AP Computer Science A

Teacher: Mr. Alvey
Phone: 435-628-5255 x129
Email: alvey@pineview.org
Class Web Page: http://alveyworld.pineview.org

Course Description

AP® Computer Science A is both a college-prep course for potential computer science majors and a foundation course for students planning to study in other technical fields such as engineering, physics, chemistry, geology, etc. The course is all about learning to be a problem solver, focusing on programming methodology, procedural abstraction, and in-depth study of algorithms, data structures, and data abstractions, as well as a detailed examination of a large case study program. Instruction includes preparation for the AP Computer Science A Exam. Hardworking students will gain clear understanding of Java and the ability to adapt to any new programming language that will come along. This course is an introduction to the fundamental concepts of computer science and a key to the future of solving the world's biggest problems.

Texts and Online Resources

Jones, William C. *Java Au Naturel*. Fourth Addition. Central Connecticut State University. Copyright 2004. **http://www.javabook.com/**

College Board. *AP GridWorld Case Study.* New York: College Entrance Examination Board, 2006.

Downey, Allen B. *Think Java: How to Think Like a Computer Scientist.* Copyright Allen B. Downey. Open Source Creative Commons License. **http://www.greenteapress.com/thinkapjava/**

Java™ Platform, Standard Edition 7 - API Specification. **http://docs.oracle.com/javase/7/docs/api/**

CodingBat code practice. Copyright Nick Parlante. Standford University. **http://codingbat.com/**

Kahn Academy. Copyright 2012. **http://kahnacademy.com/**

TED Talks. Copyright 2012. **http://ted.com/**

Course Planner

The resources list includes the following text references: Java Au Naturel (JAN), GridWorld Case Study (GW), Think Java (TJ), Java SE API (API)

Unit (Weeks)	Title, Topics, and Student Objectives	Resources, Assessments, and Strategies
1 (0-2)	**Programming Language Basics** **Topics:** • Compiler process, high-level source code to byte code • What is a program? • Debugging and types of errors • Java basics • Introduction to IDEs (BlueJ, Eclipse) **Objectives:** • Understand basic programming terminology: compiler, IDE, executable, source code, syntax, debug • Understand the different compile time errors, runtime errors, and logic errors • Edit, compile and run a simple program in Java • Use input and output with System.out and System.in and BufferedReader	**Resource:** TJ chapter 1 and Appendix B and D, API, BlueJ IDE, Eclipse IDE **Assessment:** • Quiz: Vocabulary, and terminology • Lab: Hello World program in BlueJ and Eclipse **Strategies:** Stress the concept that the programming language is the tool a computer scientist uses to solve problems. We are going to learn one of many tools you may use. We are using Java as a tool to learn computer science, but computer science is not the study of programming languages.
2 (3-5)	**Turtle** *(Introduces objects and inheritance)* **Topics:** • Objects • Classes • Inheritance • Syntax **Objectives:** • Write and use simple classes with Turtles • Learn the basics of Java syntax and	**Resource:** JAN chapter 1, BlueJ IDE **Assessments:** • Program-specific tasks using Turtle • Create a SmartTurtle Class to teach the turtle to make big square and small squares • Use SmartTurtle object to draw three hexagons with two meeting along one side (Exercise 1.11) • Draw flowers with circles, and pedals (Exercise 1.20) • Use turtle object to draw fractal trees*

