AP Computer Science A

Semester A Summary:

The AP® Computer Science A course is equivalent to the first semester of a college level computer science course. The course involves developing the skills to write programs or part of programs to correctly solve specific problems. AP Computer Science A also emphasizes the design issues that make programs understandable, adaptable, and when appropriate, reusable. At the same time, the development of useful computer programs and classes is used as a context for introducing other important concepts in computer science, including the development and analysis of algorithms, the development and use of fundamental data structures, and the study of standard algorithms and typical applications. In addition, an understanding of the basic hardware and software components of computer systems and the responsible use of these systems are integral parts of the course.

Semester A Outline

1. **Getting Started**
    1. Orientation
    2. Course Folder Management
    3. Installing Java
        - Download, install, and setup the Java programming language.
        - Describe a brief history of the development of Java.
    4. Installing the BlueJ IDE
        - Describe the purpose of an Interactive Development Environment.
        - Install the BlueJ IDE.
    5. BlueJ Tutorial
        - Describe the layout of the BlueJ IDE and it major features.
        - Create, open, and compile a project.
        - Create, edit, compile, and remove a class.
        - Run a Java program.
    6. Hello World
        - Implement programming constructs to print output to the console (CB II-B3).
    7. Stylish Java/Sucessful Habits
        - Describe the general conventions of Java style.
    8. Check Point Alpha
        - Describe the purpose of a post mortem review (PMR).
        - Assess your understanding with a self-evaluation checklist for Unit 1.
2. **Arithmetic Expressions and Variables**
    1. Introduction
    2. Order of Operations
        - Evaluate arithmetic expressions using the order of operations.
    3. Printing Arithmetic Expression
        - Implement programming constructs to print output to the console [CB II-B3].

    - Write syntactically correct arithmetic expressions that comply with the order of operations.
4. Primitive Data Types: ints
    - Use primitive data types (e.g., ints) in arithmetic expressions [CB IV-].
    - Understand and modify existing code [CB III-C].
5. Primitive Data Types: doubles
    - Use primitive data types (e.g., doubles) in arithmetic expressions [CB IV-A].
    - Understand and modify existing code [CB III-C].
6. Arithmetic Expressions
    - Implement programming constructs to declare and assign variables [CB II-B2a].
    - Apply the rules for correctly naming variables.
7. Primitive Data Type Conversions
    - Analyze and use primitive data type conversions in arithmetic expressions.
    - Understand the significance of a byte in the context of primary memory storage [CB V-A1].
8. Pitfalls, Surprises, and Shortcuts
    - Analyze the finite representations and limits of numerical values (e.g., integer bounds, floating point imprecision, and round-off errors) [CB III-A:H2].
    - Implement programming constructs to print output to the console [CB II-B3].
    - Write syntactically correct arithmetic expressions that use shortcut operators.
9. Challenge Program
    - Implement programming constructs to print output to the console [CB II-B3].
    - Write syntactically correct arithmetic expressions that comply with the order of operations.
10. Checkpoint Beta

3. **Getting Started with Strings**
    1. Getting Started with Strings
    2. Pseudocode, Recipe for Success
        - Describe the use of algorithmic thinking in program design.
        - Write pseudocode for simple tasks.
    3. Primitive Data Types: char
        - Interpret the use of primitive data types (e.g. chars) in expressions [CB IV-A].
    4. String Objects
        - Implement String object programming constructs [CB II-B1].
    5. Escape Sequences
        - Utilize escape sequences when printing String literals.
    6. The Java API
        - Read and understand method summaries and details in the Java API.
    7. String Class Methods: The Basics
        - Implement programs using Java class libraries. [CB II-C]
        - Implement programming constructs to declare classes.[CB II-B2c]
    8. The Scanner Class
        - Implement programs using Java class libraries. [CB II-C]
    9. Parsing
        - Analyze the structure and implementation of a simple class.
    10. Challenge Program
        - Debug programs by identifying and correcting errors. [CB III-B2]
        - Implement programs using Java class libraries. [CB II-C]
        - Implement programming constructs to declare classes.[CB II-B2c]

