**Part 1: Java Syntax**

**Basic Elements**

* Identifiers are the user-defined names in programming. (ex: variables, constants, methods, classes, labels, packages, etc.)
  + Made up of characters letters, underscores (\_), dollar signs ($), or digits.
  + Must NOT start with a digit as the first character
  + Must not be a reserved word
  + Case Sensitive
* Built-in (primitive) types (boolean, char, byte, short, int, long, float, double)
* Literals:
  + integer, floating-point (double, float), character, string
  + Escape sequences for special characters
    - used for writing character literals, and used in string literals
    - Know the common ones

**Operators:**

* General Concepts
  + operators and operands
  + Arity (unary, binary, ternary)
  + Precedence and associativity
  + Cascading
* Arithmetic operators: + - / \* %
  + understand int vs. floating-point division (/ and %)
* increment and decrement: ++ --
  + Post-increment vs. Pre-increment: x++ vs. ++x
  + Post-decrement vs. Pre-decrement: x-- vs. --x
* assignment operator: =(single equal sign)
  + Understand the concept of **L-value** vs **R-value**
* Other shortcut operators: += -= \*= /= %=
* automatic type conversions -- know which ones are allowed
* cast operations for explicit type-conversion
* Logical operators: == != < <= > >=
* Boolean operators: && (AND), || (OR), ! (NOT)

**Console I/O**

* System.out -- standard output (object of type PrintStream)
* Understand how to do console output with System.out, including the methods:
  + print()
  + println()
  + printf()
* Simple string concatenation
* System.in -- standard input
* Be able to do basic console input with the Scanner class, reading from System.in

**Control Structures**

* *Flow of Control* concepts:
  + sqeuential
  + Selection (branching)
  + Repetition (looping)
* The if/else selection statement
  + Syntax format and rules for if and for if/else statements
  + How the *test expression* is evaluated
  + Syntax vs. readability conventions (indentation, etc)
  + Know when to use a block
* The switch selection statement
  + Keywords switch, case, default, break
  + Understand how switch blocks work
  + only works with integer and character expressions
  + know when to use break
* Conditional operator
  + Special operator that acts similar to if-statements
  + Three operands (ternary operator)
  + Operator consists of three parts, separated by symbols: ? :
  + Example: x = (y < 0 ? 5 : 10);
* while and do-while loops
  + Syntax format and rules for both
  + How the *test expression* is evaluated
  + Know the difference between while and do-while
  + Know when to use a block
  + Know when the loop repeats and when it quits
* for loops
  + Three parts to the header: initial condition, test expression, iterative expression
  + Good for counting-controlled loops
  + scope of loop control variable (if declared inside header)
* Special statements affecting loop control
  + break
  + continue
* **Remember**: In Java, the *test expression* of an if-statement (and in loops) *must* be a Boolean expression -- i.e. some operation or phrase that returns a Boolean value.

**Using Java Libraries**

* Know how to use the basic libraries that we've looked at in examples, including:
  + java.util.Scanner
  + java.util.Random

**Methods (i.e. functions)**

**Method Basics**

* Know what a method is
* Reasons for writing methods (divide-and-conquer, reusability)
* Perspective: builder vs. caller

**Using Methods**

* Know how to **call** a method
  + syntax
  + passing arguments
  + using the returned value
* Understand how they are like mathematical functions
* Predefined methods
  + methods in libraries
  + static vs. instance -- know the difference in the call syntax

**Building Methods**

* Prototype -- name, return type, parameter list
  + Know the format
  + Know what to specify in the parameter list
* Header includes modifiers (like public and static), as well as prototype
* Definition -- header, along with method body
  + the keyword return -- returning values from methods
  + using the formal parameters in the method
* Scope (and how it relates to methods)
  + Meaning of *scope*
  + Variables local to function blocks
  + Variables local to internal blocks (like loop bodies)
  + Scope of method parameters
* void as a return type
* Empty parameter lists
* Method overloading - methods with same name and different parameter lists

**Strings**

* Know how to construct a String object
* Know the common String methods: length(), concat(), equals(), compareTo(), charAt(), substring(), trim(), toLowerCase(), toUpperCase(), replace(), valueOf(), and the concatenation operator +
* Know the difference between String and StringBuilder
* Know the common StringBuilder methods: append(), insert(), delete(), along with common methods that work the same as for String (charAt(), length() for example), and know how to construct one

