**Java Fundamentals**

* History
  + Dynamic programs
  + web programs / applets
  + Security by confining programs to Java execution environment
  + Portability
* Bytecode
  + Highly optimized set of instructions executed by Java run-time system (JVM)
  + Java -> bytecode -> JVM (interpreter)
  + Allows portability
* Object Oriented Programming
  + Encapsulation
    - Binding of code / data it manipulates
    - Private / public
    - Java’s basic unit of encapsulation is the class
  + Polymorphism
    - “one interface, multiple methods”
    - Reduce complexity by allowing same interface to be used to specify a general class of action
  + Inheritance
    - One object acquires properties of another

**Data Types and Operators**

* Data types
  + Important because Java is strongly typed language
  + Prevents errors
* Literals
  + 12 is an int but 12L is a long
  + 12.0 is a double but 12.0F is a float
  + Starting with JDK 7 you can use \_ as to integers/floating points
    - 123\_45\_1234 is 123,451,234
  + Hexadecimal
    - Begins with 0x or 0X
  + Octal
    - starts with a zero
    - ex. 011 is 9
  + Binary
    - Starts with 0B
    - 0B1100 is 12

**Control Statements**

* For loops

int i, j;

for(int i = 0; j = 10; i < j; i++, j--) {

// do stuff

}

* + You can leave some parts of the for loop out (initialization / incrementing)
* Enhanced for loop
* Use break as a form of goto
  + break *label*;
  + define label blocks like this:
    - one: { // do stuff }
  + continue can also use labels

**Classes, Objects, and Methods**

* Class – template that defines the form of an object
  + Variables and methods that constitute a class are called members of the class
  + Well defined classes should define one and only one logical entity
* Objects – instances of a class
  + **new** creates a reference to an objet
    - dynamically allocates memory for an object at run time
    - all classes in java are dynamically allocated
    - can fail if not enough memory to create the object
* when assigning primitive values to a variable, the value is copied
* when assigning objects to variables, you are assigning it to a reference
* Garbage collection
  + Java has auto garbage collection
  + When no references to an object exist, it is assumed that it is no longer needed, and the memory occupied by the object is released
  + Only does this when 1) there are objects to recycle and 2) there is a need to recycle them
* finalize() method
  + method that is called just before an object’s final destruction
  + it happens just before garbage collection so we never really know when it will happen
* this keyword
  + refers to the object being called

**More Data Types and Operators**

* Arrays
  + Assignment creates a reference
* Queues FIFO
  + Generally have two operations: get and put
  + Operations are consumptive (after retrieval, cannot be retrieved again)
  + Can be full or empty
  + Two types: circular and non-circular
    - Circular – reuses locations in underlying array when elements are removed
    - Non-circular – does not reuse location and eventually becomes exhausted
* Stacks LIFO
* For each / enhanced for loop
  + for(type itr-var : collection)
  + use when you have to iterate through a collection from start to finish
  + can be force stopped using a break statement
* Strings
  + Are objects in Java
  + .equals is used to check for equality of objects. == is used to check if references are equal
  + Strings are immutable
    - StringBuffer objects can be used instead for mutable Strings
* Command Line Arguments
  + This is what the args[] is in the main method
  + You can send arguments through the command line: java program arg1 arg2 arg3
* Bitwise Operators **[spend some time on this]**
  + Can be used on long, int, short, char, or byte
  + Called bitwise operators because they test, set, or shift individual bits that make up a value
  + & - AND
  + | - OR
  + ^ - Exclusive OR (XOR)
  + ~ - one’s complement (unary NOT)
  + Shift Operators
    - >> - shift right
      * Preserves the sign (+ / -)
      * All bits are shifted right one position, and 0 is brought in on the left
      * In terms of integers, it halves a value
    - >>> - unsigned shift right
      * Does not preserve sign
    - << - shift left
      * All bits are shifted left one position, and a 0 bit is brought in on the right
      * In terms of integers, it doubles a value
    - When shifting byte/short Java will auto promote these to int type when evaluating an expression
* The ? Operator (ternary operator)
  + Exp1 ? Exp2 : Exp3
    - Exp1 is a Boolean expression
    - Exp2 and Exp3 are of any type that is not void
    - If Exp1 is true then Exp2 is evaluated and becomes value of the entire ? expression
      * If false then Exp3

**Methods and Classes**

* Controlling access to class members
  + Public – outside can access
  + Private – only methods inside the class can access it
    - Restricting access is OO b/c it prevents object misuse
  + Protected
  + Default – when no modifier is used
    - Access limited to package
* How arguments are passed
  + Call-by-value
    - Copies value of an argument into the formal parameter of the subroutine.
    - Changes made to the value have no effect on the argument in the call
    - This happens when a primitive type is passed
  + Call-by-reference
    - A reference to an argument (not the value of the argument) is passed to the parameter
    - This reference is used to access the actual argument specified in the call
    - Objects are implicitly passed by reference
* Returning objects
* Method Overloading
  + Signature – the term for a method’s name + its parameters
* Overloading Constructors
* Recursion
  + When method calls itself, new local variables and parameters are allocated storage on the stack and the new method code is executed with these news variables
    - Care for stack overflow errors
  + Recursive is usually slower than iterative equivalents
  + Some algorithms are more clearly implemented using recursion
* Understanding Static
  + Use when defining a class member that will be used independently of any object of that class
  + Static variables are essentially global variables
  + All instances of the class share the same static variable
  + Static method – called thru its class name without any object of that class being created.
    - Restrictions:
      * Can directly call only other static methods
      * Can directly access only static data
      * Do not have a this reference
  + Static blocks
    - Executed when a class is first loaded. It is executed before the class can be used for any other purpose