Table of Contents

[1) Introduction 13](#_Toc389495130)

[2) An introduction to programming 15](#_Toc389495131)

[a) History 15](#_Toc389495132)

[b) Styles of programming 15](#_Toc389495133)

[c) Java’s style of programming 15](#_Toc389495134)

[d) Programming methodology 15](#_Toc389495135)

[i) Top-down development 15](#_Toc389495136)

[ii) Procedural abstraction 15](#_Toc389495137)

[e) Ethical issues of computer science 15](#_Toc389495138)

[i) Privacy 15](#_Toc389495139)

[ii) Legal issues 15](#_Toc389495140)

[iii) Social and ethical ramifications 15](#_Toc389495141)

[iv) System reliability 15](#_Toc389495142)

[3) Installing the JRE and JDK 15](#_Toc389495143)

[a) What is the JRE? 15](#_Toc389495144)

[b) What is the JDK? 15](#_Toc389495145)

[c) Installing the JRE and JDK 15](#_Toc389495146)

[d) Testing your configuration 15](#_Toc389495147)

[e) Your first program! 15](#_Toc389495148)

[4) Setting up your development environment 15](#_Toc389495149)

[a) Introduction to IDEs 15](#_Toc389495150)

[b) Choosing your IDE 15](#_Toc389495151)

[i) JCreator 15](#_Toc389495152)

[ii) Eclipse 15](#_Toc389495153)

[c) Installing your IDE 15](#_Toc389495154)

[i) JCreator 15](#_Toc389495155)

[ii) Eclipse 15](#_Toc389495156)

[d) Testing your development environment setup 15](#_Toc389495157)

[i) JCreator 15](#_Toc389495158)

[ii) Eclipse 15](#_Toc389495159)

[5) The theory of numbers 15](#_Toc389495160)

[a) Number bases 16](#_Toc389495161)

[b) Base conversion 16](#_Toc389495162)

[c) Base arithmetic 16](#_Toc389495163)

[d) Shortcuts 16](#_Toc389495164)

[6) Basic Java 16](#_Toc389495165)

[a) System.out.println(<parameter>) 16](#_Toc389495166)

[b) System.out.print(<parameter>) 16](#_Toc389495167)

[c) Keywords and reserved words 16](#_Toc389495168)

[d) Comments 16](#_Toc389495169)

[e) Javadoc @param 16](#_Toc389495170)

[f) Javadoc @return 16](#_Toc389495171)

[g) Javadoc tool\* 16](#_Toc389495172)

[7) Java’s simple data types 16](#_Toc389495173)

[a) Variables 16](#_Toc389495174)

[b) Constants 16](#_Toc389495175)

[c) Assignment 16](#_Toc389495176)

[d) Integer types 16](#_Toc389495177)

[i) byte\* 16](#_Toc389495178)

[ii) short\* 16](#_Toc389495179)

[iii) int 16](#_Toc389495180)

[iv) long\* 16](#_Toc389495181)

[v) Integer bounds 16](#_Toc389495182)

[e) Floating point types 16](#_Toc389495183)

[i) float\* 16](#_Toc389495184)

[ii) double 16](#_Toc389495185)

[f) Letters and words 16](#_Toc389495186)

[i) char\* 16](#_Toc389495187)

[ii) String 16](#_Toc389495188)

[(1) Concatenation 16](#_Toc389495189)

[g) Arithmetic operations 16](#_Toc389495190)

[i) Plus (+) 16](#_Toc389495191)

[ii) Minus (-) 17](#_Toc389495192)

[iii) Multiply (\*) 17](#_Toc389495193)

[iv) Divide (/) 17](#_Toc389495194)

[v) Remainder (%) 17](#_Toc389495195)

[h) Assignment operations 17](#_Toc389495196)

[i) Arithmetic then assignment (+=, -=, \*=, /=, %=) 17](#_Toc389495197)

[i) Unary operations 17](#_Toc389495198)

[i) Positive (+) 17](#_Toc389495199)

[ii) Negative (-) 17](#_Toc389495200)

[iii) Increment (++) 17](#_Toc389495201)

[(1) Postfix\* 17](#_Toc389495202)

[(2) Prefix\* 17](#_Toc389495203)

