**Sunday**

**Evening Class: Introduction** (120 minutes)

Objective: Coders will get comfortable with each other, the instructor, and the format of the course. Coders will understand the basics of using the command line and git. There will be an introduction to programming and to the most basic possible Java program.

* (~15 minutes) Encourage the coders to get to know each other with a few ice breakers
* (~5 minutes) Discuss why they are here and write down goals for the week
* (~5 minutes) Give the students an idea of the schedule for the week
* (~10 minutes) Classroom expectations: Let the students know what you expect from them - raising hands, being respectful to one another, collaborative not competitive, etc. Expectations on the computer: staying on the site the teacher tells you to be on. Encourage students to ask questions whenever anything’s unclear.
* (~10 minutes) Show the command line: ls, cd, mkdir, rmdir, rm, nano, clear, javac, java.
* (~10 minutes) Command line Practice activity: in the Day 1 Folder, give students the file called “Command Line Practice” and have them replicate that structure on their desktops using only the command line.
* (~15 minutes) Github setup and basics
  + Have all coders make a github account here: <https://github.com/>
  + Walk them through creating a local repo, creating and committing a file, and creating a Github repo and connecting it to the local, and pushing changes to github - details are in source and the student manual if needed. Make sure they understand that all projects in the class should be in github, and that all commits should have an appropriate commit message.
    - Through this, coders should be familiar with the following commands: git init, git status, git add, git commit -m “your message here”, git push origin branchname, and git branch
* (~30 minutes) Graph Paper Programming activity - to get students who have never programmed on board with the basics.
* (~20 minutes) Netbeans IDE Tour. Guide students through creating a basic “Hello, World” application using the Netbeans IDE (example in In Class Examples)
  + New->project, name it “HelloWorld”, put it in the git projects folder.
  + Add file->Java main class, name is “HelloWorld”.
  + Where it says “// TODO code application logic here”, replace with “System.out.println(“Hello,World”);”.
  + Press the green arrow to run, then save file and add and commit to Git
  + Give students time to poke around Netbeans. Share this shortcuts list: <https://shortcutworld.com/en/NetBeans/8/win/all>

**Monday**

**Morning Class: Variables, Math, Logic, User Input** (180 minutes)

Objective: Coders will be familiar with several variable types in Java. They will learn how to use math operations and shortcuts, and logic operators for numbers and strings.

* (~10 minutes) Have the coders do Assignment 1: Printing to reinforce the Java basics, use of Netbeans, and use of git.
* (~15 minutes) Introduction of Variables:
  + Introduce the following types of variables in Java: int: an integer, double: a floating point value (ex. 1.25), boolean: can only be true or false, string: a string of characters, always defined using quotation marks.
  + Introduce how to create and instantiate a variable: “type name; name = value;” or “type name = value;”. Names can be words, always start with lowercase letter, convention for multiword is “firstSecondThird.”
  + Go over the difference between printing a String and printing a variable’s value.
  + Talk through the example project VariablePractice.java in the In Class Examples folder. Do some math with the ints and have then create and define new variables in each type.
* (~20 minutes) Have coders do Assignment 2: Variables to practice working with different types of variables.
* (~10 minutes) Go over Java’s math operations: adding Strings and numbers, subtracting, multiplying, and dividing. Go over shortcuts: +=, -=, ++, --.
* (~20 minutes) Have coders do Assignment 3: Math Operations to get to know math operations and shortcuts in Java.
* (~10 minutes) Go over how to define and use booleans using logic operators on numbers: ==, !=, >, <, >=, <=. Explain that comparing Strings is different, and show how the .equals and .compareTo methods work. Then show the compound logic operators ! (not), && (and), and || (or).
* (~20 minutes) Have coders do Assignment 4: Booleans to get to know boolean logic.
* (~15 minutes) Introduce the idea of capturing user input using a scanner. Go over the UserInput.java class in the In Class Examples folder. Note that the first part just prompts the user, the second part uses ‘print’ instead of ‘println’ to keep the cursor on the same line, and the third part stores the user’s input as a variable. Note also that nextLine is usually the way to go when getting data from a user.
* (~60 minutes) Have coders do Assignment 5: Scanner to get used to capturing and using user input, and to practice the other subjects covered this morning.