        - Implement primitive data type and object programming constructs [CB II-B1]
    11. Checkpoint
    12. Exam
4. **Decisions, Decisions, Decisions**
    1. Getting Started with if Statements
    2. Number Systems
        - Represent decimal numbers in the binary, octal, and hexadecimal number systems [CB III-H1].
    3. Primitive Data Types: Booleans
        - Use primitive data types (e.g., booleans) in expressions [CB IV-A].
        - Understand and modify existing code [CB III-C].
        - Implement conditional control programming constructs [CB II-B4c].
    4. Condition Statements: if
        - Use primitive data types (e.g., booleans) in expressions [CB IV-A].
        - Implement conditional control programming constructs [CB II-B4c].
    5. Condition Statements: if-else
        - Use primitive data types (e.g., booleans) in expressions [CB IV-A].
        - Implement conditional control programming constructs [CB II-B4c].
    6. Condition Statements: if-else-if
        - Use primitive data types (e.g., booleans) in expressions [CB IV-A].
        - Implement conditional control programming constructs [CB II-B4c].
    7. Comparing Strings
        - Use primitive data types (e.g., booleans) in expressions [CB IV-A].
        - Implement conditional control programming constructs [CB II-B4c].
        - Implement Java library classes [CB II-C].
    8. Logical Operators
        - Use primitive data types (e.g., booleans) in expressions [CB IV-A].
        - Implement conditional control programming constructs [CB II-B4c].
    9. Checkpoint
        - Apply Boolean logic to a practical situation.
5. **Iterative Control Structures**
    1. Getting Started with loops
    2. while Loops (part 1)
        - Implement iterative control programming constructs [CB I-B4d].
        - Implement Java class libraries [CB II-C].
    3. while Loops (part 2)
        - Implement iterative control programming constructs [CB I -B4d].
        - Implement Java class libraries [CB II-C].
    4. Reading Text Files
        - Implement iterative control programming constructs [CB I-B4d].
        - Implement Java library classes [CB II-C].
        - Read information from sequential files.
    5. for Loops
        - Implement iterative control programming constructs [CB I-B4d].
        - Implement Java library classes [CB II-C].
    6. Nested Loops
        - Implement iterative control programming constructs [CB I-B4d].
        - Implement Java class libraries [CB II-C].
    7. Writing Text Files
        - Implement iterative control programming constructs [CB I-B4d].
        - Implement Java class libraries [CB II-C].

    - Employ techniques such as using a debugger, adding extra output statements, or hand-tracing code [CB III B-3].
8. Challenge Program
    - Implement iterative control programming constructs [CB I-B4d].
    - Implement Java class libraries [CB II-C].
    - Discuss responsible use of computer systems in relation to privacy [CB VI D4].
9. Checkpoint
10. Discussions

6. **Arrays**
    1. Getting Started with Arrays
    2. One Dimensional Arrays
        - Implement standard data structures—one-dimensional arrays [CB IV-C].
    3. Formatting Output
        - Implement programming constructs to print output to the console (CB IV-C).
        - Understand and modify existing code (CB III-C).
        - Implement iterative control programming constructs (CB I-B4d).
    4. for-each Loops
        - Implement standard data structures—one-dimensional arrays (CB IV-C).
        - Implement iterative control programming constructs (CB I-B4d).
    5. Challenge Program
        - Implement standard data structures—one-dimensional arrays (CB IV-C).
        - Implement iterative control programming constructs (CB I-B4d).
    6. Checkpoint/Challenge Exam
7. **Methods**
    1. Getting Started with Methods
    2. The Math Class
        - Implement Java library classes [CB IIC].
    3. Defining New Static Methods: Part 1
        - Implement top-down program development [CB II-A1b].
        - Implement procedural abstraction [CB II-A1d].
        - Implement program constructs for method declarations [CB II-B2e].
        - Implement program control with methods [CB II-B4a].
    4. Defining New Static Methods: Part 2
        - Implement top-down development techniques [CB II-A1b].
        - Implement procedural abstraction [CB II-A1d].
        - Create method declarations [CB II-B2e].
        - Utilize methods for program flow of control [CB II-B4a].
    5. Defining New Static Methods: Part 3
        - Implement top-down development techniques [CB II-A1b].
        - Implement procedural abstraction [CB II-A1d].
        - Create method declarations [CB II-B2e].
        - Utilize methods for program flow of control [CB II-B4a].
        - Utilize comments to internally document programs.
        - Explain the purpose of the main() method.
        - Write variable declaration statements with multiple variables.
    6. Challenge Program
        - Implement top-down development techniques [CB II-A1b].
        - Implement procedural abstraction [CB II-A1d].
        - Create method declarations [CB II-B2e].
        - Utilize methods for program flow of control [CB II-B4a].
        - Utilize comments to internally document programs.

