**Variables**

* Defining variable: age = 26;
* These can be done on the same line by using int age = 26;
* Variables cannot be declared twice inside the same method
* To use variables declared outside the main method you must declare it as static
* Variable names cannot start with numbers, however they can start with $ and \_

**Data types**

* Primitive types store values
* Integer types:
  + Byte = -128 to 127
  + Short = -32,768 to 32,767
  + Int = -2147483648 to 2147483647
  + Long = -9223372036854775808 to 9223372036854775807
* Reference types refer to a stored value location
* Decimal types:
  + Double = 4.9E-324 to 1.7977
  + Float = 1.4E-45 to 3.403
* Boolean = true/false
* Characters = any characters
* Conversion can be implicit or explicit:
  + When compiler converts a smaller type of data like an integer to a larger type such as a double automatically this is known as an implicit conversion, explicit is the opposite
* Operators are symbols that allow the ide to perform operations:
  + Addition +
  + Subtraction -
  + Multiplication \*
  + Division /
  + Modulo %
  + Equal to ==
  + Not equal to !=
  + Less than <
  + More than >
  + Less than or equal to <=
  + more than or equal to >=

**Strings**

* Whenever you create a new string java checks if the string already exists, if it does the jvm will reuse the previous string with a new name, the jvm will create a new object regardless of if the value already exists
* Strings are not primitive types they are object types in java
* Formatted strings are far easier and cleaner than using a string which contains a lot of +variable+
* The specifiers for format strings are:
  + %s string
  + %n new line
  + %f floating point
  + %d integer
  + %c character
  + %b Boolean
* To check the length of a string including spaces you use: System.out.println(\*variablename\*.length());
* To check if a string is empty you use: System.out.println(\*variablename\*.isEmpty ());
* To convert a string to uppercase you use: System.*out*.println(\*variablename\*.toUpperCase());
* To convert a string to lowercase you use: System.*out*.println(\*variablename\*.toLowerCase());
* To compare two strings ignoring their caseyou use System.*out*.println(string1.equalsIgnoreCase(string2));
* To replace a word in a string you use System.*out*.println(string.replace(“word1”,”word2”));
* To check if a string contains a specific value you use System.*out*.println(string.contains(“”));

**User inputs**

* To take a user input you need to import a scanner using import java.util.Scanner;
* To take a user input you need to initialise a scanner such like: Scanner scanner = new Scanner(System.in) – this will take an imput from the console
* The scanner then needs to save the input to a variable using String name = scanner.nextLine(); this will save the next line the user inputs into the name variable
* There are many scanners for each data type, scanner.nextInt, scanner.nextDouble… etc but if you use anything but a nexline call you need to put an extra nextline call otherwise it will skip the next scan

**Conditional statements**

* Conditional statements are statements where a condition must be met for code to be executed
* Else if statements can be used when the first condition is not met
* Else statements can be used if no condition is met

**Switch cases**

* When using switch cases you must enter the variables you are checking the cases against
* You must use break after a case otherwise it will execute all of the following case statements afterwards
* A default statement is used so that if none of the cases are met the code has a default function that it can execute
* Switch case statements are much faster than if/else statements so they are always more efficient to use

**Arrays**

* Arrays always start with the data type followed by the name then followed by new \*datatype\*[array size]
* The first index is always a 0
* To print an entire array you must convert the array into a string using the java.util.Arrays import and using the Arrays.toString() function
* You can declare an entire array in one line using \*datatype\* array[] = {};
* To change any part of the array just use array[\*location\*] = ‘’;
* To sort an array you use Arrays.sort(array);
* To sort specific parts of an array you must define the starting index and ending index
* To search for a certain value you use the array binary search method (so the array must be sorted)
* If a search fails to find a value the output will be negative
* You can use arrays.fill to fill an entire array with any value/character, this can also use a start/end index to change certain values
* Arrays in java are reference types so whenever you make a copy of an array it doesn’t actually create a copy it just creates a copy of the pointer, to create an actual copy you use Arrays.copyOf
* By using copyOfRange you can use startingindex and endingindex
* To compare arrays you must use Arrays.equals(arrayname,arrayname2)

**Loops**

* Every single for loop has 4 parts, initialisation, condition, loop body and increaser
* You can use loops to increment through arrays
* You can print multiplication tables uses nested for loops
* Nested for loops are for loops contained within for loops
* While loops only have one condition and the loop body so the initialisation must be done outside the loop
* The difference between a do while loop and a while loop is the fact that the loop body is executed and then the condition changes, while the while loop is the other way around

**ArrayLists**

* To use array lists you must first import java.util.ArrayList
* ArrayList<Integer> number = new ArrayList<Integer>(); 🡨 This is an empty Array List of integers that is called number
* Integer is the reference form type of the basic form int, this is the same with the full names of other data types
* The get method takes an integer that will be the index of the value you are printing
* If you pass a reference type and a primitive type into a get method it will remove the value
* To sort an arraylist you use arrayname.sort(Comparator.*naturalOrder*()), the comparator must be imported for this to work
* The order of the list can be sorted and reversed by using Comparator.reverseOrder()
* To find out the size of an array list you use \*arrayname\*.size
* To check if an array list contains a certain value or not you use \*arrayname\*.contains(Integer.valueOf()) which will return true or false
* To check if it is empty you use \*arrayname\*.isEmpty() which will return true or false
* Numbers.forEach applies the action held within it’s curly braces to each number in the array list

**HashMaps**

* If you try to get a key that doesn’t exist it will return null, however if you set a default value it will return whatever the default is
* Using reference types will not effect the result using containvalue
* isEmpty can be used to check if it’s empty

**OOP**

* variables at a class level in oop are called properties and if they are public they can be accessed outside their class