**JAVA BASICS**

**\*\*\*\*\*\*\*\*\*\*\***

* Popular Code editors
* **Anatomy of a JAVA Program**  
   \* Function = smallest building block

= a block of code that performs a task (i.e. button on a tv remote, have a function to send email to somebody or validating users input )

* + **Coding a function in Java**  
    specify the return type of the function

Some functions have a return type of a value of a number, a date or nothing (void)  
After the return type we have the name of our function followed by a parenthesis. The name should have a proper descriptive name that clearly identifies the purpose of this function. Note that the name starts with a small letter.

In this parenthesis we would add the parameters of this function. Parameters are used to pass values to our function.

After the parenthesis we will have curly braces and which we will have the actual JAVA code. The first curly brace will be on the line where we define our function.

Every JAVA program should at least have one function and that function is called the **main function.**

**EXAMPLE OF A FUNCTION**

void main ( ) {

….

}

**Main** is the **entry point** to our programs therefore whenever we execute a Java program the main function gets called andthe code inside this function gets executed.

Function cannot exist on their own and therefore need to belong within a CLASS  
CLASS = A container for one or more related functions  
Classes are used to organise our code, think of how items are organised at a supermarket ……each section has related products  
  
Each program should at least have one class that contains the main function and within this main class we can have **functions**

**EXAMPLE**class Main {

void main ( ) {

…..

}

}

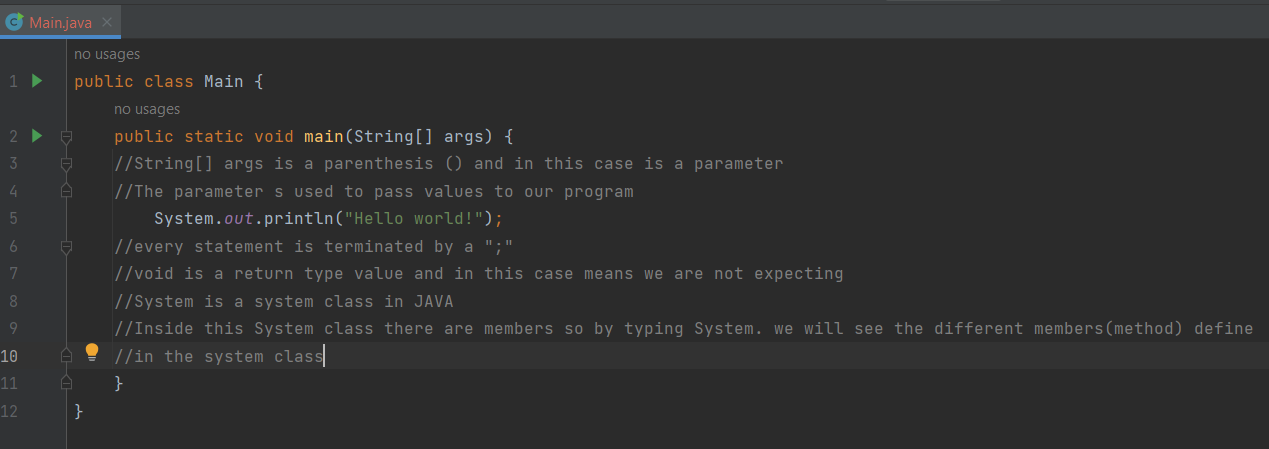
We refer to the functions within the main class as methods  
Therefore a method is a function that is part of a class  
When outside of the class it is called a function  
  
In JAVA all these classes and methods should have an **access modifier**  
An access modifier is a special keyword that determines if other classes and methods in this program can access the classes and methods  
Examples of Access Modifiers are Public(accessible from other parts of the program) or Private  
Access Modifiers are put in front of our class and method  
NOTE: The main method of the program should always be STATIC  
  
