**CODEACADEMY NOTES:**

Boolean operator precedence:

* ! evaluated first
* && evaluated second
* || evaluated third

Expressions in parenthesis evaluated first

* boolean riddle = !(1 < 8 && (5 > 2 || 3 < 5));
  + (5 > 2 || 3 < 5 )) = true
  + 1 < 8 = true
  + !true = false

Ternary conditional statement

* Composes of three parts
  + Boolean expression
  + Single statement that executes if true
  + Statement that executes if false
* Boolean expression ? ‘true statement’ : ‘false statement’

Object orientated processing:

* Class
  + Set of instructions that describe how a data structure should behaviour
* Constructor
  + Sets information about the class
  + Parameters
    - Specify what you want
    - Allow data types to be created with specified attributes
    - The value of the parameter will equal the value specified when we first use the constructor
* Instance variables
  + Specific details included in the class

The main method

* When the programme runs, the main method is executed
* Objects
  + To use a class we must create an instance of class – this is known as an object
* Methods
  + Pre-defined set of instructions
  + Declared within a class
  + We can call a method using variableName.methodName();
  + Can also add parameters to methods
* Void keyword
  + No value should be returned by the method – if we do want it to return a value we can specify the return type

Inheritance

* Can inherit behaviour from another class
* Use extends key word – indicates that the class inherited the behaviour of the defined class

For Loop

* Repeatedly runs a block of code until the specific condition is met
  + Initialisation
  + Test condition/Boolean expression – loop runs until the expression evaluates to false
  + Increment

ArrayList

* Stores a list of data of a specified type
* Create a new array list:
  + ArrayList<type> listName = new ArrayList<type>();
* Add items
  + listName.add();
* Access elements via index
  + listName.get(0)
    - this retrieves the item in the list at index 0
* insert method
  + listName.add(index to insert at, value to insert) e.g.
    - quizGrades.add(0, 100)
    - this adds the int 100 to index 0
* iterating over an array list
  + use a for loop consisting of an iteration, a test condition that will run as long as i is less than the size of the list, and an increment
  + to print out items in the list use:
    - System.out.println(listName.get(i));

For each list

* For (Integer grade : quizGrades){

System.out.println(grade);

}

* This can be read as “for each integer element called grade, in quizgrades, print out the value of grade”

HashMap

* Similar to a dictionary in that it contains a set of words and a definition for each word
* A hashmap contains a set of keys and a value for each key
* If you look up a word in a dictionary we can get a definition – similarly if you privde a HashMap with a key that exists you can retrieve the associated value
* Declaring a HashMap
  + HashMap<String, Integer> hashMapName = new HashMap<String, Integer>();
  + This will create a new map called hashMapName, will store keys of type string and values of type integer
* Filling the HashMap
  + Use the put method
  + E.g. hashMapName.put(“String”, Integer)
* Access data in an array list
  + To access a value, you have to specify the key
  + E.g. System.out.println(hashMapName.get(“String”);
* Iterating over a hashmap
  + Accessing length of a HashMap
    - hashMapName.size()
  + keySet method = returns a list of keys
* nextInt always runs between 0-high (high being top boundary number) to get between 1 and high use code (nextInt(high-low+1)+low (where low is 1)