June 3, 2014

AP Computer Science A and UIL Textbook

By: Brenden Smith

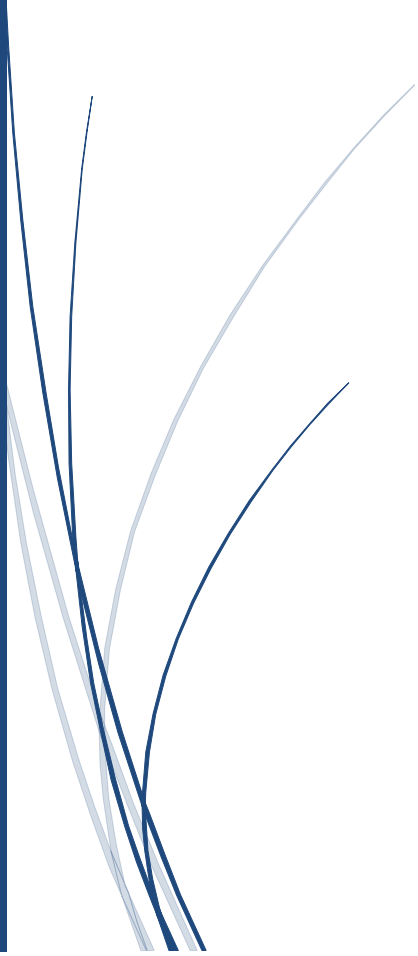

Table of Contents

Anything marked with a † is not covered on the AP exam but is on the UIL Computer Science exam.

1) Introduction

To start out, let's make sure this book is for you. This book is intended to teach the AP A Computer Science course; however, it can be used as a general learning tool for new or novice programmers. The programming language that will be used is Oracle's Java (because that's what the AP exam requires.) This book will make only a few assumptions, such as:

You have a computer to use while reading this book, so that you can run and test programs.

You have general knowledge about computer (I.E. you can open documents, you can browse your computer's hard drive, etc.)

Finally, to be cliché, you must have an eagerness to learn! If you don't want to learn this material will be the most tedious and boring thing you've ever read.

Many of you reading this textbook will have a false understanding of what computer science is. You probably are thinking that computer science is equitable with programming: you're incorrect. I'm sorry. Computer science is formally defined as the scientific and practical approach to computing and its applications. That may sound fairly complicated, but it won't be if it's translated out of computer vernacular. What the formal definition is trying to say is computer science is the science of computers: obvious, eh? That means that computer science includes more than just programming. It includes the abstract ideas behind computer's themselves (e.g. a Turing machine,) design of computer hardware, design of computer software, creation of computer software (this is where programming comes in,) and much, much more.

Over the course of this book, you will gain a fairly robust understanding of computer science, you will be able to implement that understanding in Java programs,, and you will have a great foundation to go out and learn anything about computers and computer science.

When I was in high school, I took the same course you are in, and it was the single best decision of my life. This single class has sent me down a life path that includes computers and computer science at every turn. I have a B.S. in Computer Science, and I don't believe I would have ever considered my educational and career path without my high school AP Computer Science class.

I apologize for boring you with my musings on my past, but I do have a point (two, actually.) The first point of reminiscing is to show you that this isn't just a class to put on a resume, it isn't just a class to fill time, it's a class that could change your life entirely, and I hope that you end up with the same passion for computers that I gained. The second point is to introduce a concept: the roller coaster effect of learning a programming language and other computer science topics.

The book that was used in my AP Computer Science class was written by a man named Mr. Leon Schram – and that is where the roller coaster concept was introduced to me. The roller coaster concept refers to

the emotional roller coaster that you WILL go through while in this class. When a new concept is introduced, you will not get it right away. This will make you angry or upset: this is the low point of the roller coaster. As you start to understand the concept (and climb up the hill to the next drop) you will start feeling proud of yourself. Then, there will be this moment of triumph: a light bulb moment, if you will. You will never get a better sense of satisfaction than when your program compiles, runs, and works flawlessly. It doesn't last long, though. The next concept will be introduced, and your roller coaster will plummet down that hill again. Don't worry, though, you'll get back up there, again. This concept is introduced to you here so that when you get to those lows you don't think that you're stupid: you're not. Computer science concepts are difficult to grasp, at times. So, hunker down and get through the lows and you'll get back to that high.