[iv) Decrement (--) 17](#_Toc389495204)

[(1) Postfix\* 17](#_Toc389495205)

[(2) Prefix\* 17](#_Toc389495206)

[j) Representations of numbers 17](#_Toc389495207)

[i) Integers 17](#_Toc389495208)

[(1) Common base conversions in Java 17](#_Toc389495209)

[(2) Two’s compliment\* 17](#_Toc389495210)

[ii) Floating point numbers 17](#_Toc389495211)

[(1) IEEE 754-2008\* 17](#_Toc389495212)

[(2) Round-off errors 17](#_Toc389495213)

[k) Type casting 17](#_Toc389495214)

[i) Truncation and Java’s warnings 17](#_Toc389495215)

[l) Visibility modifiers 17](#_Toc389495216)

[i) public 17](#_Toc389495217)

[ii) private 17](#_Toc389495218)

[iii) protected\* 17](#_Toc389495219)

[m) Information hiding 17](#_Toc389495220)

[8) Boolean Algebra 17](#_Toc389495221)

[a) History of Boolean algebra 17](#_Toc389495222)

[b) Venn diagrams 17](#_Toc389495223)

[c) Truth tables 17](#_Toc389495224)

[d) Iterative evaluation 18](#_Toc389495225)

[e) De Morgan’s law 18](#_Toc389495226)

[f) Product of sums\* 18](#_Toc389495227)

[g) Sum of products \* 18](#_Toc389495228)

[h) Standard form\* 18](#_Toc389495229)

[i) Logical operations 18](#_Toc389495230)

[i) and (&&) 18](#_Toc389495231)

[ii) or (||) 18](#_Toc389495232)

[iii) not (!) 18](#_Toc389495233)

[iv) Equals (==) 18](#_Toc389495234)

[v) Not equals (!=) 18](#_Toc389495235)

[vi) Xor (^)\* 18](#_Toc389495236)

[vii) Short-circuit evaluation 18](#_Toc389495237)

[j) Bitwise operations\* 18](#_Toc389495238)

[i) and (&)\* 18](#_Toc389495239)

[ii) or (|)\* 18](#_Toc389495240)

[iii) xor (^)\* 18](#_Toc389495241)

[iv) Signed left shift (<<)\* 18](#_Toc389495242)

[v) Signed right shift (>>)\* 18](#_Toc389495243)

[vi) Unsigned right shift (>>>)\* 18](#_Toc389495244)

[vii) Bitwise complement (~) 18](#_Toc389495245)

[9) Advanced String Operations 18](#_Toc389495246)

[a) Equality 18](#_Toc389495247)

[i) == vs equals method 18](#_Toc389495248)

[b) compareTo 18](#_Toc389495249)

[c) Escape sequences 18](#_Toc389495250)

[i) ” 18](#_Toc389495251)

[ii) \\ 18](#_Toc389495252)

[iii) \n 18](#_Toc389495253)

[iv) \’\* 18](#_Toc389495254)

[v) \t\* 18](#_Toc389495255)

[d) String class methods 18](#_Toc389495256)

[i) String.split()\* 18](#_Toc389495257)

[10) Java Exception and Error (Runtime and Compile-time) Messages 19](#_Toc389495258)

[a) Java exceptions 19](#_Toc389495259)

[i) Standard exceptions 19](#_Toc389495260)

[(1) ArithmeticException 19](#_Toc389495261)

[(2) NullPointerException 19](#_Toc389495262)

[(3) IndexOutOfBoundsException 19](#_Toc389495263)

[(4) ArrayIndexOutOfBoundsException 19](#_Toc389495264)

[(5) IllegalArgumentException 19](#_Toc389495265)

[b) Java errors 19](#_Toc389495266)

[i) Compile-time Errors 19](#_Toc389495267)

[ii) Runtime Errors 19](#_Toc389495268)

[c) Debugging exceptions 19](#_Toc389495269)

[i) Hand tracing code 19](#_Toc389495270)

[ii) Debugging output statements 19](#_Toc389495271)

[iii) Debuggers\* 19](#_Toc389495272)

[d) try/catch/finally\* 19](#_Toc389495273)