	objects	(Section 1.10)
3 (6-7)	**Variables and Arithmetic** **Topics:** • Using and understanding variables and type • Variable declaration, and initialization • Comments • Naming rules of variables • Arithmetic operators and expressions in Java programs • Representing numbers in different bases **Objectives:** • Understand terminology comments, variables, constants, reserved words, literals • Declare and initialize variables and constants in Java • Understand mathematical expressions in Java and their precedence • Understand how to change bases of numbers • Understand limitations of finite representations of numbers such as the range of integers, real, and float • Use the assignment operator correctly	**Resource:** TJ chapter 2 **Assessment:** • Play an error finding game called "Stump the Chump" (Exercise 2.1) • Practice developing a program gradually. Use string concatenation to display values with different types, integers and strings. (Exercise 2.2) • Practice using arithmetic operations and compound entities like time of day. Stumble upon integer divison issues. (Exercise 2.3) • Lab: Paycheck program. Have employee information entered and calculate pay. Then update program to include overtime hours in the calculations • Glossary quiz **Strategies:** • Students need time to practice with how the different types, double and int relate when they are used in mathematical operations • Present a lot of small program examples in which they have to find the errors
4 (8-9)	**Methods, Conditionals, and Recursion** **Topics:** • double and int data types, and typecasting • Calling Java library methods • Adding new methods, and invoking them • Methods in a class • Parameters, argument, and returning results • Modulus operator • Conditional statements	**Resource:** TJ chapters 3 and 4, http://www.codingbat.com/ **Assessment:** • Practice reading code and understanding the flow of execution. (Exercises 3.2, 4.2, 4.4) • Write and invoke methods with parameters (Exercises 3.3, 3.4, 4.5) • Write a recursive method to print the counting pattern of a song (Exercise 4.3) • Complete conditional drills using

	• Nested conditionals • Recursion **Objectives:** • Understand the difference between int and double how they interact • Creating new methods with zero to many parameters, and calling (invoking) the methods • Understand the modulus operators function and why it is so useful in solving problems • Use type casting to make their data more accurate • Create conditional statements, with alternate execution, and chained conditionals using if/else if/else • Understand the concept of recursion and recognize it in a program	**Codingbat.com** • Glossary Quiz (both chapters) **Strategies:** • Assign lots of problems for students to read and decide the output and focus on small details. • Assign small programs to practice writing code and learning syntax of methods and classes. • Solve many small problems that can be solved with small conditional statements and using Java methods
5 (10-11)	**GridWorld: Part 1** **Topics:** • Objects, Instances • Constructor • Class • Attributes • Accessor and modifier methods **Objectives:** • Run the case study and analyse output • Understand how the development of a large program came about by reading the chapter of the case study • Observe and experiment with the GridWorld case study • Understand the Bug class • Extend the Bug class by creating a specialized bug to meet some new type of bug requirement	**Resource:** TJ chapter 5, GW Part 1 **Assessment:** • Write a moveBug method for the bug (TJ Exercise 5.1) • Write a randomBug method the uses Math class random (TJ Exercise 5.2) • Write the makeBugs method that creates bug with different colors (TJ Exercise 5.3) • Follow case study Part 1 and experiment. (GW Exercises 1 - 4) **Strategies:** • Read the manual for the case study thoroughly • Be familiar with all the classes and interfaces discussed • Review GUI Summary
6 (12)	**Program development and useful methods**	**Resource:** TJ chapter 6

	Topics: • Fruitful Methods (that return values), Conditional paths and dead ends • Incremental program development • Scaffolding code • Method composition and Overloading • Boolean Expressions and relational operators and Logical operators • Boolean Methods • Revisit Recursion **Objectives:** • Create a recursive method to solve a problem • Understand how to write, call and compose methods that return values • Understand basic design strategies using incremental program development and top down design • Understand how to read and write conditional statements (boolean expressions) using relational and logical operators.	**Assessment:** • Practice writing Boolean method (Exercise 6.1) • Use conditional statements to write a fruitful method (Exercise 6.3) • Understand logical operators and flow of execution (Exercise 6.4) • Use method composition (Exercise 6.5) • Diagram and translate recursive problems from know mathematical algorithms (Exercises 6.6, 6.7, 6.10) • Glossary Quiz **Strategies:** • Assign lots of code to read and to write and study together as well as individually. • Visually represent the flow of the program and how diagram the flow of a recursive solution to a problem. • Study new glossary words and tie them back to previous words
7 (13-14)	**Iteration and Encapsulation** **Topics:** • Assignment operator vs. Equality operator (= vs ==) • The while control statement • Infinite loops • Relationship between recursion and iteration • Encapsulate code into generalized methods • Local variables **Objectives:** • Construct syntactically correct loops and conditional statements • Use while loops to print tabular	**Resource:** TJ chapter 7 **Assessment:** • Study the flow of a program using loops by drawing tables to track variable values and decide output. (Exercise 7.1) • Lots of practice writing methods that use iteration to solve repetitive problems. (Exercises 7.2-76) • Glossary Quiz **Strategies:** • Give students lots of experience reading, following and writing methods