**EXAMPLE**   
  
public class Main {

public void main ( ) {

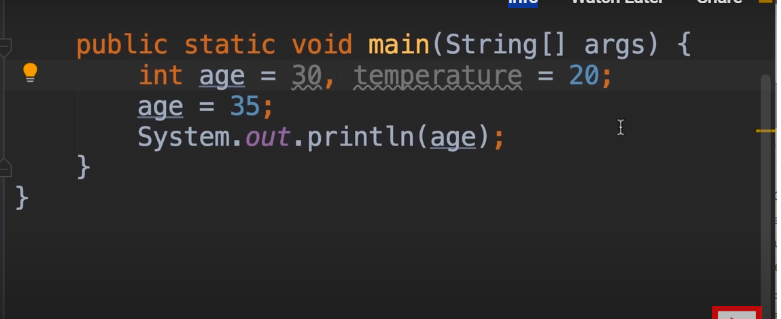
…….

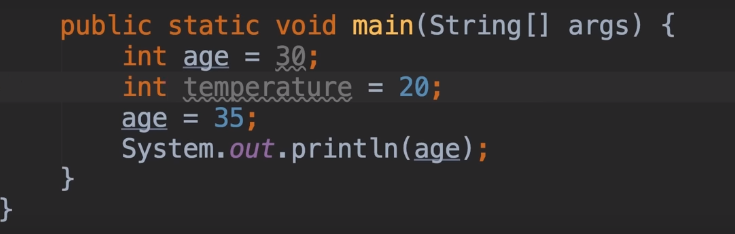
}  
}

This is the basic structure of a JAVA program

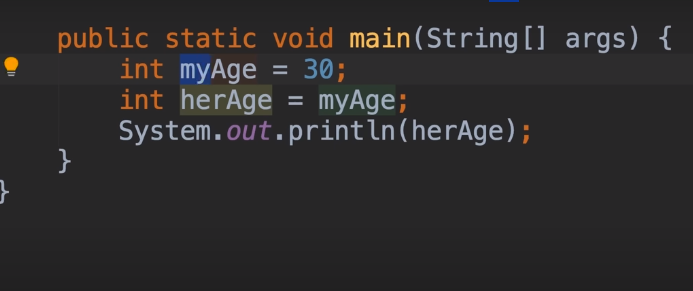
To name classes we use the **P**ascal**C**ase meaning  
 the first letter of every word should be upper case  
 To name methods we the **c**amel**C**ase  
  
Example 2  


**VARIABLES**

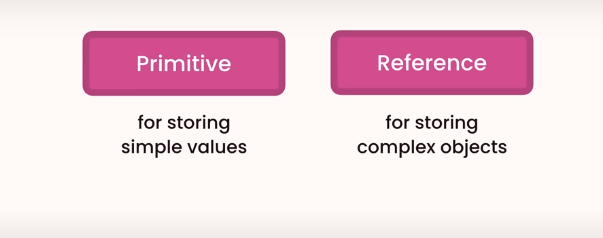
We use variables to temporarily store data in the computer’s memory   
We always initialise our variables before storing them  
Using a comma we can declare multiple variables on the same line but this is dirty coding (Fig 1) so it is better to declare one variable in each line (Fig 2)  
  
