

AP[®] Computer Science A Syllabus

Course Resources

- Horstmann, Cay S. *Java Concepts (Advanced Placement Version)*. 5th Edition. Hoboken, NJ: Wiley, 2008. ISBN: 978-0-470-18160-7.
- Trees, Frances P., and Horstmann, Cay S. *Java Concepts: Advanced Placement Computer Science Study Guide*. 5th Edition. Hoboken, NJ: Wiley, 2008. ISBN: 978-0-470-18161-4
- *GridWorld Case Study*, The College Board, 2006
- Web site: WileyPLUS Subscription to accompany Java Concepts 5/E for Java 5 and 6
- *AP Computer Science Course Description, May 2007, May 2008*, The College Board, 2006
- *2004 AP Computer Science A Exam*, The College Board, 2005
- Software: Sun Microsystems Java Standard Edition Development Kit
- Software: Helios Software Solutions TextPad[®] 5.0
- Software: Alice 2.0
- Web site: Author's web site: <http://horstmann.com/>
- Web site: Publisher's Student Companion Site: <http://bcs.wiley.com/he-bcs/Books?action=index&itemId=0471697044&bcsId=2215>
- Web site: Instructor's web site: <http://www.xxxxapcs.org/>
- Web site: Java™ Platform, Standard Edition, API Specification: <http://java.sun.com/javase/6/docs/api/>
- Web site: Java™ Documentation: <http://java.sun.com/javase/6/docs/>
- Web site: College Board AP Student Web Site: http://www.collegeboard.com/student/testing/ap/sub_compscica.html?compscica
- Web site: ACM Code of Ethics: <http://www.acm.org/constitution/code.html>

Units of Study

Unit 1: Introduction to AP Computer Science

Unit Goals:

- To learn about the AP Program
- To learn the difference between AP Computer Science A and AB
- To become familiar with resources available on the College Board AP Student website
- To become familiar with Alice
- To write modify an existing program in Alice
- Students work through part one of the GridWorld Case Study, performing the GridWorld experiments and observing the attributes and behavior of the actors
- To create a new program using Alice
- To become familiar with the ACM Code of Ethics
- To discuss, as a class, the ethical and social implications of computers and how these issues relate to the school's honor code
- Each student will select a case from the EFF legal case archives and present arguments from both sides of the case to the class from one of the following topics: System reliability, privacy, piracy and intellectual property rights, social and ethical ramifications of computing

Resources:

- Software: Alice 2.0
- Horstmann: Preface, page vii
- *AP Computer Science Course Description, May 2007, May 2008*, The College Board, 2006
- *GridWorld Case Study*, The College Board, 2006
- Web site: College Board AP Student Web Site:
http://www.collegeboard.com/student/testing/ap/sub_compscica.html?compscica
- Web site: ACM Code of Ethics:
<http://www.acm.org/constitution/code.html>
- Web site: Electronic Frontier Foundation:
<http://www.eff.org/>

Unit 2: Hardware and Software

Unit Goals:

- To understand the activity of programming
- To learn about the architecture of computers
- To learn about machine code and high-level programming languages
- To become familiar with your computing environment and your compiler
- To learn how to install Java and TextPad on a home computer
- To compile and run a Java program
- To recognize syntax and logic errors
- To read and discuss "The ENIAC"

Topics:

- What is programming?
- Anatomy of a computer
- Translating human-readable programs into machine code
- The Java programming language
- Writing and compiling a simple program
- Errors
- The compilation process

Language Features:

- Method call
- `System.out.println` and `System.out.print`

Resources:

- Horstmann: Chapter 1. Introduction, page 1
- Horstmann: Appendix A. Java Language Coding Guidelines
- Lab: Becoming Familiar with your Computer
- Software: Sun Microsystems Java Standard Edition Development Kit
- Software: Helios Software Solutions TextPad® 5.0

Unit 3: Using Objects

Unit Goals:

- To learn about variables
- To understand the concepts of classes and objects
- To be able to call methods
- To learn about parameters and return values
- To be able to browse the API documentation
- To realize the difference between objects and object references
- To read and discuss "Mainframes"

Topics:

- Types and variables
- The assignment operator
- Objects, classes and methods
- Method parameters and return values
- Number types
- Constructing objects
- Accessor and mutator methods
- Implementing a test program
- API documentation
- Object references

Language Features:

- Variable definition
- Assignment
- Object construction
- Importing a class from a package
- The primitive types: int and double
- class java.lang.String implements java.lang.Comparable
 - int length()