If you haven't noticed, this book is written very informally. This is intentional. I want you to feel like I am standing in front of you and teaching you like an instructor would. I hope to have the foresight to answer any questions you may have (when I fail at this, please ask your instructor – you can never ask enough questions when it comes to computers.) And I want you to enjoy the learning process this book presents.

Before we get to the meat of the book, I have one last comment. This course will be very fast paced: even more so, now, because what use to be two classes has been condensed into one. Because of this, I will explain the abstract concepts by and while using Java code. This means that some things won't make sense at all at first, and they won't be explained until later. Someone interested in computers is generally inquisitive, but sometimes throughout this book you're just going to have to take things for granted. However, I promise these things WILL be explained later in detail, and will make sense then. So, if something doesn't make sense, and I ask you to just do it: just do it.

I'd like to make one other quick note: read the book thoroughly (it is said it takes three thorough readings to learn information from text, take note of this) even if you don't want to; the tests and quizzes will be difficult if you don't (very much on purpose.

2) An introduction to programming

a) History

The history of computer science if very ambiguous and tends to lead to other people with dissenting views than you've been taught or have known. The one thing everyone agrees on is that computers were designed to do the job of a computer (a computer pre-electronic computers was a person who did computation work by hand.)

Before I give you the lesson on history (boring, I know) I need to technically define a few words. A computer: A machine that can be programmed to manipulate symbols. Computers can perform complex and repetitive procedures quickly, precisely and reliably and can quickly store and retrieve large amounts of data." And I need to define science: the intellectual and practical activity encompassing the systematic study of the structure and behavior of the physical and natural world through observation and experiment. So, computer science (if you don't mind my synopsis,) is the

systematic study of the structure and behavior of computers through measurements and experiences that involve data storage and quickness. I would highlight that last sentence, it may come up later.

The first computers to arrive on the scene were the analogue type. Analog computers used moving gears, crystal vibration frequencies (like the one in your watch,) and AC motors. They were very bulky and very hard to manage if they had a bug (sidenote: the word computer bug comes directly from an analog computer that had a moth land in one of the vacuum tubes causing a computer crash.)

Then, we moved on to digital computers, digital computers store information electronically in a complex set of electrical engineering principles. If you'd like information about the electrical engineering side, just send me and e-mail. MY e-mail is on the cover page.

b) Styles of programming
c) Java's style of programming
d) Programming methodology
    i) Top-down development
    ii) Procedural abstraction
e) Ethical issues of computer science
    i) Privacy
    ii) Legal issues
    iii) Social and ethical ramifications
    iv) System reliability

3) Installing the JRE and JDK

a) What is the JRE?
b) What is the JDK?
c) Installing the JRE and JDK
d) Testing your configuration
e) Your first program!

4) Setting up your development environment

a) Introduction to IDEs
b) Choosing your IDE

i) JCreator
ii) Eclipse

c) Installing your IDE
   i) JCreator
   ii) Eclipse
d) Testing your development environment setup
   i) JCreator
   ii) Eclipse

5) The theory of numbers

a) Number bases
b) Base conversion
c) Base arithmetic
d) Shortcuts

6) Basic Java

a) System.out.println(<parameter>)
b) System.out.print(<parameter>)
c) Keywords and reserved words
d) Comments
e) Javadoc @param
f) Javadoc @return
g) Javadoc tool†