FIG1  
   
  
FIG 2

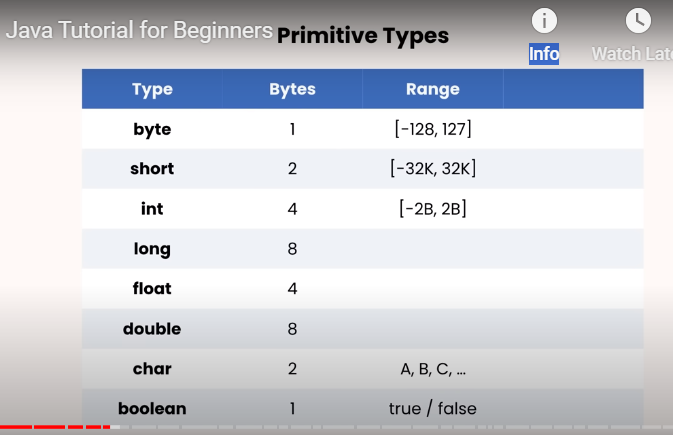


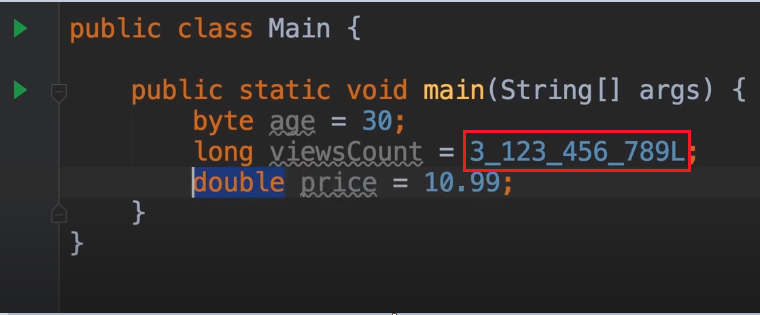
We declare a variable as with the above example where  
**Int** (integer) is a **type** of variable(whole numbers), age is the **identifier (name or label)** and the equal sign is **assignment variable**In the above example we are **initialising** the variable  
Also remember that we need a ”;” at the end of each statement

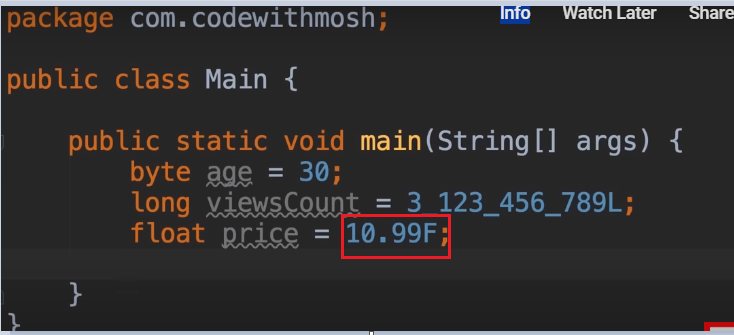
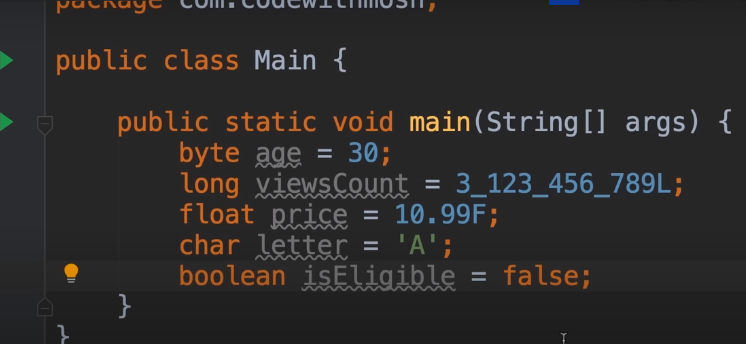
We use camel casing when naming in our methods. See example   
Fig3  


**Types in JAVA**

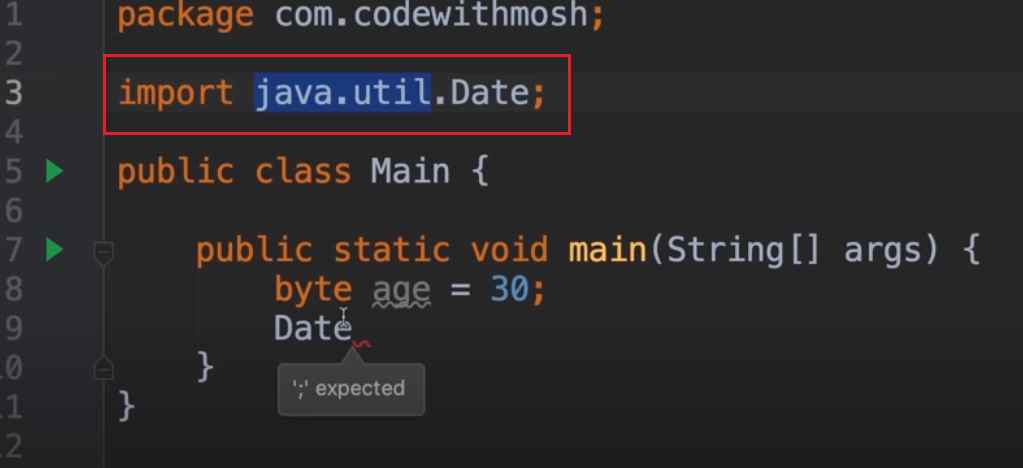
Here are two types in JAVA mainly Primitive and Reference Types  