    7. Checkpoint

8. **Object Oriented Programming**
    1. Getting Started with Objects
    2. Real World Objects
        - Describe the purpose of the Java Virtual Machine.
        - Understand the relationship between classes and objects.
    3. Instances of a Class
        - Define class, object, attribute, and behavior.
        - Analyze the structure of different programming styles.
    4. Default Constructors
        - Design and implement a class [I-B1].
        - Apply functional decomposition [I-B4].
    5. Discussion Topic
        - Discuss the application of computer science to real-world problems.
    6. Constructors that take Parameters
        - Design and implement a class [CB I-B1].
        - Apply functional decomposition [CB I-B4].
        - Implement object-oriented development [CB II-A1a].
        - Implement method declarations [CB II-B2e].
    7. Overloading Methods and Two Classes
        - Read and understand a problem description, purpose, and goals [CB I-A1].
        - Read and understand class specifications and relationships among classes [CB i-A3].
        - Add as many classes as you need.
    8. Constructing Multiple Objects
        - Design and implement a class [CB I-B1].
        - Implement object-oriented development [CB II-A1a].
        - Implement encapsulation and information hiding [CB II-A1c].
        - Categorize errors as compiler, runtime, or logic [CB III-B1].
    9. Arrays of Objects
        - Apply functional decomposition [CB I-B4].
        - Implement object-oriented development [CB II-A1a].
        - Implement encapsulation and information hiding [CB II-A1c].
    10. Java Docs
        - Create extended documentation using javadocs.
        - Read and understand a problem description, purpose, and goals [CB I-A1].
        - Design and implement a class [CB I-B1].
        - Implement object-oriented development [CB II-A1a].
    11. Array Lists I
        - Design and implement an object-oriented class [CB I-B1].
        - Choose appropriate data representation and algorithms [CB I-B3].
    12. Array Lists II
        - Implement the ArrayList data structure.
        - Read and understand a problem description, purpose, and goals. [CB I-A2]
        - Design and implement a class. [CB I-B1]
        - Implement Object-oriented development. [CB II-A1a]
    13. Challenge Program
        - Read and understand a problem description, purpose, and goals [CB I-A1].
        - Apply data abstraction and encapsulation [CB I-A2].
        - Choose appropriate data representation and algorithms [CB I-B3].
    14. Checkpoint
    15. Exam

9. **Analog vs. Digital**
    1. What is a Computer?
    2. Computer Anatomy 101
        - Explain the purpose of primary and secondary memory. [CB VI-A1]
        - Explain the role of the CPU and co-processors. [CB VI-A2]
        - Identify and explain the purpose of common computer peripherals. [CB VI-A3]
        - Describe the purpose of an operating system. [CB VI-B3]
        - Describe a single-user computer system. [CB VI-C1]
        - Describe a networked computer system. [CB VI-C2]
        - Distinguish between local area and wide area networks.
    3. Computer History: Back in the Day
        - Describe the early history of calculating devices.
        - Explain the differences between the four generations of computers.
    4. Four Generations of Modern Computers
        - Describe the characteristics of the four generations of modern computers.
        - Explain the impact of Moore's Law.
    5. Challenge Program
        - Read and understand a problem description, purpose, and goals. [CB I A-1]
        - Design and implement a class. [CB I-B1]
        - Apply functional decomposition. [CB I-B4]
        - Create and manipulate a two dimensional array.
    6. Checkpoint
10. **Semester Exam**
    1. Semester Exam