	data • Understand encapsulation and generalizing methods for reuse. • Understand terminology: control statements, infinite loops, local variables	that use the while loop to solve problems.
8 (15-16)	**Strings and Things** **Topics:** • Invoking methods on String objects • Find the length of a String • Traversing a string • Run-time errors and reading the stack trace. (and exceptions) • Reading documentation • Increment and decrement operators (- -, ++) • Strings are immutable **Objectives:** • Understand that Strings are collections of characters • Read and understand String method documentation. (java.lang) • Understand appropriate use of increment and decrement operators • Understand the atypical properties of the String object. (Immutable, incomparable, etc)	**Resource:** TJ chapter 8 **Assessment:** • Write a simple method that traverses a String and manipulates the string. (Exercise 8.1) • Learn to read and understand the stack trace of an error. (Exercise 8.2) • Practice encapsulation using Strings (Exercises 8.3-8.4) • Experiment with data type concatenation (Exercise 8.5) • Read and follow and correct Java programs (Exercises 8.6-8.8) • Write methods to solve problems with Strings (Exercises 8.9-8.12) • Continue study of GridWorld and add mathematical methods. (Exercise 8.13) • Glossary Quiz **Strategies:** • Write several methods that read and manipulate Strings
9 (17-18)	**Mutable Objects** **Topics:** • The Abstract Window Toolkit (java.awt) • import from other packages • new operator for creating object instances (new object references) • Instance variables and dot notation • Points and Rectangles • Objects as parameters and return types	**Resource:** TJ chapter 9, TJ Appendix A **Assessment:** • Read, follow, and track flow of programs that use objects (Exercises 9.1-9.3) • Write programs using BigInteger objects (Exercises 9.4-9.5) • Use objects and java.awt to create simple graphics (Exercises A.1-A.3) • Glossary Quiz **Strategies:**

	• Objects are mutable • null • Aliasing and Garbage collection • Primitive and object types **Objectives:** • Understand which packages are imported automatically (Math, String, etc) • Understand the details of assigning a variable to an new object reference. • Access instance variables using dot notation and changing their values • Use null to initialize object variables • Understand when an object will be claimed by garbage collection.	• Focus on understanding object oriented concepts using analogy and example. • Create visual graphics to help understand the concept of objects • Lots of practice reading, studying, and writing code
10 (19-20)	**GridWorld: Part 2** **Topics**: • BoxBug class • Constructor method • Extending the Bug class • Overriding methods **Objectives**: • Understand that the constructor method is called when we create a new Bug by invoking new. • Understand how extended classes are related to the base class they extend. • Understand how and why we should override class methods	**Resource:** TJ chapter 10, GW Part 2 **Assessment:** • Practice writing classes that extend the Bug class (GW Exercises Part 2:1-4) • Extend Termites from the bug class (Exercises 10.2-10.3) **Strategies:** • Write code using the case study framework to illustrate major object oriented concepts.
11 (21-22)	**The Computer and Society Research Project** **Topics**: • How has the computer affected society? • Evolution and history of computer languages • Ergonomics and lifestyle • Ethics, what can be done/what	**Resource:** Khan Academy, TED.com, Pioneer Online, Internet, library, handouts **Assessment:** • Research paper on the History of computer languages. • Creative paper on the future of computer science and technology in our society

	should be done • Computer careers and fields of study **Objectives**: • Understand history and future of computer science • Understand the role of computer science in solving/creating problems in society • Research computer careers and the need for creative problem solvers	• Research and presentation on the need for computer scientists in our economy • Write about a computer ethics issue of choice **Strategies:** • Research, write, present, and discuss historical and current state of computer science and look a little into the future to see what is coming.
12 (23)	**Creating your own objects** **Topics**: • Defining new classes • class defines an object type • Contructors • Instance variables • Types of object methods (Pure functions, modifiers) • Creating new objects • Algorithms **Objectives**: • Understand that defining a new class also creates a new object type with the same name. • Understand that every object belongs to some object type, that it is an instance of some class. • Understand that invoking new to create an object invokes a constructor method. • Understand that you provide one or more constructors as part of the class definition. • Learn syntax issues about class definitions such as capital letters starting the names of classes.. • Understand that human have the creative task of designing algorithms, and computers have the unintelligent task of executing them.	**Resource:** TJ chapter 11 **Assessment**: • Practice the mechanical part of creating a new class definition and code to test it. (Exercises 11.1, 11.2) • Write a class definition that includes a variety of methods including constructors, modifiers and pure functions (Exercise 11.3) • Glossary Quiz **Strategies**: • Read lots of coded examples of class definitions, and creating instances of objects. • Write and design lots of class definitions • Select appropriate algorithms and data structure to solve problems • Practice simple algorithm design.