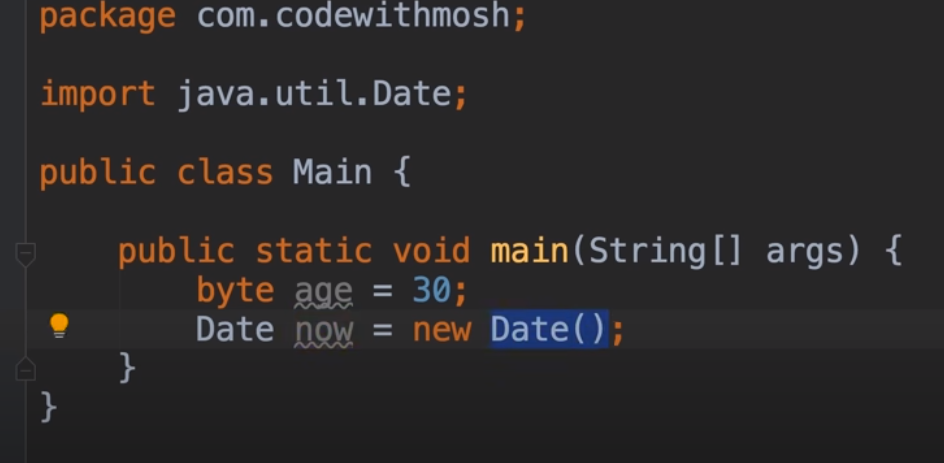
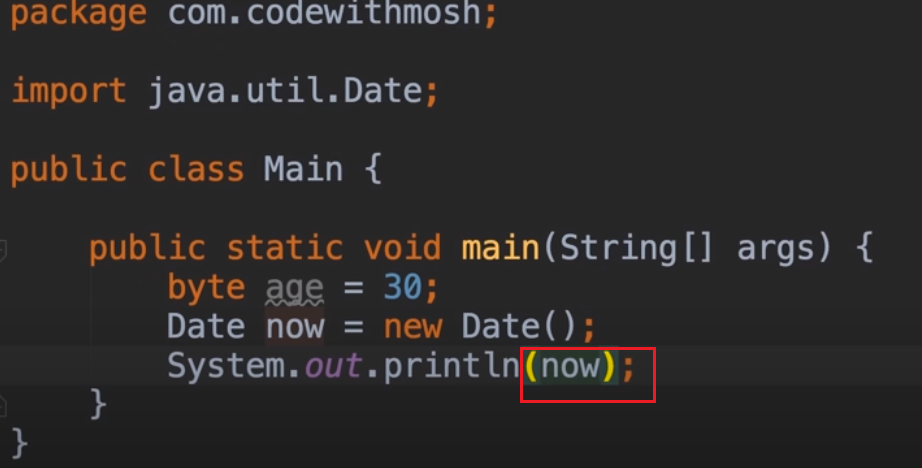
PRIMITIVE Types  
  
  
**Byte,short,int,long** = these are for storing whole numbers with no decimal points  
**Float and double** are for storing numbers with decimal points  
  
