AP® Computer Science A

Advanced Placement® Computer Science A is a fast-paced course equivalent to a college introductory programming class. Students will learn about the exciting kinds of problems tackled by computer science while exploring the field's most important tool—programming. The focus will be on developing systematic problem-solving strategies that can be applied to real-world problems. The course will be anchored around projects that will explore a broad range of fields that use programming to solve problems. Through these projects, students will study common, reusable algorithms and learn to analyze them for correctness and speed.

The course will cover fundamentals of programming syntax and methodology using the Java programming language. Java is a modern, object-oriented programming language used to create professional software. In addition to gaining fluency in Java, students will develop general skills and understandings in computer science.

Prerequisites

Students enrolled in this course are expected to have successfully completed Algebra 1 or its equivalent. They must also have strong writing and reading comprehension skills, since much of the practice of computer science requires clear communication of ideas and concepts.

Course materials

Reges, Stuart, and Stepp, Martin. Building Java Programs: A Back to Basics Approach. 4th ed. (Pearson, 2016).

Philosophy

This course will emphasize procedural decomposition, object use and algorithm design early in the curriculum. Writing object classes will be covered in the latter part of the course to insure that students have a solid foundation in programming fundamentals before moving on to more advanced topics in object-oriented software. Each new concept will be presented in a short, interactive lecture and be followed up by small-scale programming exercises. Students will then complete larger projects that use programming to explore interesting problem domains. **Students are engaged in hands-on laboratory experiences, integrated throughout the course, which account for 20 hours of course time.** Short quizzes similar to the small-scale programming exercises will provide students with frequent feedback on the depth of their understanding of the material.

Curricular Requirements	**Page**
CR1 The course teaches students to design and implement computer based solutions to problems.	3,4,5,6,7,8
CR2a The course teaches students to use and implement commonly used algorithms.	4,6,7
CR2b The course teaches students to use commonly used data structures.	4
CR3 The course teaches students to select appropriate algorithms and data structures to solve problems.	4,6,7,8
CR4 The course teaches students to code fluently in an object-oriented paradigm using the programming language Java.	5,6
CR5 The course teaches students to use elements of the standard Java library from the AP Java subset in Appendix A of the AP Computer Science A Course Description.	5,6
CR6 The course includes a structured lab component comprised of a minimum of 20 hours of hands-on lab experiences.	1
CR7 The course teaches students to recognize the ethical and social implications of computer use.	8

Unit Title	***# weeks***	***Topics Covered***	***Objectives (SWBAT…)***	***Projects/Activities***	***BJP(4th ed)/ Practice-It***	***BJP reference***
First Semester						
Unit 1: Introduction to Programming and Java	2	Definitions of "algorithm" and "computer science"; Computational thinking skills; `String` and console output; Procedural decomposition; `static void` methods	Define "algorithm" and "computer science"; Create simple programs to print output to the console; Break complex problems into well-defined subtasks; Define and call `static void` methods	Logic problems Hello World Song mini-project (BJP ch. 1 Programming Projects #1-3, 5)	Chapter 1 – Self-check #6-9, 11, 14, 22, 23, 26, 29; Exercises #1-9, 11, 12, 14, 16	Chapter 1
Unit 2: Working with Data and Basic Control Flow	3	Concepts of types and variables in programming; Primitive types in Java; Variables and assignment; Arithmetic operators; Compound assignment; Precedence; Casting, type conversion; Concept of scope Concept of control flow; Simple `for` loops	Describe the purpose of types; Define "variable"; List and describe the Java primitive types; Define "precedence"; Write simple expressions and statements using arithmetic operators; Describe the effects when converting among types; Define "control flow"; Write programs that use `for` loops to repeat operations	(BJP ch 2 Programming Projects #1, 4	Chapter 2 – Self-check #1, 3, 4-8, 10, 13-17, 21-24, 26, 27, 29-36 Exercises #1	Sections 2.1-2.4
Unit 3: Advanced Data and Control Flow	4	`String` processing; Nested loops; Parameters; Return values; Generating random numbers; Class constants; Basic Boolean logic Conditionals; Relational and Boolean operators; `while` and `do-while` loops;	Create programs that use nested loops effectively; Define and call methods that have parameters and/or return values; Write an expression to generate a random integer between any two values; Describe when class constants should be used and write code that uses them effectively;	Programming Projects Pokémon Battle; Fraction Calculator project: Create a fractional calculator that can perform basic arithmetic functions on fractional input.	Chapter 3 – Self-check #1-7, 14-21, 24-26 Exercises #1, 12, 14, 15 Chapter 4 – Self-check #1-9 Exercises #1-5 Chapter 5 – Self-check #1-7, 27, 29; Exercises #2, 4-6	Sections 3.1-3.3, 4.1-4.2, 5.1-5.3