[11) Java Methods 19](#_Toc389495274)

[a) What is a method? 19](#_Toc389495275)

[b) Sequential execution 19](#_Toc389495276)

[c) Your first method! 19](#_Toc389495277)

[d) Parameters 19](#_Toc389495278)

[i) Formal parameters 19](#_Toc389495279)

[ii) Actual parameters 19](#_Toc389495280)

[e) Variables in methods 19](#_Toc389495281)

[f) Calling methods 19](#_Toc389495282)

[g) Pre and post conditions 19](#_Toc389495283)

[h) Assertions within methods 19](#_Toc389495284)

[i) Assert keyword\* 19](#_Toc389495285)

[i) Method signatures 19](#_Toc389495286)

[j) Method overloading 19](#_Toc389495287)

[k) Procedural abstraction 19](#_Toc389495288)

[l) Functional decomposition 19](#_Toc389495289)

[m) Code reuse 19](#_Toc389495290)

[12) Java Classes 20](#_Toc389495291)

[a) What is a class? 20](#_Toc389495292)

[b) Your first class! 20](#_Toc389495293)

[c) Class variables 20](#_Toc389495294)

[d) Accessor (get) methods 20](#_Toc389495295)

[e) Mutilator (set) methods 20](#_Toc389495296)

[f) Class interactions 20](#_Toc389495297)

[g) Visibility modifiers and information hiding revisited 20](#_Toc389495298)

[i) public 20](#_Toc389495299)

[ii) private\* 20](#_Toc389495300)

[iii) protected\* 20](#_Toc389495301)

[13) Object Oriented Design 20](#_Toc389495302)

[a) Encapsulation 20](#_Toc389495303)

[b) Polymorphism 20](#_Toc389495304)

[c) Abstraction 20](#_Toc389495305)

[14) Java Arrays 20](#_Toc389495306)

[a) What is an array? 20](#_Toc389495307)

[b) How to define an array 20](#_Toc389495308)

[c) Arrays initialized at definition\* 20](#_Toc389495309)

[d) Anonymous arrays 20](#_Toc389495310)

[e) Two dimensional arrays 20](#_Toc389495311)

[f) Ragged arrays\* 20](#_Toc389495312)

[g) n dimensional arrays\* 20](#_Toc389495313)

[15) Looping and Conditional Branching 20](#_Toc389495314)

[a) for loop 20](#_Toc389495315)

[b) while loop 20](#_Toc389495316)

[c) do-while loop\* 20](#_Toc389495317)

[d) if 20](#_Toc389495318)

[e) if/else 20](#_Toc389495319)

[f) if/else if 21](#_Toc389495320)

[g) if/else if/else 21](#_Toc389495321)

[h) Shorthand if (?:)\* 21](#_Toc389495322)

[i) switch\* 21](#_Toc389495323)

[j) break\* 21](#_Toc389495324)

[k) return 21](#_Toc389495325)

[l) continue\* 21](#_Toc389495326)

[16) Advanced Input and Output\* 21](#_Toc389495327)

[a) Scanner\* 21](#_Toc389495328)

[b) System.in\* 21](#_Toc389495329)

[c) System.out\* 21](#_Toc389495330)

[d) System.err\* 21](#_Toc389495331)

[e) Stream\* 21](#_Toc389495332)

[f) Integer.parseint(args)\* 21](#_Toc389495333)

[g) Double.parseDouble(args)\* 21](#_Toc389495334)

[h) System.out.printf(args)\* 21](#_Toc389495335)

[i) Regex\* 21](#_Toc389495336)

[(1) .\* 21](#_Toc389495337)

[(2) +\* 21](#_Toc389495338)

[(3) \*\* 21](#_Toc389495339)

[(4) \d\* 21](#_Toc389495340)

[(5) \D\* 21](#_Toc389495341)

[(6) \s\* 21](#_Toc389495342)

[(7) \S\* 21](#_Toc389495343)

[(8) \w\* 21](#_Toc389495344)

[(9) \W\* 21](#_Toc389495345)

[(10) [abc]\* 21](#_Toc389495346)

[(11) [^abc]\* 21](#_Toc389495347)

[(12) [a-zA-Z]\* 21](#_Toc389495348)