**Afternoon Class: While Loops, Conditionals, and Random Numbers** (180 minutes)

Objective: Coders will be exposed to the concepts of conditionals and while loops and they will practice them all in several assignments. They will learn how to use random number generators.

* (~20 minutes) Do the Conditionals activity.
* (~15 minutes) Explain conditionals in Java. Go over if statements, if-else, and else if using the Conditionals (if), Conditionals 2 (if-else) and Conditionals 3(else if) in the In Class Examples folder.
* (~30 minutes) Have coders do Assignment 6: Conditionals.
* (~20 minutes) Do the Loops activity.
* (~10 minutes) Explain While Loops in Java. Explain that they are often used to keep doing a process until something happens - until a user enters the magic password, rolls doubles, etc. Sometimes they are used to do something a set number of times using a counter. Walk through the example file WhileLoops in the in Class Examples folder.
* (~20 minutes) Have the coders do Assignment 7: While Loops
* (~5 minutes) Explain random number generation using Math.random(). Go over example file NumberGame.java in the In Class Examples folder.
* (~20 minutes) Have coders do Assignment 8: Doubles to practice random number generation and while loops.
* (~40 minutes) Have coders do Assignment 9: HighOrLow for more difficult practice of all the new concepts from this afternoon.

**Evening Class: Seminar 1** (90 minutes)

Objective: Students will be introduced to the programming language Python. They will do a tutorial in Python, and compare and contrast Python with Java.

* (~5 minutes) Introduce Python: Python is an interpreted, high-level programming language. It used general-purpose, so it can be used for a lot of different types of projects. Python differs from many other languages in its use of whitespace instead of punctuation, and its extremely efficient and succinct syntax.
* (~70 minutes) Have the coders work through the Python tutorial here: <https://www.codecademy.com/learn/python>, as much as they can get through.
* (~5 minutes) Discuss how Python is similar to Java. Discuss common features in syntax, how the languages are similarly object-oriented, and other similarities.
* (~10 minutes) Discuss the differences between Python and Java. There are many differences in syntax, and in the purpose and common uses. Discuss the pros and cons of using each language.

**Tuesday**

**Morning Class: Functions, Arrays, and For Loops** (180 minutes)

Objective: Coders will learns what functions are and how to define and use them. They will learn how to use arrays. They will learns what a for loop is, compare for loops to while loops, and practice for loops.

* (~10 minutes) Introduce functions. Functions are called Methods in Java. It is a set of code which has a name and can be called (used) at any point in a program by using its name. Methods are like subprograms that act on data and often return a value. Go over the parts of a method:
  + private/public: used only in this class or can be used somewhere else
  + Static: for now just use static
  + Void or a type: what this function returns. Can be nothing (void) or any type of thing.
  + Name: the name of the function
  + In parentheses, the input variables: type and name, separated by commas.
* (~15 minutes) Go through the Functions project in the In Class Examples folder, which uses several different kinds of functions.
* (~35 minutes) Have coders do Assignment 10: Functions. Be aware that student who have not yet taken pre-algebra will struggle with this assignment.
* (~20 minutes) Introduce arrays! An array is a collection of elements that all have the same type. Explain how to create a new array, how to put values in it, and how to use the length by going through the Arrays.java file in In Class Examples.
* (~40 minutes) Have coders do Assignment 11: Arrays.
* (~15 minutes) Introduce for loops by going through the In Class Example ForLoops.java.
* (~45 minutes) Have coders do Assignment 12: For Loops.

**Afternoon Class: Using Classes and Writing Unit Tests** (180 minutes)

Objective: Coders will be introduced to the concept of Object Oriented Programming and how classes are objects in Java. Coders will be introduced to constructors, getters and setters, and other common methods. Coders will practice using OOP by writing several classes that use each other. Coders will be introduced to Test Driven Development and write their first Junit tests.

