

AP[®] Computer Science AB

Course Description

This course extends the concepts of AP[®] Computer Science A with an emphasis on object-oriented programming (OOP) and design. Students will analyze, design, modify and implement many of the advanced data structures used in computer science that are universal to most programming languages. Other topics include program design and implementation, algorithm analysis, and object-oriented programming design. AP[®] Computer Science emphasizes programming methodology with a focus on problem solving and algorithm development. It is intended to serve both as an introductory course for computer science majors and for students who will major at disciplines that require significant involvement with computing.

Computer Labs

The AP[®] Computer Science AB course is taught in a computer lab. Each student has a computer workstation. Lectures are interactive – many students take notes electronically and program sample java classes with the instructor. Most class periods are about one third lecture and two thirds lab or written work. Each class is a 100 minute block that meets twice a week.

Course Strategies

Topics are presented in an open class discussion/lecture environment using a tablet PC, power point presentations, and LCD projector. Following initial discussion, students analyze some java code by completing a sample program as a class. Each student types in their own program, but the coding is done together. To reinforce the topic, students complete written assignments that are then discussed in class.

After completion of these activities, students are then given a programming [lab] assignment. Assignments are posted on the course website. Each project requires the student to post their finished assignment to their online portfolio with a detailed project summary and links to their source code and javadocs. Most projects are completed as applets. Several require a link to a .jar file. Most projects are designed and implemented individually, however several larger assignments are completed using the Pair Programming methodology.

Programming projects are primarily completed during class. However, students are expected to be working [designing algorithms, writing code, etc.] on their projects outside of class.

Following completion of the programming project, students take a quiz over the topic. Quizzes consist of 3 – 4 multiple choice questions and 2-3 free response type questions. There is a short cumulative test after each unit consisting of multiple choice and free response type questions.

There is a Midterm exam and Final exam for each semester. These are cumulative and consist of multiple choice and free response type questions.

Textbook(s) and resources

Horstmann, Cay. Java Concepts – AP® Edition, 4th Ed. New York, Wiley, 2005.

Trees, Fran, Advanced Placement Computer Science Study Guide, 4th Ed. New York, Wiley, 2006.

GridWorld Case Study. The College Board, 2006.

Institute for Mathematics and Computer Science Online Courses (eIMACS) **www.eimacs.com**,
<http://www.eimacs.com/>

Armstrong, Stacey. *A+ Computer Science: Computer Science Curriculum Solutions*.

<http://apluscompsci.com>, 2006

Recommended References

Java Concepts Companion Website: <http://bcs.wiley.com/he->

[bcs/Books?action=index&itemId=0471697044&bcsId=2215](http://bcs.wiley.com/he-bcs/Books?action=index&itemId=0471697044&bcsId=2215)

AP® CS college board site: <http://www.collegeboard.com/ap/students/compsci/>

Sun's Java Programmers forum: <http://forum.java.sun.com/index.jspa>

AP®CS Java Concepts Study Guide WebSite: <http://www.ftrees.com/Study%20Guide.htm>

ICE@Georgia Tech: Institute for Computing Education online AP® Multiple Choice practice questions: http://manatee.cc.gt.atl.ga.us/apExam/begin_test.jsp

JavaBot, <http://www.javabat.com/>

AP® Central : Computer Science AB Quick Reference Guide, 2006

Student Outcomes:

After successfully completing this course, students will be able to:

- Design and implement computer-based solutions to problems in several application areas.
- Learn implement, and modify well-known algorithms and data structures.
- Develop and select appropriate algorithms and data structures to solve problems.
- Code fluently in a well-structured object oriented fashion using the JAVA programming language.
- Design and implement object oriented programs using integrated JAVA IDE such as Netbeans, Eclipse, Blue J, or JBuilder.
- Students are expected to be familiar with and be able to use standard JAVA AP®I (Application Programming Interface).
- Read and understand a large program and a description of the design and development process leading to such a program. (Examples of such programs are the Advanced Placement case studies.)
- Identify the major hardware and software components of a computer system, their relationship to one another, and the roles of these components within the system.
- Recognize the ethical and social implications of computer use.