So from previous example we realise that to store someone’s age we do not need the **Int** type as that is used to store as big as 2 billion whereas we can use byte which can store up to the **numbers 128.**Instead we can use **int** to store the number of times a youtube video has been viewed  
Good practice when naming variables is to use meaningful names i.e int viewsCount  
In JAVA we can use an underscore to separate numbers like how we would use commas in

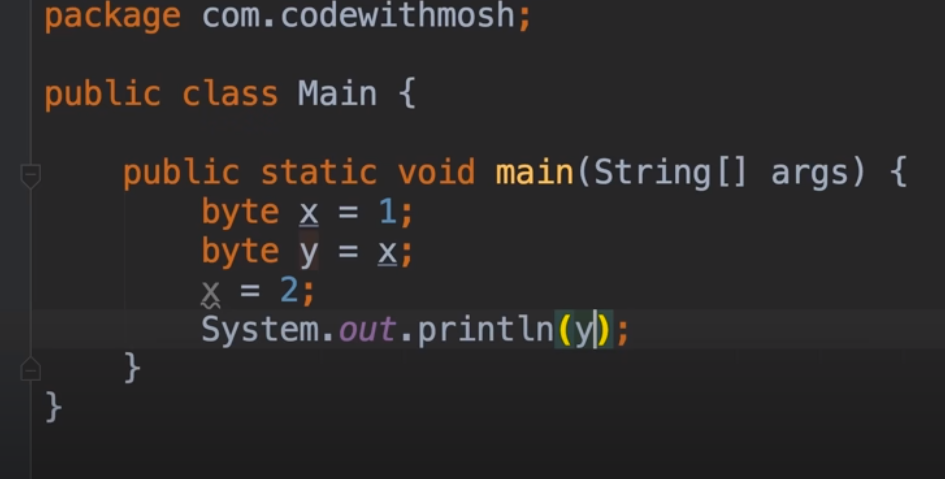
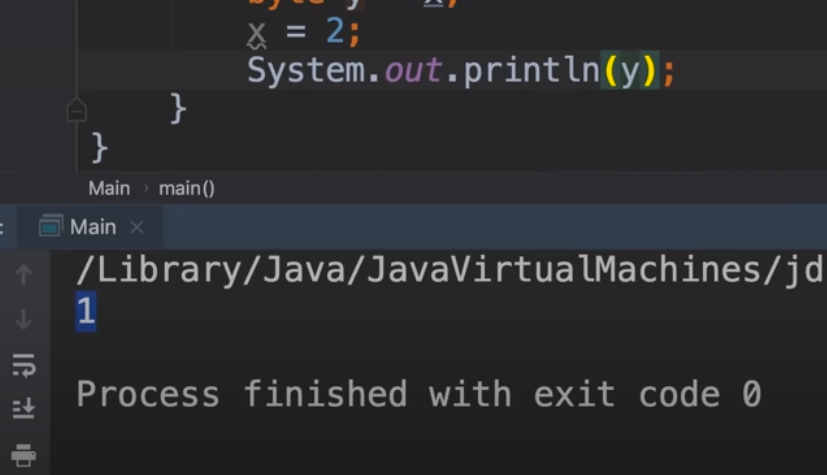
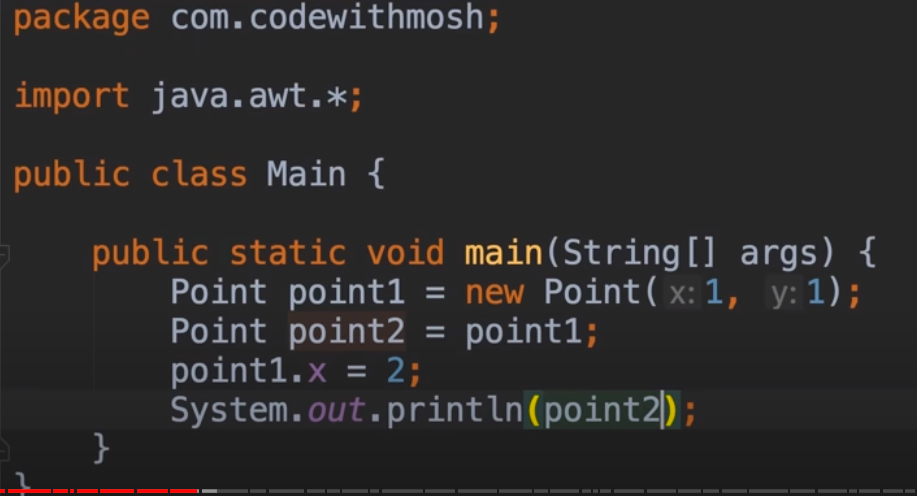
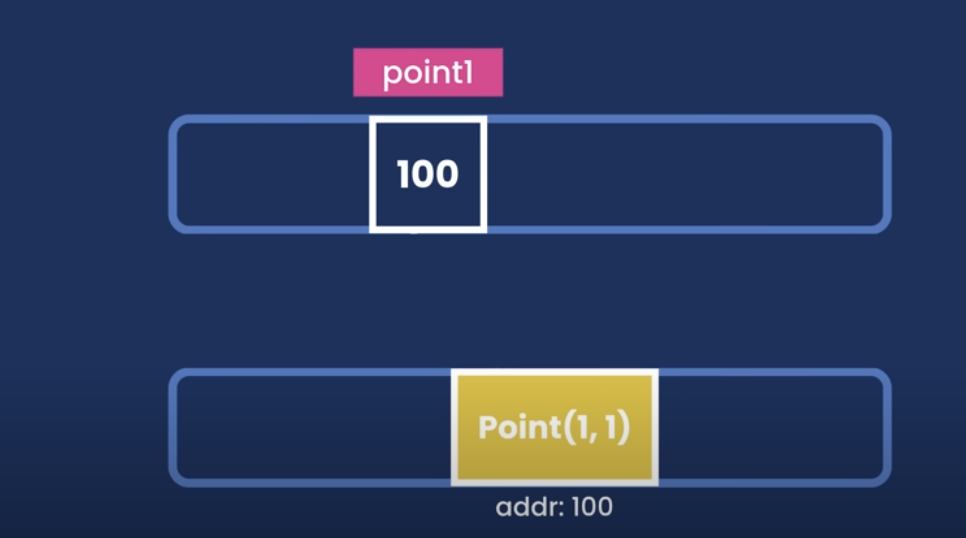
**NOTE:**  
When declaring a variable that is **long,** the compiler will see this as still an Integer so we need to put an “L” as the suffix of the defined number. Dame occurs with a **float** we need to add an F at the end. See below example  