* (~10 minutes) Explain Object Oriented Programming (OOP): a way of programming based on the concept of "objects", which contain data (fields, attributes), and procedures (methods). The methods have the same format as the ones they’ve already looked at, but they can be used by another class.
* (~20 minutes) Walk through the Car project in the In Class Examples folder. Note there are 2 classes, Car and CarExample, which is a main class.
* (~30 minutes) Have coders do Assignment 13: Car, Part 1. They will do Part 2 very soon after you introduce unit testing.
* (~10 minutes) Introduce unit testing to the class. Explain that each method - and each possible decision within each method - has to be unit tested. We will use Junit to test. n Netbeans, to create a new Junit test set, right click on the project you want to test, select ‘New’, then select ‘Junit test…’
* (~10 minutes) Walk through the example file in the Car project called CarTest.
* (~20 minutes) Have coders do Assignment 13: Car, Part 2.
* (~80 minutes) Have the coders do Assignment 14: Person. This is the hardest assignment yet and combines a lot of the concepts we have done so far. If people get stuck, show them the solution for PhoneBookExample and see if that helps them work backwards.

**Evening Class: Seminar 2** (90 minutes)

Objective: Students will be introduced to the programming language C. They will do a tutorial in C, and compare and contrast C with Python and Java.

* (~5 minutes) Introduce C: C is an imperative programming language with a static type system. C has a memory-use system that uses pointers, and is very fast. C is often used in operating systems, compilers, editors, databases, and database management systems.
* (~70 minutes) Have the coders work through the C tutorial here: <http://www.learn-c.org/> , as much as they can get through.
* (~5 minutes) Discuss how C is similar to Java and how it’s similar to Python. Discuss common features in syntax and other similarities.
* (~10 minutes) Discuss the differences between C and Python and Java. There are many differences in syntax, memory access, and in the purpose and common uses. Discuss the pros and cons of using C versus Python and versus Java.

**Wednesday**

**Morning Class: Input/Output, Polymorphism, and ArrayLists** (180 minutes)

Objective: Coders will learn how to do file I/O in Java. Then, they will become familiar with the concept of polymorphism and inheritance in Java. They will learn about ArrayLists and practice using them.

* (~15 minutes) Go over basic I/O in Java using only BufferedWriter and BufferedReader. We are not covering try-catch blocks, but explain that we always use them for I/O in case something goes wrong. In Class Example FileOutput.java writes a simple file. Try changing the context to write and running it again, and changing the name of the file and running it again. FileInput.java reads a file and prints it out to the console. Again, change around the file and run again.
* (~15 minutes) Have coders do Assignment 15: I/O.
* (~30 minutes) Have coders do Assignment 16: Puzzles.
* (~60 minutes) Introduce Polymorphism, interfaces, and inheritance. Walk through the In Class Examples project Animals. Because this is such a complex concept, we’re not going to have an assignment on it.
* (~15 minutes) Introduce ArrayLists. Compare and contrast Arrays and ArrayLists. ArrayLists are more often used because their size can change and they don’t rely as much on indices. Go over contructors, adding and remove elements, finding elements, and getting the size of an ArrayList.
* (~45 minutes) Have coders do Assignment 17: ArrayLists.

**Thursday**

**Morning Class: Challenge Projects** (180 minutes)

Objective: Coders will be given a list of projects that are increasingly challenging, and will complete all that they can within the class period. The projects combine everything they’ve learned.

* (~180) Have coders do as many of the projects in Assignment 18: Challenge Projects as they can in the time.

**Afternoon Class: Begin Final Projects** (180 minutes)

Objective: Coders will be given a list of tough projects to choose from for their final assignment for the week. They will start their final projects. (Middle school difference: Middle schoolers have to finish the final project in this period, so it is highly recomended that they just finish a challenge project from the morning as their final project.)

