Declaration means creating a variable

int i

int is a type of variable

i is name of the variable

int arr[] this is also declaration

[] this is dimensions of array

Whenever we are declaring an array

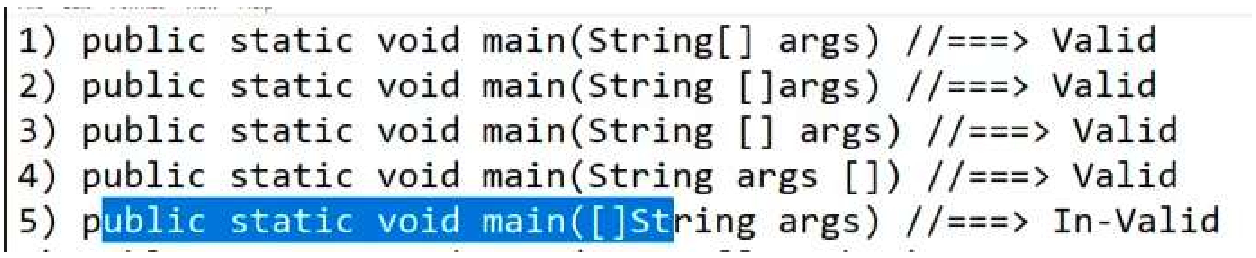
We can write with a space after the identifier (here identifier is arr means name) int arr []

We can write without a space after the identifier (here identifier is arr means name) int arr[]

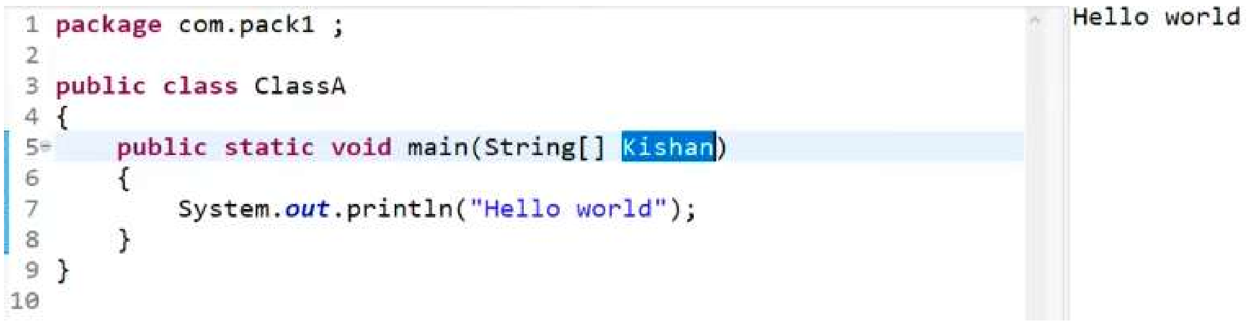
We can write without space or with space before the identifier int []arr or int [] arr

But we should never write a dimension of array ([]) before the data type of that array. ([] int arr this false).

Here arr is just a name we can go with any other



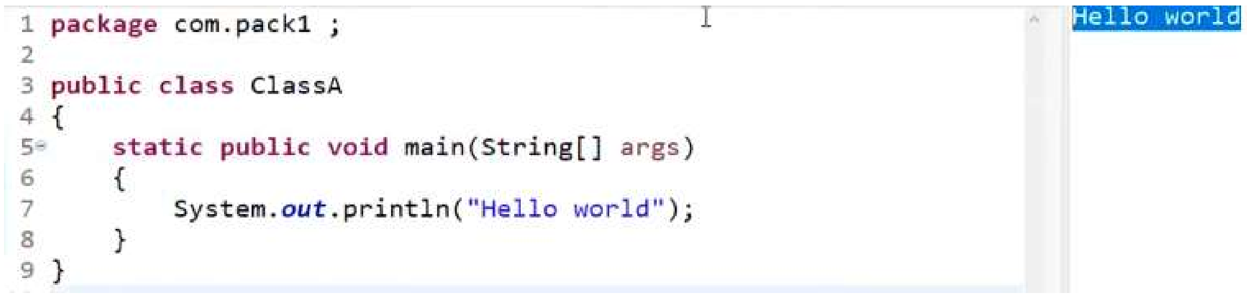
```
1) public static void main(String[] args) //===> Valid
2) public static void main(String []args) //===> Valid
3) public static void main(String [] args) //===> Valid
4) public static void main(String args []) //===> Valid
5) public static void main([]String args) //===> In-Valid
```


```
package com.pack1 ;

public class ClassA
{
    public static void main(String[] Kishan)
    {
        System.out.println("Hello world");
    }
}