[(13) Pattern class\* 21](#_Toc389495349)

[17) Java Objects 21](#_Toc389495350)

[a) What is an object? 22](#_Toc389495351)

[b) Static vs non-static methods and variables 22](#_Toc389495352)

[c) The new keyword 22](#_Toc389495353)

[d) this 22](#_Toc389495354)

[e) this(args)\* 22](#_Toc389495355)

[f) Default initialization of variables\* 22](#_Toc389495356)

[g) Initialization blocks\* 22](#_Toc389495357)

[h) null keyword 22](#_Toc389495358)

[18) Advanced Datatypes 22](#_Toc389495359)

[a) Java packages\* 22](#_Toc389495360)

[b) Importing packages 22](#_Toc389495361)

[c) Conceptual datatypes 22](#_Toc389495362)

[i) Object representation of simple datatypes 22](#_Toc389495363)

[(1) Byte\* 22](#_Toc389495364)

[(2) Short\* 22](#_Toc389495365)

[(3) Integer 22](#_Toc389495366)

[(4) Long\* 22](#_Toc389495367)

[(5) Double 22](#_Toc389495368)

[(6) Float\* 22](#_Toc389495369)

[(7) Character\* 22](#_Toc389495370)

[ii) String 22](#_Toc389495371)

[iii) Lists 22](#_Toc389495372)

[iv) Sets\* 22](#_Toc389495373)

[v) Maps\* 22](#_Toc389495374)

[vi) Trees\* 22](#_Toc389495375)

[vii) Queues\* 22](#_Toc389495376)

[viii) Stacks\* 22](#_Toc389495377)

[ix) Heaps\* 22](#_Toc389495378)

[d) Reference storage vs value storage 22](#_Toc389495379)

[i) Method parameter subtleties 22](#_Toc389495380)

[e) Implementing datatypes 22](#_Toc389495381)

[i) Lists 22](#_Toc389495382)

[(1) ArrayList 22](#_Toc389495383)

[(2) LinkedList 23](#_Toc389495384)

[ii) Sets\* 23](#_Toc389495385)

[(1) HashSet\* 23](#_Toc389495386)

[(2) TreeSet\* 23](#_Toc389495387)

[iii) Maps\* 23](#_Toc389495388)

[(1) HashMap\* 23](#_Toc389495389)

[(2) TreeMap\* 23](#_Toc389495390)

[iv) Queues\* 23](#_Toc389495391)

[(1) Queue\* 23](#_Toc389495392)

[(2) PriorityQueue\* 23](#_Toc389495393)

[v) Stacks\* 23](#_Toc389495394)

[(1) Stack\* 23](#_Toc389495395)

[vi) Heaps\* 23](#_Toc389495396)

[f) Enhanced (for each) for loop 23](#_Toc389495397)

[g) Implementing trees\* 23](#_Toc389495398)

[i) Retouch on reference storage\* 24](#_Toc389495399)

[ii) Implementing trees by reference\* 24](#_Toc389495400)

[h) Autoboxing and unboxing\* 24](#_Toc389495401)

[19) Data Structure Operations, Searching, and Sorting 24](#_Toc389495402)

[a) Data structure operations 24](#_Toc389495403)

[i) Traversing 24](#_Toc389495404)

[(1) Lists 24](#_Toc389495405)

[(2) Stacks\* 24](#_Toc389495406)

[(3) Queues\* 24](#_Toc389495407)

[ii) Inserting 24](#_Toc389495408)

[(1) Lists 24](#_Toc389495409)

[(2) Stacks\* 24](#_Toc389495410)

[(3) Queues\* 24](#_Toc389495411)

[iii) Deleting 24](#_Toc389495412)

[(1) Lists 24](#_Toc389495413)

[(2) Stacks\* 24](#_Toc389495414)

[(3) Queues\* 24](#_Toc389495415)

[b) Searching 24](#_Toc389495416)

[i) Sequential search 24](#_Toc389495417)

[ii) Binary search 24](#_Toc389495418)

[c) Sorting 24](#_Toc389495419)

[i) Selection sort 24](#_Toc389495420)

[ii) Insertion sort 24](#_Toc389495421)