7) Java's simple data types

a) Variables
b) Constants
c) Assignment
d) Integer types

i) byte†
ii) short†
iii) int
iv) long†
v) Integer bounds

e) Floating point types
i) float†
ii) double

f) Letters and words
i) char†
ii) String
(1) Concatenation

g) Arithmetic operations
i) Plus (+)
ii) Minus (-)
iii) Multiply (*)
iv) Divide (/)
v) Remainder (%)

h) Assignment operations
i) Arithmetic then assignment (+=, -=, *=, /=, %=)

i) Unary operations
i) Positive (+)
ii) Negative (-)
iii) Increment (++)
(1) Postfix†
(2) Prefix†
iv) Decrement (--)
(1) Postfix†
(2) Prefix†

j) Representations of numbers

i) Integers
   (1) Common base conversions in Java
   (2) Two's compliment†
ii) Floating point numbers
   (1) IEEE 754-2008†
   (2) Round-off errors

k) Type casting
   i) Truncation and Java's warnings
l) Visibility modifiers
   i) public
   ii) private
   iii) protected†
m) Information hiding

8) Boolean Algebra

a) History of Boolean algebra
b) Venn diagrams
c) Truth tables
d) Iterative evaluation
e) De Morgan's law
f) Product of sums†
g) Sum of products †
h) Standard form†
i) Logical operations
   i) and (&&)
   ii) or (||)
   iii) not (!)
   iv) Equals (==)
   v) Not equals (!=)
   vi) Xor (^)†
   vii) Short-circuit evaluation
j) Bitwise operations†

i) and (&)†
ii) or (|)†
iii) xor (^)†
iv) Signed left shift (<<)†
v) Signed right shift (>>)†
vi) Unsigned right shift (>>>)†
vii) Bitwise complement (~)

9) Advanced String Operations

a) Equality
   i) == vs equals method
b) compareTo
c) Escape sequences
   i) \”
   ii) \\
   iii) \n
   iv) \’†
   v) \t†
d) String class methods
   i) String.split()†

10) Java Exception and Error (Runtime and Compile-time) Messages

a) Java exceptions
   i) Standard exceptions
      (1) ArithmeticException
      (2) NullPointerException
      (3) IndexOutOfBoundsException
      (4) ArrayIndexOutOfBoundsException
      (5) IllegalArgumentException
b) Java errors
   i) Compile-time Errors
   ii) Runtime Errors
c) Debugging exceptions

i) Hand tracing code
ii) Debugging output statements
iii) Debuggers†

d) try/catch/finally†

11) Java Methods

a) What is a method?
b) Sequential execution
c) Your first method!
d) Parameters
   i) Formal parameters
   ii) Actual parameters
e) Variables in methods
f) Calling methods
g) Pre and post conditions
h) Assertions within methods
   i) Assert keyword†
i) Method signatures
j) Method overloading
k) Procedural abstraction
l) Functional decomposition
m) Code reuse

12) Java Classes

a) What is a class?
b) Your first class!
c) Class variables
d) Accessor (get) methods
e) Mutilator (set) methods
f) Class interactions
g) Visibility modifiers and information hiding revisited

i) public
ii) private†
iii) protected†

13) Object Oriented Design

a) Encapsulation
b) Polymorphism
c) Abstraction

14) Java Arrays

a) What is an array?
b) How to define an array
c) Arrays initialized at definition†
d) Anonymous arrays
e) Two dimensional arrays
f) Ragged arrays†
g) n dimensional arrays†

15) Looping and Conditional Branching

a) for loop
b) while loop
c) do-while loop†
d) if
e) if/else
f) if/else if
g) if/else if/else
h) Shorthand if (?:)†
i) switch†
j) break†
k) return
l) continue†

16) Advanced Input and Output†

a) Scanner†
b) System.in†
c) System.out†
d) System.err†
e) Stream†
f) Integer.parseint(args)†
g) Double.parseDouble(args)†
h) System.out.printf(args)†
    i) Regex†
        (1) .†
        (2) +†
        (3) *†
        (4) \d†
        (5) \D†
        (6) \s†
        (7) \S†
        (8) \w†
        (9) \W†
        (10) [abc]†
        (11) [^abc]†
        (12) [a-zA-Z]†
        (13) Pattern class†