**Arrays**

* Understand arrays, this includes multi-dimensional arrays
* Know how to declare an array variable, and then create the array with the new operator  
  int[] array1 = new int[size];
* Understand how array indexing is used, this includes multi-dimensional arrays   
  Remember array indexing starts at zero and **NOT** one.   
  given   
  array =   
  { {1, 2, 3, 4},   
    {5, 6, 7},   
    {8} }   
  Then array[1][2] is equal to 7.
* Know that the length of an array (say named array) can be found by using array.length
* Know that the reference (name) of an array can point to any array of that type   
  (two array references can point to the **same** array)
* Understand all the ways to declare an array variable. All the following are the same and are legal   
  (the first is the preferred way):   
  int[][][] thingy;   
  int[][] thingy[];   
  int[] thingy[][];   
  int thingy[][][];
* Understand how to declare, create, and initialize an array all at once   
  int[] bubba = {2, 5, -56};
* Know how to pass arrays in and out of methods

**Part 2: Java OOP**

**Classes and Objects**

* Objects and classes and the relationship between them
* Understand how to declare an object reference variable and then create the object with new, along with how to invoke the chosen constructor
* Understand how objects are passed in and out of methods
* Constructors:
  + Know what a constructor is for
  + Know how to identify a constructor in a class definition (i.e. the constructor has the same name as the class and no return type)
  + Know how to invoke a constructor (when building an object with new
* The modifiers public and private
* Accessor and mutator methods
* The special toString() method
* The static modifier, and the difference between static variables and methods vs. instance variables and methods, as well as which variables can be accessed from which methods
* The keyword this
* Class scope vs. local scope (like in a method)
* Arrays of objects (array of reference variables, each can attach to an object)
* Object as method parameter (pass by reference)

**Inheritance and Polymorphism**

* Superclasses and subclasses, and deriving a class with the keyword extends
* The use of super to invoke parent constructors and methods
* Understand overriding methods in derived classes
* Every class is derived in some way from class Object. Understand these 3 special methods inherited from Object by every class:
  + equals()
  + toString()
  + clone()
* Understand the modifiers protected and final, and what they do when applied to class data and methods
* Abstract classes and their relationship with abstract methods
* Understand polymorphism and dynamic binding
  + How base class variables can be attached to derived class objects
  + How base class methods can be overridden in derived classes
  + How dynamic binding allows method calls (through base class variables) to still invoke child versions of overridden methods
  + When casting is necessary between parent and child, and when it is not

**Part 3: Java GUI**

**Basic Concepts**

* java.awt -- original Java GUI package
* javax.swing -- newer GUI package containing the library of "Swing components"
* Know what a component is
* Know what a container is
* Know what an event is, and how it relates to GUI applications

**Painting with class Graphics**

* Understand when the paintComponent() method gets called for a component. Especially the "system-level triggers" that were discussed
* Understand what kinds of drawing can be done inside paintComponent() through the Graphics class methods
* These include draw methods for shapes like: lines, ovals, rectangles, polygons, round rectangles, arcs, and strings
* Understand the calls we've seen in examples that used class Graphics, including the draw and fill methods, as well as the methods for changing colors and fonts
* Understand the basic use of the Color class, to create a color using the RGB color scheme, or one of the Color constants
* Know how to set up a polygon (it requires sets of x,y coordinates, and it can be set up with two parallel arrays, one for x coordinates and one for y coordinates), and how to draw it with class Graphics
* Understand the difference between the Graphics and Graphics2D classes

Some practice array algorithms to try (coding practice)

(You should be able to do these and other similar array algorithms)

Compute and print the sum (product, average) of the elements of a numerical array.

1. A method that returns the maximum element of an array
2. A method that prints all array elements that are between two given values (parameters)
3. A method that counts up all the even numbers in an array
4. A method that returns an array that contains all odd elements from an original array (incoming parameter)

**class** Student {

// instance variable (non-static)

**private** **int** testGrade;

// static variable (class variable)

**private** **static** **int** *numStudents* = 0;

// class constant

**private** **final** **static** **int** *pointsPossible* = 100;

**public** Student() {

testGrade = 0;

}

**public** **void** setGrade(**int** gr) {

testGrade = gr;

}

**public** **int** getGrade() {

**return** testGrade;

}

**public** **static** **void** incrementNumStudents() {

*numStudents*++;

}

**public** **static** **int** getNumStudents() {

**return** *numStudents*;

}

}

In this sample code:

* **testGrade is an intance variable**. Each object of type Student will have its own copy of testGrade
* **numStudents is a class varaible (static)**. There is only one variable shared by the whole class. **The variable's value can be changed, but changes are seen by all objects**
* **pointsPossible is a class constant**. There is only one variable (because of static), and its value cannot be changed
* **setGrade and getGrade are instance methods**. **They must be called through individual objects**
* **incrementNumStudents and getNumeStudents are static methods**. They cannot access instance variables of the class, but they can be called through the class name, regardless of whether any objects have been created