be changed.	Chapter 14, Lesson 1
VAR-2.A.3 - Arrays can store either primitive data or object reference data.	Chapter 14, Lesson 1
VAR-2.A.4 - When an array is created using the keyword new, all of its elements are initialized with a specific value based on the type of elements: * Elements of type int are initialized to 0 * Elements of type double are initialized to 0.0 * Elements of type boolean are initialized to false * Elements of a reference type are initialized to the reference value null. No objects are automatically created	Chapter 14, Lesson 1
VAR-2.A.5 - Initializer lists can be used to create and initialize arrays.	Chapter 14, Lesson 1
VAR-2.A.6 - Square brackets ([]) are used to access and modify an element in a 1D array using an index.	Chapter 14, Lesson 1
VAR-2.A.7 - The valid index values for an array are 0 through one less than the number of elements in the array, inclusive. Using an index value outside of this range will result in an ArrayIndexOutOfBoundsExceptionbeing thrown.	Chapter 14, Lesson 1
TOPIC 6.2: Traversing Arrays	
VAR-2.B.1 - Iteration statements can be used to access all the elements in an array. This is called traversing the array.	Chapter 14, Lesson 1 Chapter 14, Lesson 5
VAR-2.B.2 - Traversing an array with an indexed for loop or while loop requires elements to be accessed using their indices.	Chapter 14, Lesson 1 Chapter 14, Lesson 5
VAR-2.B.3 - Since the indices for an array start at 0 and end at the number of elements − 1, "off by one" errors are easy to make when traversing an array, resulting in an ArrayIndexOutOfBoundsExceptionbeing thrown.	Chapter 14, Lesson 1 Chapter 14, Lesson 5
TOPIC 6.3: Enhanced forLoop for Arrays	
VAR-2.C.1 - An enhanced for loop header includes a variable, referred to as the enhanced for loop variable.	Chapter 14, Lesson 5
VAR-2.C.2 - For each iteration of the enhanced for loop, the enhanced for loop variable is assigned a copy of an element without using its index.	Chapter 14, Lesson 5
VAR-2.C.3 - Assigning a new value to the enhanced for loop variable does not change the value stored in the array.	Chapter 14, Lesson 5
VAR-2.C.4 - Program code written using an enhanced for loop to traverse and access elements in an array can be rewritten using an indexed for loop or a while loop.	Chapter 14, Lesson 5
TOPIC 6.4: Developing Algorithms Using Arrays	
CON-2.I.1 - There are standard algorithms that utilize array traversals to: * Determine a minimum or maximum value * Compute a sum, average, or mode * Determine if at least one element has a particular property * Determine if all elements have a particular property * Access all consecutive pairs of elements * Determine the presence or absence of duplicate elements * Determine the number of elements meeting specific criteria	Chapter 17, Lesson 4 Chapter 19, Lessons 2-3 Chapter 20, Lesson 1

CON-2.I.2 - There are standard array algorithms that utilize traversals to: * Shift or rotate elements left or right * Reverse the order of the elements	Chapter 17, Lesson 4 Chapter 19, Lessons 2-3 Chapter 20, Lesson 1

UNIT 7: ArrayList	**CITATION(S)**
TOPIC 7.1: Introduction to ArrayList	
VAR-2.D.1 - An ArrayList object is mutable and contains object references.	Chapter 14, Lesson 4
VAR-2.D.2 - The ArrayList constructor ArrayList() constructs an empty list.	Chapter 14, Lesson 4
VAR-2.D.3 - Java allows the generic type ArrayList<E>, where the generic type Especifies the type of the elements.	Chapter 14, Lesson 4
VAR-2.D.4 - When ArrayList<E> is specified, the types of the reference parameters and return type when using the methods are type E.	Chapter 14, Lesson 4
VAR-2.D.5 - ArrayList<E> is preferred over ArrayList because it allows the compiler to find errors that would otherwise be found at run-time.	Chapter 14, Lesson 4
TOPIC 7.2: ArrayList Methods	
VAR-2.D.6 - 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.	Chapter 14, Lesson 4
VAR-2.D.7 - The following ArrayList methods—including what they do and when they are used—are part of the Java Quick Reference:	Chapter 14, Lesson 4
int size() -Returns the number of elements in the list	Chapter 14, Lesson 4
boolean add(E obj) - Appends obj to end of list; returns true	Chapter 14, Lesson 4
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	Chapter 14, Lesson 4
E get(int index) - Returns the element at position index in the list	Chapter 14, Lesson 4
E set(int index, E obj) — Replaces the element at position index with obj; returns the element formerly at position index	Chapter 14, Lesson 4
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	Chapter 14, Lesson 4
TOPIC 7.3: Traversing ArrayLists	
VAR-2.E.1 - Iteration statements can be used to access all the elements in an ArrayList. This is called traversing the ArrayList.	Chapter 14, Lesson 4 Chapter 14, Lesson 5
VAR-2.E.2 - Deleting elements during a traversal of an ArrayList requires using special techniques to avoid skipping elements.	Chapter 14, Lesson 4
VAR-2.E.3 - Since the indices for an ArrayListstart at 0 and end at the number of elements − 1, accessing an index value outside of this range will result in an ArrayIndexOutOfBoundsExceptionbeing thrown.	Chapter 14, Lesson 4