Resources:

- Horstmann: Chapter 2. Using Objects, page 33
- PowerPoint Slides
- Lab: Objects, Classes and Methods

Unit 4: Implementing Classes

Unit Goals:

- To become familiar with the process of implementing classes
- To be able to implement simple methods
- To understand the purpose and use of constructors
- To understand how to access instance fields, local variables, and parameter variables
- To appreciate the importance of documentation comments
- To read and discuss "Electronic Voting Machines"

Topics:

- Levels of abstraction
- Specifying the public interface of a class
- Commenting the public interface
- Instance Fields
- Implementing constructors and methods
- Unit testing
- Categories of variables
- Implicit and explicit method parameters

Language Features:

- Method definition
- Constructor definition
- Class definition
- Instance field declaration
- The return statement

Resources:

- Horstmann: Chapter 3. Implementing Classes, page 75
- PowerPoint Slides
- Lab: Designing the Public Interface of a Class

Unit 5: Fundamental Data Types

Unit Goals:

- To understand integer and floating-point numbers
- To recognize the limitations of the numeric types
- To become aware of causes for overflow and round-off errors
- To understand the proper use of constants
- To write arithmetic expressions in Java
- To use the String type to define and manipulate character strings
- To learn how to read program input and produce formatted output
- To read and discuss "The Pentium Floating-Point Bug"
- To read and discuss "International Alphabets"

Topics:

- Number types
- Constants
- Assignment, increment, and decrement
- Arithmetic operations and mathematical functions
- Calling static methods
- Strings
- Reading input

Language Features:

- Numeric casts: (int) and (double)
- Constant definition
- Static method call
- Primitive types: int, double, and boolean
- Assignment operator: =
- Increment and decrement operators: ++ and --
- Combined arithmetic/assignment operators: +=, -=, *=, /=, and %=
- Arithmetic operators: +, -, *, /, and %
- String concatenation operator: +
- Escape sequences: \\, \", and \n
- class java.lang.Math
 - static int abs(int x)
 - static double abs(double x)
 - static double pow(double base, double exponent)
 - static double sqrt(double x)
 - static double random()
- class java.lang.String implements java.lang.Comparable
 - int length()
 - String substring(int from, int to)
 - String substring(int from)
 - int indexOf(String str)

Resources:

- Horstmann: Chapter 4. Fundamental Data Types, page 123
- Handout: Methods of the String class: length, substring, and indexOf
- Handout: Methods of the Math class: abs, pow, sqrt, and random
- Horstmann: Appendix L. Number Systems
- PowerPoint Slides
- Lab: Using Numbers
- Video: The Story of 1

Unit 6: Decisions

Unit Goals:

- To be able to implement decisions using if statements
- To understand how to group statements into blocks
- To learn how to compare integers, floating-point numbers, strings, and objects
- To recognize the correct ordering of decisions in multiple branches
- To program conditions using Boolean operators and variables
- To read and discuss "Artificial Intelligence"

Topics:

- The if statement
- Comparing values
- Multiple alternatives
- Using boolean expressions
- Test coverage

Language Features:

- The if statement
- Block statement
- Relational operators: ==, !=, <, <=, >, and >=
- Logical operators: &&, ||, and !
- "Short circuit" evaluation of Boolean operators
- class java.lang.Object
 - boolean equals(Object other)
- interface java.lang.Comparable
 - int compareTo(Object other)
- class java.lang.String implements java.lang.Comparable
 - boolean equals(String other)
 - int compareTo(String other)

Resources:

- Horstmann: Chapter 5. Decisions, page 163
- PowerPoint Slides
- Lab: The if Statement

Unit 7: Recursion

Unit Goals:

- To learn about the method of recursion
- To understand the relationship between recursion and iteration
- To analyze problems that are much easier to solve by recursion than by iteration
- To learn to “think recursively”
- To be able to use recursive helper methods
- To understand when the use of recursion affects the efficiency of an algorithm
- To read and discuss "The Limits of Computation"

Topics:

- Triangle numbers
- Permutations
- Recursive helper methods
- The efficiency of recursion
- Mutual recursions

Language Features:

- Recursive methods

Resources:

- Horstmann: Chapter 13. Recursion, page 511
- PowerPoint Slides
- Lab: Recursion

Unit 8: Iteration

Unit Goals:

- To be able to program loops with the while, for, and do statements
- To avoid infinite loops and off-by-one errors
- To understand nested loops
- To learn how to process input
- To implement simulations
- To read and discuss "Spaghetti Code"
- To read and discuss "Correctness Proofs"

Topics:

- while loops
- for loops
- Nested loops
- Processing sentinel values
- Random numbers and simulations
- Using a debugger
- A sample debugging session

Language Features:

- The while statement
- The for statement

Resources:

- Horstmann: Chapter 6. Iteration, page 203
- PowerPoint Slides
- Lab: Simple Loops

Unit 9: Arrays and Array Lists

Unit Goals:

- To become familiar with using arrays and array lists
- To learn about wrapper classes, auto-boxing, and the enhanced for loop
- To study common array algorithms
- To learn how to use two-dimensional arrays
- To understand when to choose array lists and arrays in your programs
- To implement partially filled arrays
- To read and discuss "An Early Internet Worm"

Topics:

- Arrays
- Array lists
- Wrappers and auto-boxing
- The enhanced for loop
- Simple array algorithms
- Two-dimensional arrays
- Copying arrays
- Regression testing

Language Features:

- Array construction
- Array element access
- The for each loop
- class java.util.ArrayList
 - int size()
 - boolean add(E obj)
 - void add(int index, E obj)
 - E get(int index)
 - E set(int index, E obj)
 - E remove(int index)
- class java.lang.Integer implements java.lang.Comparable
 - Integer(int value)
 - int intValue()
- class java.lang.Double implements java.lang.Comparable
 - Double(double value)
 - double doubleValue()

Resources:

- Horstmann: Chapter 7. Arrays and Array Lists, page 249
- PowerPoint Slides
- Lab: Using Arrays

Unit 10: Designing Classes

Unit Goals:

- To learn how to choose appropriate classes to implement
- To understand the concepts of cohesion and coupling
- To minimize the use of side effects
- To document the responsibilities of methods and their callers with preconditions and postconditions
- To understand the difference between instance methods and static methods
- To introduce the concept of static fields
- To understand the scope rules for local variables and instance fields
- To learn about packages
- To read and discuss "The Explosive Growth of Personal Computers"

Topics:

- Choosing classes
- Cohesion and coupling
- Accessors, mutators, and immutable classes
- Side effects
- Preconditions and post conditions
- Static methods
- Static fields
- Scope
- Packages
- Unit test frameworks

Language Features:

-
- `class java.lang.Math`
 - `static double sqrt(double x)`

Resources:

- Horstmann: Chapter 8. Designing Classes, page 293
- PowerPoint Slides
- Lab: Choosing Classes

Unit 11: Interfaces and Polymorphism

Unit Goals:

- To learn about interfaces
- To be able to convert between class and interface references
- To understand the concept of polymorphism
- To appreciate how interfaces can be used to decouple classes
- To learn how to implement helper classes as inner classes
- To understand how inner classes access variables from the surrounding scope
- To implement event listeners for timer events
- To read and discuss "Operating Systems"

Topics:

- Using interfaces for code reuse
- Converting between class and interface types
- Polymorphism
- Using interfaces for callbacks
- Inner classes

Resources:

- Horstmann: Chapter 9. Interfaces and Polymorphism, page 337
- PowerPoint Slides
- Lab: Implementing an Interface

Unit 12: Inheritance

Unit Goals:

- To learn about inheritance
- To understand how to inherit and override superclass methods
- To be able to invoke superclass constructors
- To learn about protected and package access control
- To understand the common superclass Object and how to override its toString and equals methods
- To read and discuss "Scripting Languages"

Topics:

- Inheritance
- Inheritance hierarchies
- Inheriting instance fields and methods
- subclass construction
- Converting between subclass and superclass types
- Polymorphism
- Access control
- Object: The cosmic superclass

Language Features:

- class java.lang.Object
 - boolean equals(Object other)
 - String toString()

Resources:

- Horstmann: Chapter 10. Inheritance, page 377
- PowerPoint Slides
- Lab: Inheritance

Unit 13: Input/Output and Exception Handling

Unit Goals:

- To learn how to throw exceptions
- To be able to design your own exception classes
- To understand the difference between checked and unchecked exceptions
- To learn how to catch exceptions
- To know when and where to catch an exception
- To read and discuss "The Ariane Rocket Incident"

Topics:

- Reading and writing text files
- Throwing exceptions
- Checked and unchecked exceptions
- Catching exceptions
- The finally clause
- Designing your own exception types

