Table of Contents

[I. Introduction 2](#_Toc362580472)

[II. Installing the JRE and JDK 3](#_Toc362580473)

# 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 designed 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 complete understanding of computer science, you will be able to implement that understanding in Java programs, you will understand how to work with large projects (well, a large project: the GridWorld case-study,) 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 a B.S. in Electrical Engineering, 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 want 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, we will jump right into a programming language instead of leaning general concepts first. 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.

# An introduction to programming

## History

## Styles of programming

## Java’s style of programming

## Programming methodology

### Top-down development

### Procedural abstraction

### Pre and post conditions

## 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

# Java’s simple data types

## Variables

## Constants

## Integer types

### byte

### short

### int

### long

### Common base conversions in Java

## Floating point types

### float

### double

### IEEE 754-2008

## Letters and words

### char

### String