13 (24)	**Arrays** **Topics**: • Declaring and initializing array variables • Indexing arrays and access array elements • for loops • Arrays and objects • Random numbers • Counting with arrays **Objectives**: • Understand array syntax with square brackets. • Understand how to iterate through arrays with for loops • Understand that arrays are similar to objects, but have differences • Learn to copy arrays using for loops • Understand how arrays are passed to methods • Understand how to create new arrays of different types.	**Resource:** TJ chapter 12 **Assessment**: • Practice using for loops to iterate through the elements of an array (Exercise 12.1) • Practice using a method to copy and array (Exercise 12.2) • Practice writing random methods (Exercises 12.3-12.4) • Write methods that take arrays as parameters (Exercises 12.5-12.7) • Read and understand code with arrays (Exercises 12.8-12.9) • Write a selection sort to sort arrays (Exercise 12.11) • Design algorithms and write methods to solve simple problems (Exercises 12.12-12.16) • Glossary Quiz **Strategies**: • Lots of practice looping and indexing elements in arrays. • Compare Arrays to objects to highlight differences and similarities. • Select appropriate algorithms and data structures to solve simple problems
14 (25-26)	**Arrays of Objects and Objects of Arrays** **Topics**: • Develop complex programs using OOP • Create Card and Deck classes and associated methods that operate on them • Merge Sort Algorithm • Linear Search, Bisection Search algorithms • Pseudocode **Objectives**: • See the development process of complex programs using objects, arrays, loops, etc.	**Resource:** TJ chapters 13-14 **Assessment**: • Follow along and create the classes and methods from the chapters (Exercises 13.1-13.2) • Extend the functionality of Cards for use in card games. (Exercises 13.3-13.5) • Implement the shuffling and sorting for cards and decks (Exercise 14.1) **Strategies**: • Card and Deck objects case study used to practice object oriented programming techniques • Use the Deck object to see the need for

	(putting it all together) • See the need for the merge sort in a program context. • Understand how to use pseudocode as a tool for design • Understand and compare the use of linear search and bisection search algorithms with one dimensional arrays of objects.	searching and sorting algorithms. • Focused discussion on comparing the linear search algorithm and bisection (binary) search algorithm and which one is the best to solve the problem for the Deck object.
15 (27-28)	**Object oriented programming and Inheritance** **Topics**: • Pure OOP (not just using objects for procedural programming) • Object methods vs. class methods (static methods). • Implicit vs Explicit references • Overriding methods (such as toString(), equals()) • this • Inheritance, class hierarchy **Objectives**: • Understand the purpose of using object oriented programming • Understand object inheritance • Understand when it is appropriate to override methods • Create object methods that use implicit references and this.	**Resource:** TJ chapter 15 **Assessment**: • Transform class methods to object methods. (Exercises 15.1-15.4) • Continue using Card and Deck objects to write another card game program. (Exercise 15.5) • Glossary Quiz **Strategies**: • Continue the case study and introduce new object oriented concepts and design techniques.
16 (29-31)	**Gridworld: Part 3** **Topics**: • ArrayList (collections) • Type Parameters • Java Interfaces, Implementations • API (Application Programming Interface) • public and private **Objectives**: • Understand Java Interfaces	**Resource:** TJ chapter 16 **Assessment**: • Customize bug class definitions that reacts to different objects on the grid. (Exercise 16.1) • Continue the GridWorld case study Answer questions during reding (GW Problem Sets 1-9) • Group Activity, groups of 3-5. (GW Part 3 and Part 4 Group Exercises) • Practice creating and modifying Actors

	• Understand data encapsulation and isolating object classes from each other using private methods and variables • Implement Interfaces for the GridWorld case study	in the GridWorld (GW Part 3 Exercises 1-6) • GridWorld Part 5 (time permitting)
17 (32-36)	**Review** **Topics:** • Review AP Computer Science A topics **Objectives:** • Prepare for the AP CS A Exam by reviewing material and taking practice exams	**Resource:** Previous free-response questions from AP Central **Assessment:** • Practice exams

Grades:
Grades will be determined by tests, assignments, and participation using the standard grade scale used by Washington County school district.