Resources:

- Horstmann: Chapter 11. Exception Handling, page 429
- PowerPoint Slides
- Lab: Throwing Exceptions

Unit 14: Object-Oriented Design

Unit Goals:

- To learn about the software life cycle
- To learn how to discover new classes and methods
- To understand the use of CRC cards for class discovery
- To be able to identify inheritance, aggregation, and dependency relationships between classes
- To master the use of UML class diagrams to describe class relationships
- To learn how to use object-oriented design to build complex programs
- To read and discuss "Programmer Productivity"
- To read and discuss "Software Development - Art or Science"

Topics:

- The software life cycle
- Discovering classes
- Relationships between classes

Resources:

- Horstmann: Chapter 12. Object-Oriented Design, page 457
- PowerPoint Slides
- Lab: Discovering Classes

Unit 15: Sorting and Searching

Unit Goals:

- To study several sorting and searching algorithms
- To appreciate that algorithms for the same task can differ widely in performance
- To understand the big-Oh notation
- To learn how to estimate and compare the performance of algorithms
- To learn how to measure the running time of a program
- To read and discuss "The First Programmer"

Topics:

- Selection sort
- Profiling the selection sort algorithm
- Analyzing the performance of the selection sort algorithm
- Merge sort
- Analyzing the performance of the merge sort algorithm
- Searching
- Binary search
- Sorting real data

Language Features:

- `interface java.lang.Comparable`
 - `int compareTo(Object other)`

Resources:

- Horstmann: Chapter 14. Sorting and Searching, page 549
- PowerPoint Slides
- Lab: Sorting

Unit 16: GridWorld: The AP Exam Case Study

Unit Goals:

- Students work through parts two, three and four of the GridWorld Case Study
- Perform the GridWorld experiments and observe the attributes and behavior of the actors
- Become familiar with the Bug and Runner classes
- Become familiar with the GridWorld class hierarchy
- Learn to extend GridWorld classes
- Learn to write programs using the GridWorld classes

Resources:

- GridWorld Case Study
- Trees: The AP Exam Case Study

Unit 17: Review for AP Exam

Unit Goals:

- To review the topics that will be covered on the AP Exam
- To review the Java language features that will be covered on the AP Exam
- To review the reference materials that will be provided with the AP Exam
- To understand strategies for taking the AP Exam
- To take the 2004 Released Exam for practice
- To take the 2005-2007 Free-Response Questions for practice
- To examine the 2004-2007 Scoring Guidelines

Language Features:

- All language features are reviewed in this unit

Resources:

- College Board: AP Topic Outline
- College Board: AP Java Subset
- Trees: Strategies for Taking the AP CS Exam
- Trees: Cumulative Review 1
- Trees: Cumulative Review 2
- Trees: Cumulative Review 3
- College Board: 2004 AP Computer Science A Released Exam
- College Board: 2005 Free-Response Questions
- College Board: 2006 Free-Response Questions
- College Board: 2007 Free-Response Questions
- College Board: 2004 Scoring Guidelines
- College Board: 2005 Scoring Guidelines
- College Board: 2006 Scoring Guidelines
- College Board: 2007 Scoring Guidelines

Curricular Requirements

This teacher hereby certifies that he/she has read the most recent AP Computer Science Course Description, available as a free download on the AP Computer Science A Course Home Page. This teacher further certifies that this course includes all of the topics listed in the "Computer Science A" column of the Topic Outline in the AP Computer Science Course Description.

	Curricular Requirement	Unit(s)
C3	The course teaches students to design and implement computer-based solutions to problems in a variety of application areas. In addition to completing the lab exercise associated with each unit, students are required to submit one major programming project each 9-week grading period. Each major project focuses on a different application area.	All
C4	The course teaches students to use and implement commonly used algorithms and data structures.	9, 15
C5	The course teaches students to develop and select appropriate algorithms and data structures to solve problems.	2-17
C6	The course teaches students to code fluently in an object-oriented paradigm using the programming language Java. The course teaches students to use standard Java library classes from the AP Java subset delineated in Appendixes A and B of the AP Computer Science Course Description.	2-17
C7	The course teaches students to read and understand a large program consisting of several classes and interacting objects, and enables students to read and understand the current AP Computer Science Case Study posted on AP Central.	16
C8	The course teaches students to 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.	2
C9	The course teaches students to recognize the ethical and social implications of computer use.	1

Correlation to AP Topic Outline

This teacher hereby certifies that this course includes all of the topics listed in the "Computer Science A" column of the Topic Outline in the AP Computer Science Course Description.