		Fencepost and sentinel loops; Console input	Write and evaluate Boolean expressions; Write programs using conditional statements; Write programs using `while` and `do-while` loops; Receive and tokenize console input using `Scanner`			
Unit 4: Arrays, ArrayLists	4	Definition and uses of arrays; Defining and using single-dimensional arrays; Defining and using multi-dimensional arrays; Using arrays as parameters and return values; Reference semantics; `ArrayList` type and API; Enhanced `for` loop (for-each loop) Token-based file input with `Scanner`; Line-based file input; Writing to files with `PrintStream`	Explain what arrays are and why they are useful; Define, populate, access, traverse, and manipulate single- and multi-dimensional arrays; Explain the difference between how primitives and arrays are treated when passed as parameters; Define methods that take arrays as parameters and/or return arrays; Write code that constructs, modifies, and manipulates `ArrayLists`; Compare and contrast arrays and ArrayLists and the scenarios in which each is useful; Define and evaluate enhanced `for` loops; Read input from a file using `Scanner`; Write output to a file using `PrintStream`	AP lab – Magpie;	Chapter 7 – Self-check #1, 7, 9, 12-17, 19-23, 25-30 Exercises #3, 9, 10, 14, 16	Sections 7.1-7.5, 10.1
Second Semester						

Unit 5: Introduction to Object-Oriented Programing	4	Definitions of "encapsulation" and "abstraction"; Difference between class and client; Instance fields and methods; Getters and setters; `public` and `private`; Class fields and methods; Constructors; `equals` and `toString`; `this` keyword	Define "encapsulation" and "abstraction" and explain why they are important; Explain the difference between a class and a client program; Define a custom class; Write a client program to use a custom class; Explain when to use instance fields/methods versus class fields/methods; Explain when to use `public` versus `private` access; Define zero- and non-zero argument constructors; Effectively override the `equals` and `toString` methods	AP lab – Pictures;	Chapter 8 – Self-check #1-5, 9-11, 13-16, 22-30;	Chapter 8
Unit 6: Inheritance and Polymorphism	4	Definitions of "inheritance" and "polymorphism"; Difference between "is-a" and "has-a" relationships; Extending classes; Overriding methods; Constructor mechanics for inherited classes; `super` keyword; Upcasting and downcasting; Polymorphism; Extending and using abstract classes; Implementing and using interfaces	Define "inheritance" and "polymorphism"; Determine whether two classes have an "is-a" or a "has-a" relationship; Define classes that extend another class, including an abstract class; Define classes that implement interfaces; Properly implement constructors of derived classes, using `super` when needed; Trace the execution of one or more methods through a class hierarchy; Determine whether a cast	TextExcel project: 12x22 spreadsheet which can hold text, real numbers, and dates and evaluate formulas (requires creation of several interacting classes including sheet, cells, values). Utilize functions from the Java Math library.	Chapter 9 – Self-check #3, 4, 9, 10, 18, 20; Exercises #4, 8 Programming Project 1, 3 *(Note: Self-check #11-15, 17 are not in Practice-It!)*	Chapter 9

			between two types is valid within a given hierarchy; Determine whether a method call is valid within a given class hierarchy			
Unit 7: Searching and Sorting	3	Basic algorithmic complexity (runtime); Choosing between algorithms to complete a given task; Sequential search; Binary search; Insertion sort; Selection sort;	Identify which of two algorithms has a faster runtime; Define and implement sequential search for various types of collections; Define and implement binary search for various types of collections; Compare and contrast different methods for searching in ordered and unordered lists; Define and implement insertion sort for various types of collections; Define and implement selection sort for various types of collections;	AP lab - Elevens	Chapter 13 – Self-check #4-6, 16-21, 23; Exercises #1-3 *(Note: Most problems are not in Practice-It!)*	Sections 13.1, 13.3 (*Note: Insertion sort is **NOT** covered in BJP)*
Unit 8: Recursion	2	Definition of "recursion"; Possible recursion failures and their causes (infinite recursion); Tracing execution of recursive methods to determine output/return value; Writing simple recursive methods; Recursive binary search; Mergesort	Define "recursion"; Identify methods that will result in infinite recursion and explain why; Compare and contrast recursion and iteration and identify scenarios when one is more effective than the other; Identify the output or return value of recursive methods;		Chapter 12 – Self-check #1-10 Exercises #1, 3 Chapter 13 – Self-check #27-30	Sections 12.1, 12.2, 13.4

			Define simple recursive methods; Implement binary search recursively; Define and implement Mergesort; Compare and contrast the running times of insertion, selection, and merge sort for different applications			
Unit 9: Review for AP Exam	3	Review of previous topics as needed; Format and structure of AP exam; AP free response scoring guidelines; Common AP multiple choice question formats/topics; Test-taking strategies	Describe the format of the AP Computer Science exam; Explain how AP free response questions are scored; List common topics and formats for AP multiple choice questions	Practice AP exams Practice AP questions	N/A	N/A
Unit 10: Post-AP Exam Project	4-5	Project and/or projects of the teachers' and/or students' choice	TBD by teacher	N/A	N/A	N/A
Throughout Year						
Unit A: Computing Careers and Culture	N/A	Computing in society; computing college and career options; beneficial and harmful effects of technology; impact of technology on everyday life; Ethical and social implications of computer science	Describe the impact of computing and technology on society and culture; Identify possible college and career options in computing; Discuss current events in computing and technology	College/career panel; Guest speakers; Videos (e.g. TED talks); Supplemental readings (e.g. Blown to Bits); Students presentations on current events addressing ethical challenges that accompany the advancement of computing; Journal entries	N/A	N/A