General Topics	Week
Developing and Managing Online Portfolio	1
CS 1 Topic Review [AP®CS-A]	2-3
Two Dimensional Arrays	4-5
ArrayLists	6
Iterators	7
Comparable, Interfaces, and Abstract Classes	8
GridWorld Chapters 1 -4	9-10
Hashing, Maps and Sets	11-13
Recursion	14
Linked Lists	15-17
Stacks	18
Queues	19
Semester Exam	20
Trees, Binary Trees	21-24
Heaps, Priority Queues	24-25
Big O Notation, Advanced Searching and Sorting	26 - 27
GridWorld Case Study Chapter 5	28 - 30
AP® Exam review	31-33

<p>Week 1 Topic: Developing and Managing Online Portfolio</p> <ul style="list-style-type: none"> • Interactions of hardware and software components in the lab • User responsibility of computer • Fundamentals of html • Netbeans IDE 	<p>Objectives:</p> <ul style="list-style-type: none"> • Create a framed website using Dreamweaver • Create and edit a project in NetBeans • Successfully post a programming project
<p>Weeks 2 -3 Topic: CS 1 Topic Review [AP®CS – A]</p> <ul style="list-style-type: none"> • Object Oriented Design concepts: <ul style="list-style-type: none"> ○ Instance fields and methods ○ Object and references ○ Super classes and sub classes ○ Designing a class. • NetBeans IDE <ul style="list-style-type: none"> ○ Creating an applet using the 	<p>Objectives:</p> <ul style="list-style-type: none"> • Demonstrate previous programming knowledge and techniques. • Design and implement several classes

<p>form editor.</p> <ul style="list-style-type: none"> ○ Writing javaDocs ○ Compiling and running applet. ● Using the Java AP®I ● Using the Sun Java Developer’s Forum 	<p>Readings:</p> <ul style="list-style-type: none"> ● Java Concepts: selections from Chapters: 2-4, 6,7, and 9. <p>Guided Practice: <i>Topic discussion, Example program analysis and modification. A+ Computer Science assorted labs and worksheets.</i></p> <p>Programs:</p> <ul style="list-style-type: none"> ● Quadratic Formula: design and implement a class that models the quadratic formula using an applet. ● Pirate Talk: design and implement a class that translates ordinary text into pirate language. Use an applet in your design. <p>Assessment:</p> <ul style="list-style-type: none"> ● Assorted Quizzes ● Unit Test
<p>Weeks 4-5 Topic: Two Dimensional Arrays</p>	<p>Objectives:</p> <ul style="list-style-type: none"> ● Declare and instantiate 2-D array objects ● Use nested loops to manipulate 2-D arrays ● Design and implement methods using 2-D arrays <p>Readings:</p> <ul style="list-style-type: none"> ● Java Concepts: pp. 298 – 303 ● eIMACS: read section on 2-D arrays online <p>Guided Practice: <i>Topic discussion, Example program analysis and modification. A+ Computer Science assorted labs and worksheets.</i></p>

	<p>Programs:</p> <ul style="list-style-type: none">• Determinants and Matrices: Design a class that models an nxm matrix. Implement methods for operations: addition, subtraction and multiplication. Implement methods for determinant and inverse.• Game of Life <p>Assessment:</p> <ul style="list-style-type: none">• Assorted Quizzes• Unit test
<p>Week: 6 Topic: ArrayLists</p>	<p>Objectives:</p> <ul style="list-style-type: none">• Demonstrate previous programming knowledge of ArrayLists and methods.• Design and implement several classes using ArrayLists <p>Readings:</p> <ul style="list-style-type: none">• Java Concepts: pp. 298 – 303• eIMACS: read section on ArrayLists online <p>Guided Practice: <i>Topic discussion, Example program analysis and modification. A+ Computer Science assorted labs and worksheets.</i></p> <p>Program:</p> <ul style="list-style-type: none">• <i>TextAnalyzerApplet- using Scanner Class and ArrayLists:</i> Design and implement a TextAnalyzer class that counts the number of words pasted in a JTextArea, searches for a word submitted by the user, and counts the number of times that word appears in the text. Class must use an ArrayList to store each word from the JTextArea.