Semester B Summary:

The AP® Computer Science course is equivalent to the first semester of a college level computer science course. The course involves developing the skills to write programs or part of programs to correctly solve specific problems. AP Computer Science also emphasizes the design issues that make programs understandable, adaptable, and when appropriate, reusable. At the same time, the development of useful computer programs and classes is used as a context for introducing other important concepts in computer science, including the development and analysis of algorithms, the development and use of fundamental data structures, and the study of standard algorithms and typical applications. In addition, an understanding of the basic hardware and software components of computer systems and the responsible use of these systems are integral parts of the course.

Semester B Outline

1. **Computing in Context**
    1. Technology and Society
    2. Privacy Issues
        - Discuss privacy issues and the responsible use of technology. [CB VI-D2]
        - Take precautions to avoid identity theft.
    3. Security Issues
        - Describe the different types of malware.
        - Take precautions to protect the security of your computer system.

    4. Legal Issues
        - Discuss legal issues related to intellectual property and the responsible use of technology. [CB VI-D3]
        - Take precautions to avoid becoming victimized by online criminals.
    5. Future Issues
        - Reflect upon the impact that future technologies will have on society.
    6. API Practice Lab
        - the pow() and sqrt() methods of the Math class
        - the compareTo(), equals(), length(), substring(), and indexOf() methods of the String class
        - the size() and get() methods of the List interface
        - method syntax, parameters, and return types
        - turning requirements into code
        - list references as arguments
        - list processing
        - for loops
        - conditionals
        - troubleshooting and problem solving
    7. Check Point

2. **Recursion**
    1. Getting Started with Recursion
    2. Divide et Impera
        - Describe the principle of divide and conquer.
    3. Real World Recursion
        - Begin to recognize recursion in real world situations.
    4. The Recursive Leap of Faith
        - Describe the basic concept of recursion.
    5. There and Back Again
        - Analyze Piecewise functions using the Simplify-Substitute-Solve strategy.
        - Understand and evaluate recursive methods. [CB II-B4e]
    6. Are We There Yet?
        - Implement recursive methods. [CB II-B4e]
    7. Challenge Program
        - Implement recursive methods. [CB II-B4e]
    8. Mystery Message
        - Apply divide and conquer and recursive thinking to a real world situation.
    9. Create Your Own Challenge Exam
    10. Tic-Tac-Toe Lab: Part 1
        - By completing this lesson, you will gain experience in the following subjects:
        - Conditionals
        - While and for loops
        - 1-D and 2-D arrays
        - Arrays of references
        - Classes and objects
        - Method syntax, parameters, and return values
        - Method overloading and overriding
        - Object oriented programming
        - Encapsulation
        - Inheritance
        - Polymorphism
        - Abstract classes and methods
        - Constructors

        - Exception handling
        - Java API, String methods
        - Random number generation
        - Troubleshooting and problem solving
    11. Checkpoint
3. **Inheritance and Polymorphism**
    1. Introduction to Inheritance and Polymorphism
    2. Extending Classes
        - Read and understand class specifications and relationships among classes [CB 1A3].
        - Extend a given class using inheritance [CB 1B5].
    3. Class Hierachies
        - Understand and implement a given class hierarchy [CB 1A4].
    4. Polymorphism
        - Create polymorphic methods.
        - Identify reusable components from existing code, using classes and class libraries [CB 1A5].
    5. Overriding Methods
        - Override methods in a subclass.
        - Override toString() and equals() from the Object class.
        - Extend existing code using inheritance [CB 3D].
    6. Challenge Program
    7. Tic-Tac-Toe Lab: Part 2
        - By completing this lesson, you will gain experience with the following subjects:
        - Conditionals
        - While and for loops
        - 1-D and 2-D arrays
        - Arrays of references
        - Classes and objects
        - Method syntax, parameters, and return values
        - Method overloading and overriding
        - Object oriented programming
        - Encapsulation
        - Inheritance
        - Polymorphism
        - Abstract classes and methods
        - Constructors
        - Exception handling
        - Java API, String methods
        - Random number generation
        - Troubleshooting and problem solving
4. **Revisiting Classes**
    1. Getting Back to Basics
    2. Design Strategy: Iteratiive and Incremental
        - Understand the program development cycle.
        - Utilize object-oriented development methodology. [CB II-A1a]
        - Write overloaded method declarations. [CB II-B2e]
    3. Static Means Never Having to Instantiate an Object
        - Write class (or static) declarations and methods. [CB II-B2c]
    4. Class Variables and Constants
        - Declare and use class variables. [CB B-2b]