I. Object-Oriented Program Design	
The overall goal for designing a piece of software (a computer program) is to correctly solve the given problem. At the same time, this goal should encompass specifying and designing a program that is understandable, can be adapted to changing circumstances, and has the potential to be reused in whole or in part. The design process needs to be based on a thorough understanding of the problem to be solved.	
	Unit(s)
A. Program design	
1. Read and understand a problem description, purpose, and goals.	1-4
2. Apply data abstraction and encapsulation.	3, 4
3. Read and understand class specifications and relationships among the classes (“is-a,” “has-a” relationships).	11, 14
4. Understand and implement a given class hierarchy.	10, 11
5. Identify reusable components from existing code using classes and class libraries.	3
B. Class design	
1. Design and implement a class.	10
3. Choose appropriate data representation and algorithms.	10
4. Apply functional decomposition.	10
5. Extend a given class using inheritance.	11, 12

II. Program Implementation

The overall goals of program implementation parallel those of program design. Classes that fill common needs should be built so that they can be reused easily in other programs. Object-oriented design is an important part of program implementation.

	Unit(s)
A. Implementation techniques	
1. Methodology	
a. Object-oriented development	3, 4, 10, 14
b. Top-down development	3, 4
c. Encapsulation and information hiding	3, 4
d. Procedural abstraction	3, 4
B. Programming constructs	
1. Primitive types vs. objects	3
2. Declaration	
a. Constant declarations	5
b. Variable declarations	3
c. Class declarations	4
d. Interface declarations	11
e. Method declarations	4
f. Parameter declarations	4
3. Console output (System.out.print/println)	2, 3, 5
4. Control	
a. Methods	3
b. Sequential	3
c. Conditional	6
d. Iteration	8
e. Recursion	7
C. Java library classes (included in the A-level (AP Java Subset))	3

III. Program Analysis

The analysis of programs includes examining and testing programs to determine whether they correctly meet their specifications. It also includes the analysis of programs or algorithms in order to understand their time and space requirements when applied to different data sets.

	Unit(s)
A. Testing	
1. Test classes and libraries in isolation.	3, 4
2. Identify boundary cases and generate appropriate test data.	6
3. Perform integration testing.	10
B. Debugging	
1. Categorize errors: compile-time, run-time, logic.	2
2. Identify and correct errors.	3
3. Employ techniques such as using a debugger, adding extra output statements, or hand-tracing code.	8
C. Understand and modify existing code	3
D. Extend existing code using inheritance	12
E. Understand error handling	
1. Understand runtime exceptions.	13
F. Reason about programs	
1. Pre- and post-conditions	10
2. Assertions	10
G. Analysis of algorithms	
1. Informal comparisons of running times	15
2. Exact calculation of statement execution counts	15
H. Numerical representations and limits	
1. Representations of numbers in different bases	5
2. Limitations of finite representations (e.g., integer bounds, imprecision of floating-point representations, and round-off error)	5

IV. Standard Data Structures

Data structures are used to represent information within a program. Abstraction is an important theme in the development and application of data structures.

	Unit
A. Simple data types (int, boolean, double)	5
B. Classes	10
C. One-dimensional arrays	9

V. Standard Algorithms

Standard algorithms serve as examples of good solutions to standard problems. Many are intertwined with standard data structures. These algorithms provide examples for analysis of program efficiency.

	Unit(s)
A. Operations on A-level data structures previously listed	
1. Traversals	9
2. Insertions	9
3. Deletions	9
B. Searching	
1. Sequential	15
2. Binary	15
C. Sorting	
1. Selection	15
2. Insertion	15
3. Mergesort	15

VI. Computing in Context

A working knowledge of the major hardware and software components of computer systems is necessary for the study of computer science, as is the awareness of the ethical and social implications of computing systems. These topics need not be covered in detail but should be considered throughout the course.

	Unit(s)
A. Major hardware components	
1. Primary and secondary memory	2
2. Processors	2
3. Peripherals	2
B. System software	
1. Language translators/compiler	2
2. Virtual machines	2
3. Operating systems	2
C. Types of systems	
1. Single-user systems	2
2. Networks	2
D. Responsible use of computer systems	
1. System reliability	1
2. Privacy	1
3. Legal issues and intellectual property	1
4. Social and ethical ramifications of computer use	1