	<p>Assessment:</p> <ul style="list-style-type: none">• Assorted Quizzes• Unit test
<p>Weeks: 7 Topic: Iterators</p>	<p>Objectives:</p> <ul style="list-style-type: none">• Demonstrate knowledge of Iterators and List Iterators• Demonstrate knowledge about when to use iterators and why to use them• Demonstrate difference between an iterator and list iterator. <p>Readings:</p> <ul style="list-style-type: none">• Java Concepts: pp 742 – 747• eIMACS: complete sections Iterators and List Iterators online. <p>Guided Practice: <i>Topic discussion, Example program analysis and modification. A+ Computer Science assorted labs and worksheets.</i></p> <p>Programs:</p> <ul style="list-style-type: none">• RemoveIT: Read in a list of words and a word to remove from the list. Remove all occurrences of the word from the list. You must use an Iterator.• Replacelt: Read in a list of words, a word to replace, and a word with which to replace the original word. Replace all occurrences of the original word with the new word. You must use a ListIterator <p>Assessment:</p> <ul style="list-style-type: none">• Assorted Quizzes• Unit test

Week: 8

Topic: Interfaces, Abstract Classes, and Comparable

Objectives:

- Demonstrate how to implement an interface
- Demonstrate how to implement the Comparable Interface.
- Demonstrate how to combine interfaces, abstract classes and inheritance to write a multi-class project.

Readings:

- Java Concepts: pp 410 – 417
- eIMACS: complete section on Abstraction online

Guided Practice: *Topic discussion, Example program analysis and modification. A+ Computer Science assorted labs and worksheets.*

Programs:

- **Sort By Vowels:** Sort all words by comparing the number of vowels that each word contains. The word with the fewest number of vowels would come first. If you have more than one word with the same number of vowels, that group would be sorted alphabetically.
- **CompareColors:** Applet: Design a JApplet that will create, display and sort ten randomly colored ovals stored in an array. The CompareColors class implements the *Comparable* interface. Students provide implementation of compareTo method.

Assessment:

- Assorted Quizzes
- Unit test

<p>Weeks: 9-10 Topic: GridWorld Chapters 1-4</p>	<p>Objectives:</p> <ul style="list-style-type: none"> • Review Testing • Class modification • Demonstrate knowledge of inheritance in GridWorld Classes. <p>Readings:</p> <ul style="list-style-type: none"> • GridWorld Case Study, Parts 1 - 4: • eIMACS: read Introduction to Case Study online. <p>Guided Practice: <i>Topic discussion, Example program analysis and modification. A+ Computer Science assorted labs and worksheets.</i></p> <p>Programs:</p> <ul style="list-style-type: none"> • DiagonalBug • RockThrowingCritic • Student designed Bugs, Critters, and new Actor classes <p>Assessment:</p> <ul style="list-style-type: none"> • Assorted Quizzes • Unit test
<p>Weeks: 11 - 13 Topic: Hashing, Maps, and Sets</p>	<p>Objectives:</p> <ul style="list-style-type: none"> • Demonstrate features of Sets and successful implementation of a Set. • Demonstrate features of Maps and successful implementation of a Map. • Demonstrates understanding of the Set and Map interfaces. • Demonstrates understanding of key – value pairs • Demonstrates understanding and use of HashSet and TreeSet implementations of the Set Interface • Demonstrates understanding and use of HashMap and TreeMap

implementations of the Map Interface

- Demonstrates understanding of efficiencies and running times of HashSet, TreeSet, HashMap, and TreeMap

Readings:

- Java Concepts: pp 776 – 795
- eIMACS: read and complete section on Sets and Maps online.

Guided Practice: *Topic discussion, Example program analysis and modification. A+ Computer Science assorted labs and worksheets.*

Programs:

- **Union/Intersection/Difference:** Read in 2 Sets of numbers. Then, you are to perform the set operations of union, intersection, and difference on the 2 Sets.
- **Odds and Evens:** Read in a list of numbers and determine which ones are odd and which ones are even. You will add all of the even numbers to a set and all of the odd numbers to a different set.
- **Spanish To English:** Read in a list of Spanish words with English equivalents and use the pairs to write a basic low level translation program. The Spanish word will be the key and the English word will be the value.
- **Relatives:** Take list of people and to whom each person is related. Then, build a list for each person and all of the people related to that person. This program must be written using a Map of Sets. The Map will store the key (person) and a value (Set of the people related to that person).
- **Histogram:** Read in a list of characters/symbols and keep track of how many times each character/symbol occurs.
- **AutoParts:** Read in and organize a list of Parts. Parts are stored in PartList. PartList contains a Map of Parts.