* (~5 minutes) Introduce the final project. Let coders know they have this period and tomorrow morning’s period to work on it. They can choose from the list “Final Project Options” or continue to work on one of the morning’s Challenge Projects. It is highly recommended that they map out the classes and methods before they start. Encourage them to try something that looks difficult.
* (~175 minutes) Coders work on final projects. If they finish early, encourage them to add more exciting features to their projects. If they are finished quite early, have them do another project from the final projects list.

**Evening Class: Seminar 3** (90 minutes)

Objective: Students will be introduced to the programming language C++. They will do a tutorial in C++, and compare and contrast C++ with C, Python, and Java.

* (~5 minutes) Introduce C++ (Pronounced See Plus Plus): C++ is built off of C, so it is compatible with most C programs. It’s a compiled language, and can be one of the fastest languages when used correctly. It is an object-oriented language. Like C, C++ has low-level memory manipulation (pointers).
* (~70 minutes) Have the coders work through the C++ tutorial here: <http://www.learncpp.com/>, as much as they can get through.
* (~5 minutes) Discuss how C++ is similar to C, Java, and Python. Discuss common features in syntax, which languages are similarly object-oriented, and other similarities.
* (~10 minutes) Discuss the differences between C++ and C, Python, and Java. There are many differences in syntax, and in the purpose and common uses. Discuss the pros and cons of using C++ versus C, versus Python and versus Java.

**Friday**

**Morning Class: Continue Final Projects** (180 minutes)

Objective: Coders will continue working on their final projects with the intention of finishing during this period. (Middle school difference: Middle schoolers do the afternoon class - wrapping up projects and preparing presentations - instead of this morning class because they have presentations in the afternoon.)

* (~180 minutes) Coders work on final projects. If they finish early, encourage them to add more exciting features to their projects. If they are finished quite early, have them do another project from the final projects list.

**Afternoon Class: Finish Final Projects and Prepare Presentations** (180 minutes)

Objective: Coders will finish their final projects if they have not yet. Then they will prepare and practice their final presentations. (Middle school difference: They do not have this class period.)

* (~5 minutes) Today, we will be making and practicing our final presentations for families which will happen tomorrow.
* (~60 minutes) Coders work on their presentations.
* (~45 minutes) Coders break into small teams and practice presenting each other and giving constructive feedback.
* (~70 minutes) Coders can continue to work on their presentations, put finishing touches and additional features on their projects, or work on a new project.

**Evening Class: Seminar 4** (90 minutes)

Objective: Students will be introduced to the programming language Racket. They will look through a Racket tutorial, and compare and contrast Racket with Python, C, C++, and Java. (Middle school difference: They do not have this class period.)

* (~5 minutes) Introduce Racket: Racket is a multi-paradigm programming language. It is related to well-known languages Lisp and Scheme. It is used as a platform to develop other languages, for scripting, and for graphics. It is considered a great learning language.
* (~70 minutes) Look over the Racket tutorial in the Friday folder with the coders, as much as you can get through.
* (~5 minutes) Discuss how Racket is similar to C, C++, Java, and Python. Discuss common features in syntax, which languages are similarly object-oriented, and other similarities.
* (~5 minutes) Discuss the differences between Racket and C++, C, Python, and Java. There are many differences in syntax, and in the purpose and common uses. Discuss the pros and cons of using Racket versus C++, C, Python, and Java.
* (~5 minutes) Discuss why Racket would be a good language to learn first when starting out in computer science.

**Sources**

<http://product.hubspot.com/blog/git-and-github-tutorial-for-beginners> for Github directions in Student Manual

<https://shortcutworld.com/en/NetBeans/8/win/all> Netbeans shortcuts

<https://learnxinyminutes.com/docs/racket/> Racket resource

<https://grahammitchell.com/assignments/> Assignment inspiration

[www.dreamincode.net/forums/topic/78802-martyr2s-mega-project-ideas-list/](http://www.dreamincode.net/forums/topic/78802-martyr2s-mega-project-ideas-list/) final projects source

<https://github.com/aakashsharan/inheritance-polymorphism-java> polymorphism in class example