17) Java Objects

a) What is an object?
b) Static vs non-static methods and variables
c) The new keyword
d) this
e) this(args)†
f) Default initialization of variables†
g) Initialization blocks†
h) null keyword

18) Advanced Datatypes

a) Java packages†
b) Importing packages
c) Conceptual datatypes
    i) Object representation of simple datatypes
        (1) Byte†
        (2) Short†
        (3) Integer
        (4) Long†
        (5) Double
        (6) Float†
        (7) Character†
    ii) String
    iii) Lists
    iv) Sets†
    v) Maps†
    vi) Trees†
    vii) Queues†
    viii) Stacks†
    ix) Heaps†
d) Reference storage vs value storage
    i) Method parameter subtleties
e) Implementing datatypes

i) Lists
   (1) ArrayList
   (2) LinkedList
ii) Sets†
   (1) HashSet†
   (2) TreeSet†
iii) Maps†
   (1) HashMap†
   (2) TreeMap†
iv) Queues†
   (1) Queue†
   (2) PriorityQueue†
v) Stacks†
   (1) Stack†
vi) Heaps†

f) Enhanced (for each) for loop
g) Implementing trees†
   i) Retouch on reference storage†
   ii) Implementing trees by reference†
h) Autoboxing and unboxing†

19) Data Structure Operations, Searching, and Sorting

a) Data structure operations

i) Traversing
   (1) Lists
   (2) Stacks†
   (3) Queues†
ii) Inserting
   (1) Lists
   (2) Stacks†
   (3) Queues†
iii) Deleting
   (1) Lists
   (2) Stacks†
   (3) Queues†

b) Searching
   i) Sequential search
   ii) Binary search

c) Sorting
   i) Selection sort
   ii) Insertion sort
   iii) Mergesort
   iv) Quicksort†
   v) Radix Sort†
   vi) Heapsort†

d) Advanced traversal
   i) Iterator
   ii) ListIterator

e) Tree operations†

Again, trees are very complicated, so they deserve their own subchapters.

i) Traversal†
   (1) In order†
   (2) Post order†
   (3) Pre order†
ii) Insertions†
iii) Deletions†

20) Introduction to Recursion

21) Runtime analysis

a) Informal comparison of runtimes
b) Exact calculation of execution counts
c) Big-O notation
d) Best/worst/average case analysis
e) Runtimes of common algorithms

22) Advanced Recursion

23) Inheritance

a) is-a vs has-a
b) Inheritance
c) Single inheritance
d) extends keyword
e) Abstraction of inheritance
   i) Class hierarchy diagram
f) Implementation of inheritance
g) Method overriding
h) super
i) super(args)
j) Some important Java class hierarchies

24) Abstraction

a) Abstract classes
   i) Abstract methods
b) Interfaces
   i) implements keyword
c) Visibility modifiers revisited
d) Information hiding revisited

25) Polymorphism

a) Object initialization
b) instanceof keyword†
c) Class casting†

26) Comparable and equals method†

a) Comparable interface†
b) Implementing compareTo†
c) Implementing equals method†

27) Testing your code

a) Test cases
b) Boundary conditions
c) Unit testing
d) Integration testing

28) Standard Java Library

a) Object
   i) Every class extends object
b) Math
c) Arrays†
   i) Arrays.sort()†
d) Collections†

i) Collections.sort()†

e) Object.clone()†

f) Exception class†

i) Hierarchy†

29) Generics†

a) What is a generic†

b) Generics used previously explained†

c) Standard Java library generics†

i) List<E>†

ii) ArrayList<E>†

iii) Collection<E>†

30) UIL cheat sheet†

a) First 15 question topics†

b) Other tips from a State Champion†