VAR-2.E.4 - 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.	Chapter 14, Lesson 5
TOPIC 7.4: Developing Algorithms Using ArrayLists	
CON-2.J.1 - There are standard ArrayList algorithms that utilize traversals to: * Insert elements * Delete elements * Apply the same standard algorithms that are used with 1D arrays	Chapter 17, Lesson 4 Chapter 19, Lessons 2-3 Chapter 20, Lesson 1
CON-2.J.2 - Some algorithms require multiple String, array, or ArrayList objects to be traversed simultaneously.	Chapter 14 Activity
TOPIC 7.5: Searching	
CON-2.K.1 - There are standard algorithms for searching.	Chapter 19, Lesson 3
CON-2.K.2 - 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.	Chapter 19, Lesson 3
TOPIC 7.6: Sorting	
CON-2.L.1 - Selection sort and insertion sort are iterative sorting algorithms that can be used to sort elements in an array or ArrayList.	Chapter 19, Lesson 2
CON-2.M.1 - Informal run-time comparisons of program code segments can be made using statement execution counts.	Chapter 20, Lessons 2-3
TOPIC 7.7: Ethical Issues Around Data Collection	
IOC-1.B.1 - When using the computer, personal privacy is at risk. Programmers should attempt to safeguard personal privacy.	Chapter 1, Lessons 4-5 Suppl. Chapter 3, Lesson 1
IOC-1.B.2 - Computer use and the creation of programs have an impact on personal security. These impacts can be beneficial and/or harmful.	Chapter 1, Lessons 4-5 Suppl. Chapter 3, Lesson 1

UNIT 8: 2D Array	**CITATION(S)**
TOPIC 8.1: 2D Arrays	
VAR-2.F.1 - 2D arrays are stored as arrays of arrays. Therefore, the way 2D arrays are created and indexed is similar to 1D array objects.	Chapter 14, Lesson 2
VAR-2.F.2 - 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.	Chapter 14, Lesson 2
VAR-2.F.3 - The initializer list used to create and initialize a 2D array consists of initializer lists that represent 1D arrays	Chapter 14, Lesson 2
VAR-2.F.4 - The square brackets [row][col] are used to access and modify an element in a 2D array	Chapter 14, Lesson 2
VAR-2.F.5 - “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.	Chapter 14, Lesson 2

TOPIC 8.2: Traversing 2D Arrays	
VAR-2.G.1 - 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.	Chapter 14, Lesson 2
VAR-2.G.2 - Nested iteration statements can be written to traverse the 2D array in "row-major order" or "column-major order."	Chapter 14, Lesson 2
VAR-2.G.3 - 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.	Chapter 14, Lesson 5
CON-2.N.1 - 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.	Chapter 19, Lesson 3
CON-2.N.2 - All standard 1D array algorithms can be applied to 2D array objects.	Chapter 14, Lesson 2

UNIT 9: Inheritance	**CITATION(S)**
TOPIC 9.1: Creating Superclasses and Subclasses	
MOD-3.B.1 - A class hierarchy can be developed by putting common attributes and behaviors of related classes into a single class called a superclass.	Chapter 15, Lesson 2
MOD-3.B.2 - Classes that extend a superclass, called subclasses, can draw upon the existing attributes and behaviors of the superclass without repeating these in the code.	Chapter 15, Lesson 2
MOD-3.B.3 - Extending a subclass from a superclass creates an "is-a" relationship from the subclass to the superclass.	Chapter 15, Lesson 2
MOD-3.B.4 - The keyword extends is used to establish an inheritance relationship between a subclass and a superclass. A class can extend only one superclass.	Chapter 15, Lesson 2
TOPIC 9.2: Writing Constructors for Subclasses	
MOD-3.B.5 - Constructors are not inherited.	Chapter 15, Lesson 6
MOD-3.B.6 - The superclass constructor can be called from the first line of a subclass constructor by using the keyword super and passing appropriate parameters.	Chapter 15, Lesson 6
MOD-3.B.7 - 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.	Chapter 15, Lesson 6
MOD-3.B.8 - 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.	Chapter 15, Lesson 6