- Assignment may include research projects, written reports, slide-show presentations, programming projects, pop quizzes, etc.
- Tests will mainly be multiple choice and code interpretation in preparation for the end of level exam
- Participation is measured by time spent working (individually and as groups), reading, and studying.

Grading Scale:

A = 93 to 100%	B+ = 87 to 89%	B- = 80 to 82%	C = 73 to 76%
A- = 90 to 92%	B = 83 to 86%	C+ = 77 to 79%	C- = 70 to 72%

ALVEYWORLD Inc.:
I manage my class through my class website called Alveyworld (alveyworld.pineview.org).
- Classwork: All required assignments will be posted on Alveyworld on the class calendar.
- Submit: All assignments must be submitted by email to **alvey@pineview.org**.
- Grades: Login to Power School through Alveyworld and keep up with your grades.
- Blogs: All students will journal their experiences in Alveyworld by writing in an online blog.
- A list of supplies needed for my class are listed on the Alveyworld website

Homework:
If possible, assignments may be completed at home using a computer with Internet access, and a python interpreter. It is not required to have these resources at home, but it certainly helps. There will be reading assignments, book reports, and other things to be completed at home.

Absences: A large part of a student's grade is based on participation. Being absent for any reason will have a negative effect on student grades. Students will be responsible for any announcements made during class, so regular attendance is mandatory.

Classroom Rules:

1. Be Responsible
2. Show Respect
3. Follow teacher directions
4. Follow Pine View High School Rules (including dress-code)
5. Follow the Washington County School District Internet Access rules.
*. Please practice good hygiene habits such as regular hand washing, bathing, etc.
*. Get plenty of sleep, exercise, and eat healthy

If these rules are followed we will all learn new and exciting things and become better people.

Consequences if a student chooses to break the rules:
First time: Verbal warning
Second time: Email Parents
Third time: Loss of School Computer account
Forth time: Administrative referral
Severe disruption or safety violation will result in immediate referral to administration.
I want, above all, to help the students enjoy learning and to feel successful in doing so. I will do everything I can to help them. Please feel free to contact me by phone or email at any time.

Acceptable Use Policy
Mr. Alvey's Computer Lab 2010-2011

In addition to the rules and regulations put forth by the Washington County School District AUP, the computers and Internet access in my laboratory are to be used only for learning, study, research, and development of technology skills. The computers and Internet access are not for personal use. Personal use includes the following, unless specified in a specific assignment:

- *Playing, downloading, or storing games.*
- *Browsing the Internet for personal interests, etc.*
- *Visiting email, bulletin board, blog, message board, or chat sites*
- *Viewing, downloading, sharing, or storing videos*
- *Listening, downloading, sharing, or storing music*
- *Downloading pictures, programs, files, etc.*
- *Inappropriate, or offensive use of digital camera*

Violation of the acceptable use policy will result in loss of computer privileges and/or dismissal from the class. Inappropriate and/or abusive use of the computers and Internet access will not be tolerated and may result in state or federal punishment. Inappropriate and/or abusive uses include: Pornography, Gambling, Security Hacking, Piracy, Vandalism, Theft, etc.
The computers and Internet access available in my lab are powerful tools provided for learning and must be treated with respect and used with great responsibility.

- (cut here and return signatures)

I____(parent/guardian), have read and understand the above Disclosure and Acceptable Use Policy, including the associated Washington County acceptable use policy (AUP), and hereby give permission for my student to use the computers and Internet access provided in Mr. Alvey's laboratory, and will take responsibility for any misuse.

X____ ____________________
Parent/Guardian Signature Date

I____(student's full name), have read and understand the above Disclosure and Acceptable Use Policy and hereby promise to respect Mr. Alvey's laboratory, including the associated Washington County acceptable use policy (AUP), and will take full responsibility if I misuse the computers and Internet access.

X____ ____________________
Student Signature Date