    - Declare and use constants. [CB B-2a]
  5. Revisiting Randomness
    - Choose appropriate data representation and algorithms. [CB I-B3]

  6. this or That Variable
    - Utilize the keyword this to distinguish between instance and local variables.
  7. Thinking Outside the Box
    - Design and implement a class. [CB I-B1]
    - Apply functional decomposition. [CB I-B4]
    - Implement object-oriented development. [II-A1a]
  8. Challenge Program
    - Read and understand a problem description, purpose, and goals. [CB I-A1]
    - Apply data abstraction and encapsulation. [CB I-A2]
    - Design and implement a class. [CB I-B1]
    - Choose appropriate data representation and algorithms. [CB I-B3]
    - Apply encapsulation and information hiding. [CB II-A1c]
  9. Checkpoint
  10. Challenge Exam
  11. Discussion
  12. Tic-Tac-Toe Lab: Part 3
    - By completing this lesson, you will gain experience with the following subjects:
    - Conditionals
    - While and for loops
    - 1-D and 2-D arrays
    - Arrays of references
    - Classes and objects
    - Method syntax, parameters, and return values
    - Method overloading and overriding
    - Object oriented programming
    - Encapsulation
    - Inheritance
    - Polymorphism
    - Abstract classes and methods
    - Constructors
    - Exception handling
    - Java API, String methods
    - Random number generation
    - Troubleshooting and problem solving

5. **Abstraction**
  1. Introduction to Abstractions
  2. Abstract Classes
    - Read and understand procedural abstraction [CB 2A1D].
    - Create an abstract class and extend it.
  3. Built-In Interfaces
    - Create a List reference variable that refers to an ArrayList

  4. Interfaces
    - Read and understand interface declarations [CB 2B2D].
    - Create an interface and implement it
  5. Comparable T Interface
    - Understand the Comparable<T> interface.
    - Implement the Comparable<T> interface

    6. Challenge Program
    7. Challenge Exam
    8. Iteration and Recursion Lab: Part 1
        - By completing this lesson, you will gain experience with the following subjects:
        - iteration
        - recursion
        - method syntax, parameters, and return types
        - turning requirements into code
        - array references as arguments
        - array processing
        - type conversions
        - for loops
        - conditionals
        - exception handling
        - troubleshooting and problem solving

6. **Standard Algorithms**
    1. Introduction to Standard Algorithms
        - Choose appropriate data representation and algorithms [CB-1B3].
        - Understand traversals.
    2. Traversals
        - Choose appropriate data representation and algorithms [CB-1B3].
        - Understand traversals.
    3. Replacements
        - Choose appropriate data representation and algorithms [CB-1B3].
        - Understand traversals.
    4. Insertions
        - Choose appropriate data representation and algorithms [CB-1B3].
        - Understand insertions.
    5. Deletions
        - Choose appropriate data representation and algorithms [CB-1B3].
        - Understand deletions.
    6. Challenge Program
    7. Iteration and Recursion Lab: Part 2
        - By completing this lesson, you will gain experience with the following subjects:
        - iteration
        - recursion
        - method syntax, parameters, and return types
        - turning requirements into code
        - array references as arguments
        - array processing
        - type conversions
        - for loops
        - conditionals
        - exception handling
        - troubleshooting and problem solving
    8. Checklist
7. **Sorting**
    1. Introduction to Sorting
    2. Bubble Sort
        - Choose appropriate data representation and algorithms [CB-1B3].