[iii) Mergesort 24](#_Toc389495422)

[iv) Quicksort\* 24](#_Toc389495423)

[v) Radix Sort\* 24](#_Toc389495424)

[vi) Heapsort\* 24](#_Toc389495425)

[d) Advanced traversal 24](#_Toc389495426)

[i) Iterator 24](#_Toc389495427)

[ii) ListIterator 24](#_Toc389495428)

[e) Tree operations\* 24](#_Toc389495429)

[i) Traversal\* 25](#_Toc389495430)

[(1) In order\* 25](#_Toc389495431)

[(2) Post order\* 25](#_Toc389495432)

[(3) Pre order\* 25](#_Toc389495433)

[ii) Insertions\* 25](#_Toc389495434)

[iii) Deletions\* 25](#_Toc389495435)

[20) Introduction to Recursion 25](#_Toc389495436)

[21) Runtime analysis 25](#_Toc389495437)

[a) Informal comparison of runtimes 25](#_Toc389495438)

[b) Exact calculation of execution counts 25](#_Toc389495439)

[c) Big-O notation 25](#_Toc389495440)

[d) Best/worst/average case analysis 25](#_Toc389495441)

[e) Runtimes of common algorithms 25](#_Toc389495442)

[22) Advanced Recursion 25](#_Toc389495443)

[23) Inheritance 25](#_Toc389495444)

[a) is-a vs has-a 25](#_Toc389495445)

[b) Inheritance 25](#_Toc389495446)

[c) Single inheritance 25](#_Toc389495447)

[d) extends keyword 25](#_Toc389495448)

[e) Abstraction of inheritance 25](#_Toc389495449)

[i) Class hierarchy diagram 25](#_Toc389495450)

[f) Implementation of inheritance 25](#_Toc389495451)

[g) Method overriding 25](#_Toc389495452)

[h) super 25](#_Toc389495453)

[i) super(args) 25](#_Toc389495454)

[j) Some important Java class hierarchies 25](#_Toc389495455)

[24) Abstraction 25](#_Toc389495456)

[a) Abstract classes 26](#_Toc389495457)

[i) Abstract methods 26](#_Toc389495458)

[b) Interfaces 26](#_Toc389495459)

[i) implements keyword 26](#_Toc389495460)

[c) Visibility modifiers revisited 26](#_Toc389495461)

[d) Information hiding revisited 26](#_Toc389495462)

[25) Polymorphism 26](#_Toc389495463)

[a) Object initialization 26](#_Toc389495464)

[b) instanceof keyword\* 26](#_Toc389495465)

[c) Class casting\* 26](#_Toc389495466)

[26) Comparable and equals method\* 26](#_Toc389495467)

[a) Comparable interface\* 26](#_Toc389495468)

[b) Implementing compareTo\* 26](#_Toc389495469)

[c) Implementing equals method\* 26](#_Toc389495470)

[27) Testing your code 26](#_Toc389495471)

[a) Test cases 26](#_Toc389495472)

[b) Boundary conditions 26](#_Toc389495473)

[c) Unit testing 26](#_Toc389495474)

[d) Integration testing 26](#_Toc389495475)

[28) Standard Java Library 26](#_Toc389495476)

[a) Object 26](#_Toc389495477)

[i) Every class extends object 26](#_Toc389495478)

[b) Math 26](#_Toc389495479)

[c) Arrays\* 26](#_Toc389495480)

[i) Arrays.sort()\* 26](#_Toc389495481)

[d) Collections\* 26](#_Toc389495482)

[i) Collections.sort() 26](#_Toc389495483)

[e) Object.clone()\* 26](#_Toc389495484)

[f) Exception class\* 27](#_Toc389495485)

[i) Hierarchy\* 27](#_Toc389495486)

[29) Generics\* 27](#_Toc389495487)

[a) What is a generic\* 27](#_Toc389495488)

[b) Generics used previously explained\* 27](#_Toc389495489)

[c) Standard Java library generics\* 27](#_Toc389495490)

[i) List<E>\* 27](#_Toc389495491)

[ii) ArrayList<E>\* 27](#_Toc389495492)

[iii) Collection<E>\* 27](#_Toc389495493)

