**Test 1 Review Sheet**

This is a checklist of topics that have been covered in the course so far.

**General Computer and Programming Concepts**

* Components (CPU, RAM, disk storage, I/O devices)
* bits and bytes
* evolution of programming languages
* interpreted vs. compiled languages
* Steps involved in software development
* Basic background of Java
* syntax vs semantics

**Building Basic Java Programs**

* A Java source file must end with the extension .java and should have the exact same name as the public class name.
* To compile a class in a Java source code file (say Yadda.java) type:   
  javac Yadda.java
* Each class (there may be more than one) in a source code file, is compiled into a .class file.
* To execute a java application (say named Yadda.class) type   
  java Yadda
* The main method has the following signature   
  public static void main(String[] args)
* Single line comments in Java start with //   
  Block-style comments in Java start with /\* and end with \*/

**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)
  + short-circuit evaluation of && and ||

**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 the basic SDK library organization: classes grouped into packages
* Know how and when to use the import statement
  + Remember, everything from java.lang is automatically imported into every Java program
* Understand the items in the API description of a Java class library:
  + Field summary
  + Constructor summary
  + Method summary
* Know the difference between static members of a class and instance members
* Know that for static members, you call them through the class name and dot-operator:
* ClassName.member
* Know that for instance members, you have to create an object first
* Know how to create a basic object using a class type:
* ClassName variableName = new ClassName(*arguments*);
* Know how to call an instance member (data or method) through that object:
* objectName.member
* Know how to use the basic libraries that we've looked at in examples, including:
  + java.lang.Math
  + 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
* Know how to interpret and how to write an enhanced for-loop, especially for use with arrays
* Understand how to interpret and how to write a variable-length parameter list on a method. Know when it is legal to do so

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