        - Understand bubble sort.
    3. Insertion Sort
        - Choose appropriate data representation and algorithms [CB-1B3].
        - Understand insertion sort.
    4. Selection Sort
        - Choose appropriate data representation and algorithms [CB-1B3].
        - Understand selection sort.
    5. Merge Sort
        - Choose appropriate data representation and algorithms [CB-1B3].
        - Understand merge sort.
    6. Challege Program
    7. Iteration and Recursion Lab: Part 3
        - By completing this lesson, you will gain experience with the following subjects:
        - iteration
        - recursion
        - method syntax, parameters, and return types
        - turning requirements into code
        - array references as arguments
        - array processing
        - type conversions
        - for loops
        - conditionals
        - exception handling
        - troubleshooting and problem solving
    8. Checklist
8. **Searching**
    1. Introduction to Searching
    2. Sequential Search
        - Choose appropriate data representation and algorithms [CB 1B3].
        - Understand sequential search.
    3. Binary Search
        - Choose appropriate data representation and algorithms [CB 1B3].
        - Understand binary search.
    4. Challenge Program
    5. Challenge Exam
    6. Sorting Lab: Part 1
        - By completing this lesson, you will gain experience with the following subjects:
        - insertion Sort
        - selection Sort
        - sorting strategy and implementation
        - comparisons
        - conditionals
        - array navigation and manipulation
        - for loops
        - while loops
        - object-oriented programming
        - constructors
        - method invocation, passing arguments to methods
        - exception handling
        - troubleshooting and problem solving

  - This lesson also demonstrates, but does not require the implementation of the following:
  - swing programming
  - enumerations
  - nested classes
  - constants
  - encapsulation
  - method syntax, parameters, and return values
  - method overriding
  - encapsulation
  - inheritance
  - polymorphism
7. Checklist

9. **Program Analysis**
   1. Introduction to Program Analysis
   2. Assertions and Exceptions
      - Perform integration testing [CB-3A3].
      - Understand runtime exceptions [CB-3E1].
   3. Challenge Assignment
   4. Sorting Lab: Part 2
      - By completing this lesson, you will gain experience with the following subjects:
      - insertion Sort
      - selection Sort
      - sorting strategy and implementation
      - comparisons
      - conditionals
      - array navigation and manipulation
      - for loops
      - while loops
      - object-oriented programming
      - constructors
      - method invocation, passing arguments to methods
      - exception handling
      - troubleshooting and problem solving
      - This lesson also demonstrates, but does not require the implementation of the following:
      - swing programming
      - enumerations
      - nested classes
      - constants
      - encapsulation
      - method syntax, parameters, and return values
      - method overriding
      - encapsulation
      - inheritance
      - polymorphism
   5. Checklist
10. **Final Exam and AP Exam Review**
    1. Getting Started with Your Review
    2. Exam Format, Grading, Hints
    3. Java Features, Part 1

4. Java Features, Part 2
5. Program Design and OOP Concepts
6. Algorithms
7. Sorting Lab: Part 3
    - By completing this lesson, you will gain experience with the following subjects:
    - insertion Sort
    - selection Sort
    - sorting strategy and implementation
    - comparisons
    - conditionals
    - array navigation and manipulation
    - for loops
    - while loops
    - object-oriented programming
    - constructors
    - method invocation, passing arguments to methods
    - exception handling
    - troubleshooting and problem solving
    - This lesson also demonstrates, but does not require the implementation of the following:
    - swing programming
    - enumerations
    - nested classes
    - constants
    - encapsulation
    - method syntax, parameters, and return values
    - method overriding
    - encapsulation
    - inheritance
    - polymorphism
8. Solutions to Past Free Response Questions
9. Practice Exams
10. 2004 Released AP Computer Science A Exam
11. Reflections
12. Final Exam