[30) UIL cheat sheet\* 27](#_Toc389495494)

[a) First 15 question topics\* 27](#_Toc389495495)

[b) Other helpful tips from a state champion\* 27](#_Toc389495496)

# 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.)

# An introduction to programming

## History

## Styles of programming

## Java’s style of programming

## Programming methodology

### Top-down development

### Procedural abstraction

## Ethical issues of computer science

### Privacy

### Legal issues

### Social and ethical ramifications

### System reliability

# Installing the JRE and JDK

## What is the JRE?

## What is the JDK?

## Installing the JRE and JDK

## Testing your configuration

## Your first program!

# Setting up your development environment

## Introduction to IDEs

## Choosing your IDE

### JCreator

### Eclipse

## Installing your IDE

### JCreator

### Eclipse

## Testing your development environment setup

### JCreator

### Eclipse

# The theory of numbers

## Number bases

## Base conversion

## Base arithmetic

## Shortcuts

# Basic Java

## System.out.println(<parameter>)

## System.out.print(<parameter>)

## Keywords and reserved words

## Comments

## Javadoc @param

## Javadoc @return

## Javadoc tool\*

# Java’s simple data types

## Variables

## Constants

## Assignment

## Integer types

### byte\*

### short\*

### int

### long\*

### Integer bounds

## Floating point types

### float\*

### double

## Letters and words

### char\*

### String

### Concatenation

## Arithmetic operations

### Plus (+)

### Minus (-)

### Multiply (\*)

### Divide (/)

### Remainder (%)

## Assignment operations

### Arithmetic then assignment (+=, -=, \*=, /=, %=)

## Unary operations

### Positive (+)

### Negative (-)

### Increment (++)

### Postfix\*

### Prefix\*

### Decrement (--)

### Postfix\*

### Prefix\*

## Representations of numbers

### Integers

### Common base conversions in Java

### Two’s compliment\*

### Floating point numbers

### IEEE 754-2008\*

### Round-off errors

## Type casting

### Truncation and Java’s warnings

## Visibility modifiers

### public

### private

### protected\*

## Information hiding

# Boolean Algebra

## History of Boolean algebra

## Venn diagrams

## Truth tables

## Iterative evaluation

## De Morgan’s law

## Product of sums\*

## Sum of products \*

## Standard form\*

## Logical operations

### and (&&)

### or (||)

### not (!)

### Equals (==)

### Not equals (!=)

### Xor (^)\*

### Short-circuit evaluation

## Bitwise operations\*

### and (&)\*

### or (|)\*

### xor (^)\*

### Signed left shift (<<)\*

### Signed right shift (>>)\*

### Unsigned right shift (>>>)\*

### Bitwise complement (~)

# Advanced String Operations

## Equality

### == vs equals method

## compareTo

## Escape sequences

### \”

### \\

### \n

### \’\*

### \t\*

## String class methods

### String.split()\*

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

## Java exceptions

### Standard exceptions

### ArithmeticException

### NullPointerException

### IndexOutOfBoundsException

### ArrayIndexOutOfBoundsException

### IllegalArgumentException

## Java errors

### Compile-time Errors

### Runtime Errors

## Debugging exceptions

### Hand tracing code

### Debugging output statements

### Debuggers\*

## try/catch/finally\*

# Java Methods

## What is a method?

## Sequential execution

## Your first method!

## Parameters

### Formal parameters

### Actual parameters

## Variables in methods

## Calling methods

## Pre and post conditions

## Assertions within methods

### Assert keyword\*

## Method signatures

## Method overloading

## Procedural abstraction

## Functional decomposition

## Code reuse

# Java Classes

## What is a class?

## Your first class!

## Class variables

## Accessor (get) methods

## Mutilator (set) methods

## Class interactions

## Visibility modifiers and information hiding revisited

### public

### private\*

### protected\*

# Object Oriented Design

## Encapsulation

## Polymorphism

## Abstraction

# Java Arrays

## What is an array?