MOD-3.B.9 - Regardless of whether the superclass constructor is called implicitly or explicitly, the process of calling superclass constructors continues until the Object constructor is called. At this point, all of the constructors within the hierarchy execute beginning with the Object constructor.	Chapter 15, Lesson 6
TOPIC 9.3: Overriding Methods	
MOD-3.B.10 - Method overriding occurs when a public method in a subclass has the same method signature as a public method in the superclass.	Chapter 15, Lesson 4
MOD-3.B.11 - Any method that is called must be defined within its own class or its superclass.	Chapter 15, Lesson 4
MOD-3.B.12 - A subclass is usually designed to have modified (overridden) or additional methods or instance variables	Chapter 15, Lesson 4
MOD-3.B.13 - A subclass will inherit all public methods from the superclass; these methods remain public in the subclass.	Chapter 15, Lesson 4
TOPIC 9.4: super Keyword	
MOD-3.B.14 - The keyword super can be used to call a superclass's constructors and methods.	Chapter 15, Lesson 6
MOD-3.B.15 - The superclass method can be called in a subclass by using the keyword super with the method name and passing appropriate parameters.	Chapter 15, Lesson 6
TOPIC 9.5: Creating References Using Inheritance Hierarchies	
MOD-3.C.1 - When a class S "is-a" class T, T is referred to as a superclass, and S is referred to as a subclass.	Chapter 15, Lesson 2
MOD-3.C.2 - If S is a subclass of T, then assigning an object of type S to a reference of type T facilitates polymorphism.	Chapter 15, Lesson 3
MOD-3.C.3 - If S is a subclass of T, then a reference of type T can be used to refer to an object of type Tor S.	Chapter 15, Lesson 3
MOD-3.C.4 - Declaring references of type T, when S is a subclass of T, is useful in the following declarations: * Formal method parameters * arrays — T[]varArrayList<T>var	Chapter 15, Lesson 3 Chapter 16 Activities
TOPIC 9.6: Polymorphism	
MOD-3.D.1 - Utilize the Object class through inheritance.	Chapter 15, Lesson 5
MOD-3.D.2 - At compile time, methods in or inherited by the declared type determine the correctness of a non-static method call.	Chapter 15, Lesson 5
MOD-3.D.3 - At run-time, the method in the actual object type is executed for a non-static method call	Chapter 15, Lesson 5
TOPIC 9.7: Object Superclass	
MOD-3.E.1 - The Object class is the superclass of all other classes in Java.	Chapter 15, Lesson 5
MOD-3.E.2 - The Object class is part of the java.lang package	Chapter 15, Lesson 5
MOD-3.E.3 - The following Object class methods and constructors—including what they do and when they are used—are part of the Java Quick Reference: * boolean equals(Object other) * String toString()	Chapter 15, Lesson 5

MOD-3.E.4 - Subclasses of Object often override the equals and toString methods with class-specific implementations.	Chapter 15, Lesson 5

UNIT 10: Recursion	**CITATION(S)**
TOPIC 10.1: Recursion	
CON-2.O.1 - A recursive method is a method that calls itself.	Chapter 19, Lesson 1
CON-2.O.2 - Recursive methods contain at least one base case, which halts the recursion, and at least one recursive call.	Chapter 19, Lesson 1
CON-2.O.3 - Each recursive call has its own set of local variables, including the formal parameters.	Chapter 19, Lesson 1
CON-2.O.4 - Parameter values capture the progress of a recursive process, much like loop control variable values capture the progress of a loop.	Chapter 19, Lesson 1
CON-2.O.5 - Any recursive solution can be replicated through the use of an iterative approach.	Chapter 19, Lesson 1
CON-2.O.6 - Recursion can be used to traverse String, array, and ArrayList objects.	Chapter 19, Lesson 1
TOPIC 10.2: Recursive Searching and Sorting	
CON-2.P.1 - Data must be in sorted order to use the binary search algorithm.	Chapter 19, Lesson 3
CON-2.P.2 - 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.	Chapter 19, Lesson 3
CON-2.P.3 - Binary search can be more efficient than sequential/linear search.	Chapter 19, Lesson 3
CON-2.P.4 - The binary search algorithm can be written either iteratively or recursively.	Chapter 19, Lesson 3 Chapter 19 Activity
CON-2.Q.1 - Merge sort is a recursive sorting algorithm that can be used to sort elements in an array or ArrayList.	Chapter 19, Lesson 2