	<p>Assessment:</p> <ul style="list-style-type: none"> • Assorted Quizzes • Unit test
<p>Week: 14 Topic: Recursion</p>	<p>Objectives:</p> <ul style="list-style-type: none"> • Demonstrate understanding of Base Case and Recursive steps. • Design and implement a recursive algorithm <p>Readings:</p> <ul style="list-style-type: none"> • Java Concepts: pp 663 - 686 <p>Guided Practice: <i>Topic discussion, Example program analysis and modification. A+ Computer Science assorted labs and worksheets.</i></p> <p>Programs:</p> <ul style="list-style-type: none"> • BubbleGame: Write a program to play a pattern matching game. All adjacent matching patterns are eliminated from the grid. Requires a recursive design. <p>Assessment:</p> <ul style="list-style-type: none"> • Assorted Quizzes • Unit test
<p>Weeks: 15 - 17 Topics: Linked Lists using ListNode class, Linked Lists using LinkedList class, Iterators with Lists</p>	<p>Objectives:</p> <ul style="list-style-type: none"> • Create an APLinkedList class using ListNode class • Demonstrate understanding of ListNodes and Linked List by implementing methods: addFront(), addLast(), removeFront(), removeLast(), toString(), etc. • Use LinkedList class to solve problems • Demonstrate understanding of Big O

	<p>analysis of Lists.</p> <p>Readings:</p> <ul style="list-style-type: none"> • Java Concepts: pp 741 – 757 • eIMACS: complete sections on LinkedLists, Iterators, and List Iterators • Advanced Placement Study Guide: pp 309 - 327 <p>Guided Practice: <i>Topic discussion, Example program analysis and modification. A+ Computer Science assorted labs and worksheets.</i></p> <p>Programs:</p> <ul style="list-style-type: none"> • Basic List: Write a program that demonstrates that your BasicList works as intended. BasicList is a single-linked linked list. Each node in BasicList has a data component and a reference to the next node only. Write and test all methods in BasicList. Complete methods to add, remove, get, and search. • Polynomials: Design and implement a Polynomial class that extends the Linked List class. Design methods for addition, subtraction, multiplication, and simplify. <p>Assessment:</p> <ul style="list-style-type: none"> • Assorted Quizzes • Unit test
<p>Week: 18 Topic: Stacks</p>	<p>Objectives:</p> <ul style="list-style-type: none"> • Demonstrate understanding of Stacks and LIFO • Demonstrate understanding of Stack methods. • Demonstrate understanding of Big O analysis of Stacks and methods.

	<p>Readings:</p> <ul style="list-style-type: none"> • Java Concepts: pp 762 – 774 • eIMACS: complete sections on Stacks online <p>Guided Practice: <i>Topic discussion, Example program analysis and modification. A+ Computer Science assorted labs and worksheets.</i></p> <p>Programs:</p> <ul style="list-style-type: none"> • Stack Basics: Take a string of tokens and put each token in the stack. Then, print the contents of the stack on the screen. Finally, pop each item from the stack one by one until the stack is empty. • Make a Stack Class: You are to create a stack from scratch using either an array or an ArrayList as the storage for the stack. • Balanced: Design an Applet that determines if a sequence of parenthesis is balanced. Must use the Stack class. <p>Assessment:</p> <ul style="list-style-type: none"> • Assorted Quizzes • Unit test
<p>Week: 19 Topics: Queues, Queue Interface using a LinkedList Class as implementation</p>	<p>Objectives:</p> <ul style="list-style-type: none"> • Demonstrate understanding of a queues and FIFO • Demonstrate understanding of Queue methods. • Demonstrate understanding of Big O analysis of Queues and methods <p>Readings:</p> <ul style="list-style-type: none"> • Java Concepts: pp 762 – 774 • eIMACS: complete sections on Queues online