Also note that when we define a **char field** we nee to use **single quotes** unlike string which uses double quotes  
  
Note that the word false for Boolean is a **reserved key word** as it is in orange so we cannot use it as a variable name

**REFERENCE TYPES**

  
  
When we use the date class,intellij automatically imports the date package, see below  
  
  
Note:  
We are import the date class that is in the java,util package  
When dealing with Reference Types we need to use the **new** operator **allocate memory** unlike when dealing with primitive types  
Difference between primitive type and reference is that with primitive type we do not need to allocate e memory as allocated and released by Java Runtime Environment as with reference we have to allocate memory

  
In the above example we can see that when defining the refence type we need to say Date variable type = NEW Date()  
The variable defined here is an instance of the Date class  
  
Note the short cut for System.out.println is SOUT then press TAB  
  
Since we want the value of our date object we do not put “ “ around now  
  
  
  
**Primitive type vs Reference Types(revisit this)**The other difference between the two types is memory management  
With primitive the values of x and y are at two **different memory locations** so they are completely independent of each other. If the value of X is changed then the value of Y is not affected

  
  
  
The following is an example of Reference type where we are using reference type of point or point object. Notice the import  
Note: Point1 and Point2 reference the same address and that address i.e 100 is stored upon initialisation    
The reference types store the reference to the object and not the value and so the two objects are not independent of each other  