## How to define an array

## Arrays initialized at definition\*

## Anonymous arrays

## Two dimensional arrays

## Ragged arrays\*

## n dimensional arrays\*

# Looping and Conditional Branching

## for loop

## while loop

## do-while loop\*

## if

## if/else

## if/else if

## if/else if/else

## Shorthand if (?:)\*

## switch\*

## break\*

## return

## continue\*

# Advanced Input and Output\*

## Scanner\*

## System.in\*

## System.out\*

## System.err\*

## Stream\*

## Integer.parseint(args)\*

## Double.parseDouble(args)\*

## System.out.printf(args)\*

### Regex\*

### .\*

### +\*

### \*\*

### \d\*

### \D\*

### \s\*

### \S\*

### \w\*

### \W\*

### [abc]\*

### [^abc]\*

### [a-zA-Z]\*

### Pattern class\*

# Java Objects

## What is an object?

## Static vs non-static methods and variables

## The new keyword

## this

## this(args)\*

## Default initialization of variables\*

## Initialization blocks\*

## null keyword

# Advanced Datatypes

## Java packages\*

## Importing packages

## Conceptual datatypes

### Object representation of simple datatypes

### Byte\*

### Short\*

### Integer

### Long\*

### Double

### Float\*

### Character\*

### String

### Lists

### Sets\*

### Maps\*

### Trees\*

### Queues\*

### Stacks\*

### Heaps\*

## Reference storage vs value storage

### Method parameter subtleties

## Implementing datatypes

### Lists

### ArrayList

### LinkedList

### Sets\*

### HashSet\*

### TreeSet\*

### Maps\*

### HashMap\*

### TreeMap\*

### Queues\*

### Queue\*

### PriorityQueue\*

### Stacks\*

### Stack\*

### Heaps\*

## Enhanced (for each) for loop

## Implementing trees\*

I chose to make the implementation of trees its own subchapter because it is a pretty complicated subject. It involves a lot of reference variable manipulation.

### Retouch on reference storage\*

### Implementing trees by reference\*

## Autoboxing and unboxing\*

# Data Structure Operations, Searching, and Sorting

## Data structure operations

### Traversing

### Lists

### Stacks\*

### Queues\*

### Inserting

### Lists

### Stacks\*

### Queues\*

### Deleting

### Lists

### Stacks\*

### Queues\*

## Searching

### Sequential search

### Binary search

## Sorting

### Selection sort

### Insertion sort

### Mergesort

### Quicksort\*

### Radix Sort\*

### Heapsort\*

## Advanced traversal

### Iterator

### ListIterator

## Tree operations\*

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

### Traversal\*

### In order\*

### Post order\*

### Pre order\*

### Insertions\*

### Deletions\*

# Introduction to Recursion

# Runtime analysis

## Informal comparison of runtimes

## Exact calculation of execution counts

## Big-O notation

## Best/worst/average case analysis

## Runtimes of common algorithms

# Advanced Recursion

# Inheritance

## is-a vs has-a

## Inheritance

## Single inheritance

## extends keyword

## Abstraction of inheritance

### Class hierarchy diagram

## Implementation of inheritance

## Method overriding

## super

## super(args)

## Some important Java class hierarchies

# Abstraction

## Abstract classes

### Abstract methods

## Interfaces

### implements keyword

## Visibility modifiers revisited

## Information hiding revisited

# Polymorphism

## Object initialization

## instanceof keyword\*

## Class casting\*

# Comparable and equals method\*

## Comparable interface\*

## Implementing compareTo\*

## Implementing equals method\*

# Testing your code

## Test cases

## Boundary conditions

## Unit testing

## Integration testing

# Standard Java Library

## Object

### Every class extends object

## Math

## Arrays\*

### Arrays.sort()\*

## Collections\*

### Collections.sort()

## Object.clone()\*

## Exception class\*

### Hierarchy\*

# Generics\*

## What is a generic\*

## Generics used previously explained\*

## Standard Java library generics\*

### List<E>\*

### ArrayList<E>\*

### Collection<E>\*

# UIL cheat sheet\*

## First 15 question topics\*

1. Base conversion
2. Expression
3. Loop
4. String
5. Array
6. Expression
7. Boolean logic
8. if/else
9. class, object oriented programming
10. class, object oriented programming
11. bitwise
12. Math
13. print, println, output
14. printf
15. method call, return

## Other helpful tips from a state champion\*