	<p>Guided Practice: <i>Topic discussion, Example program analysis and modification. A+ Computer Science assorted labs and worksheets.</i></p> <p>Programs:</p> <ul style="list-style-type: none"> • Palindromes: Read in a list of Strings. Store the Strings in a Queue and in a Stack. Use the Stack and Queue to determine if the list of Strings is a palindrome. • Build a Queue Class: You are to create a queue from scratch using the ListNode class as the storage for the queue. <p>Assessment:</p> <ul style="list-style-type: none"> • Assorted Quizzes • Unit test
<p>Week: 20 Topics: Semester01 Exam</p> <p>End of Semester One</p>	<p>Objectives:</p> <ul style="list-style-type: none"> • Demonstrate understanding of semester one topics and concepts <p>Assessment:</p> <ul style="list-style-type: none"> • Semester Exam: Multiple Choice and Free Response
<p>Weeks: 21-24 Topic: Trees</p>	<p>Objectives:</p> <ul style="list-style-type: none"> • Demonstrate understanding of tree terminology. • Demonstrate understanding of a Binary Search Trees and implementation • Create a BST class using the TreeNode Class. • Demonstrate understanding of recursive helper methods. • Distinguish between general trees, binary trees, binary search trees, and heaps • Demonstrate understanding of Big O

analysis of Tree methods.

Readings:

- Java Concepts: pp796 – 810
- eIMACS: complete sections on Trees and Heaps
- Advanced Placement Study Guide: pp 359 - 366

Guided Practice: *Topic discussion, Example program analysis and modification. A+ Computer Science assorted labs and worksheets.*

Programs:

- **Basic Tree Class:** Complete BasicTree class using TreeNode class. Include as many of the standard binary search methods as possible.
- **Histogram Tree:** Write a program that uses nodes to store letters and letter counts. This program is similar to a Map. In each node, you will store a Comparable reference, a count of how many of that Comparable has occurred, and references to the node's left and right nodes.
- **Anagrams:** In this program you will get a word from the user, and search a given dictionary to list all possible anagrams of the user entry. If there are no corresponding anagrams, an appropriate message is displayed. The algorithm for this problem is to take every dictionary word, convert it into a Word object and insert it into a binary tree. The dictionary is a binary tree of all the sorted representations of the words in the dictionary. The node for each element in the binary tree will have its sorted representation, and a list all words in the dictionary that have that sorted representation, i.e. anagrams.

Assessment:

- Assorted Quizzes
- Unit test

<p>Weeks: 25 Topics: Heaps and PriorityQueues</p>	<p>Objectives:</p> <ul style="list-style-type: none"> • Demonstrate understanding of how to add to, delete from, search, and perform all types of manipulations on a Heap and PriorityQueue • Demonstrate understanding of difference between a Queue and a PriorityQueue. • Perform operations on PriorityQueues: traversals, insertions, and deletions. <p>Readings:</p> <ul style="list-style-type: none"> • Java Concepts: pp815 – 832 • eIMACS: complete sections on Heaps and PriorityQueues. • Advanced Placement Study Guide: pp 372 - 378 <p>Guided Practice: <i>Topic discussion, Example program analysis and modification. A+ Computer Science assorted labs and worksheets.</i></p> <p>Programs:</p> <ul style="list-style-type: none"> • Basic Heap: Write a heap program – must be a binary tree implemented with an array. • Basic PriorityQueue: Must use a minimum heap. <p>Assessment:</p> <ul style="list-style-type: none"> • Assorted Quizzes • Unit test
<p>Week: 26 -27 Topics: Big O Notation, Advanced Searching and Sorting</p>	<p>Objectives:</p> <ul style="list-style-type: none"> • Demonstrate understanding of complexity analysis of an algorithm using Big O Notation. • Recognize typical Big O functions and be able to order them by increasing rate. • Demonstrate understanding of the

	<p>following sorting algorithms: Merge, Selection, Heap, Quick, and Insertion</p> <ul style="list-style-type: none"> • Demonstrate understanding of the following searching algorithms: Sequential and Binary • Demonstrate understanding of the Big O analysis for the searching and sorting algorithms by worst case and average case <p>Readings:</p> <ul style="list-style-type: none"> • Java Concepts: pp 704 - 740 • eIMACS: complete sections on Program Analysis and Searching and Sorting <p>Guided Practice: <i>Topic discussion, Example program analysis and modification. A+ Computer Science assorted labs and worksheets.</i></p> <p>Programs:</p> <ul style="list-style-type: none"> • Run sorting routines for different-sized data sets. • Create a short presentation comparing and contrasting sorting routines. <p>Assessment:</p> <ul style="list-style-type: none"> • Assorted Quizzes • Unit test
<p>Weeks: 28 – 30 Topic: GridWorld Case Study Chapter 5</p>	<p>Objectives:</p> <ul style="list-style-type: none"> • Demonstrate understanding of the BoundedGrid implementation. • Consider alternate implementations for the BoundedGrid class. • Demonstrate understanding of the UnboundedGrid implementation. • Consider and code alternate implementation for the UnboundedGrid class.

	<p>Readings:</p> <ul style="list-style-type: none"> • Chapter 5 of the GridWorld Case Study <p>Guided Practice: <i>Topic discussion, Example program analysis and modification, Worksheets</i></p> <p>Programs:</p> <ul style="list-style-type: none"> • Selected exercises from Part 5 of the GridWorld Case Study <p>Assessment:</p> <ul style="list-style-type: none"> • Assorted Quizzes • Unit test
<p>Weeks: 31 - 33 Topic: AP[®] Exam Review</p>	<p>Objectives:</p> <ul style="list-style-type: none"> • Demonstrate understanding of the scoring guidelines of the free response section • Discuss and code a variety of free response questions • Discuss strategies for taking the multiple choice section • Discuss and practice multiple choice questions. <p>Readings:</p> <ul style="list-style-type: none"> • ICE@Georgia Tech: Institute for Computing Education online AP[®] Multiple Choice practice questions.

Grading Scale

90-100	A
80 -89	B
70 – 79	C
60 – 69	D
59↓	F

Grading Weights

Programs	30%
Quizzes & Tests	30%
Portfolio	10%
Journals/Class Participation	05%
Midterm	10%
Final Exam	15%

Academic Honesty

Your work on exams is expected to be entirely your own; failure to adhere to this may result in an F for the course. As for written homework assignments, talking to others about confusing points is permitted and, to some extent, encouraged. However, it is expected that all actual solution writing, project typing, and project design be done by you. In particular, **the sharing of your project source files is never acceptable**. Inappropriate collaboration may result in your receiving a zero for the homework or project portions of the course.

Late Assignment Policy

No late assignments will be accepted. However, you are allowed three grace days per semester to turn in late assignments. You may use these individually or in combination.

Expectations

- All student computer activity will be subject to the NCPHS Acceptable Use Policy guidelines.
- All student work will be subject to the NCPHS Honesty and Ethics Policy guidelines.
- Students will work only at their assigned stations. If the assigned station is not functioning, please notify the teacher immediately to be re-assigned to another station.
- The class will wait to be dismissed by the teacher. The music is only the reminder to begin to bring the class activities to an end.
- There will always be adequate time to complete lab assignments in class. However, it is expected that students will utilize the time in class well, in order to complete each assignment as well as possible.
- Students are expected to behave in a businesslike manner while in the computer lab, by being on time at their workstation when class is to start, and by discussing school related matters only. **The playing of games is not allowed**. E-mail, chat lines, the Internet etc. are to be used only with permission and for appropriate purposes.

Computer Usage and Etiquette

Never touch another computer's keyboard, mouse etc. without permission. While we encourage students to collaborate and help others who are confused about something, if another student needs assistance, please explain what is needed and guide the student. Everyone learns how to do something by actually doing it on his or her own.

At the end of class, log off, clear the work area, and return the chair to its original upright position and push it against the table.

Food Policy

This one is simple... NO eating or drinking in the labs..... **PERIOD**. Please finish all beverages in the hallway before entering class. Keep food items and beverages in your bookbag during class.

Contact

Don Yanek

dyanek@northsideprep.org

773-534-3954 e26834