```

```
Hello world
```

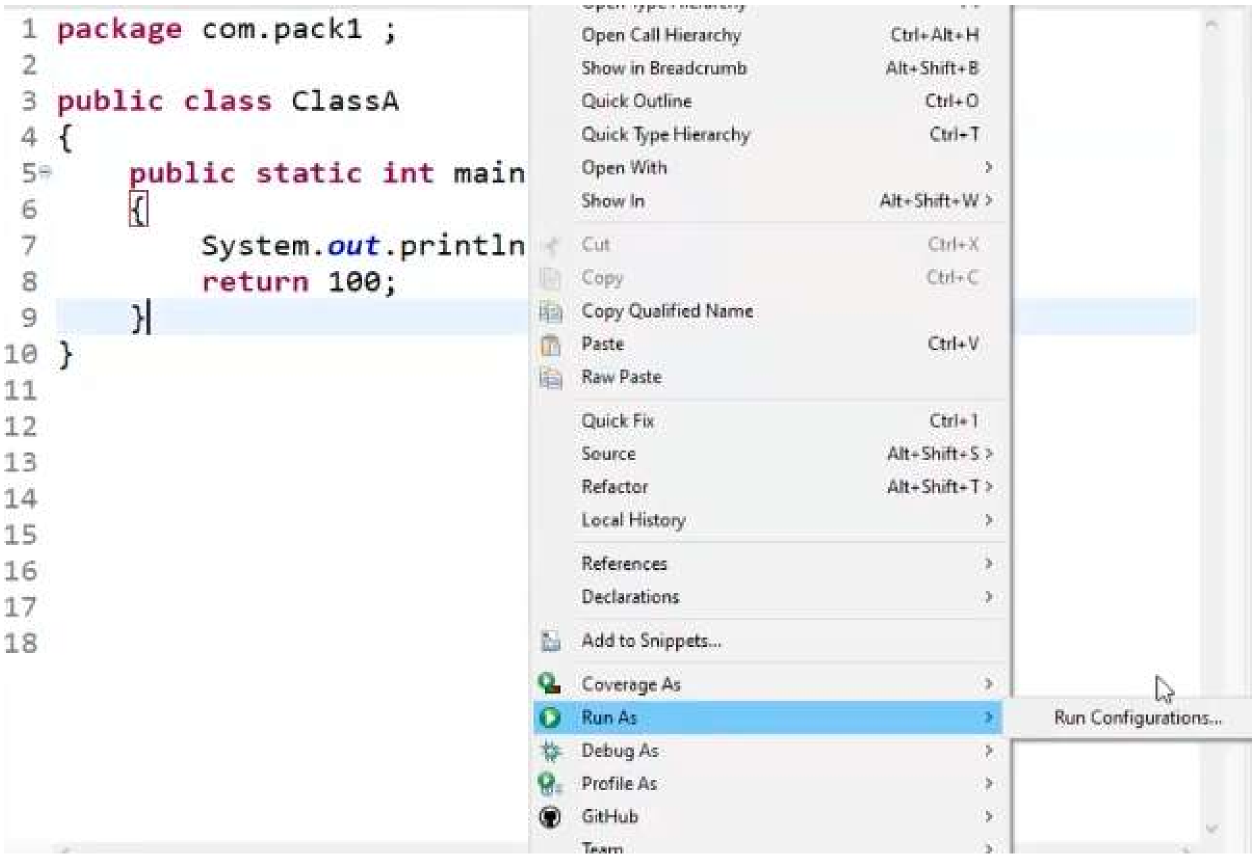
6) public static void main(String[] Kishan)//===> Valid

Always before the method name, we should have return type 100% this is mandatory.


```
package com.pack1 ;

public class ClassA
{
    static public void main(String[] args)
    {
        System.out.println("Hello world");
    }
}
```

```
Hello world
```

7) static public void main(String[] args)//===> Valid



```java
package com.pack1 ;

public class ClassA
{
    public static int main
    {
        System.out.println
        return 100;
    }
}
```

Invalid because of return type, the main method should not return anything.

```
8) public static int main(String[] args) //===> In-Valid
```

In java we have 4 superpowers

1.encapsulation

2.inheritance

3.polymorphism

1). Compile time polymorphism, it is most known as method overloading

2). runtime polymorphism, it is mostly known as method overriding.

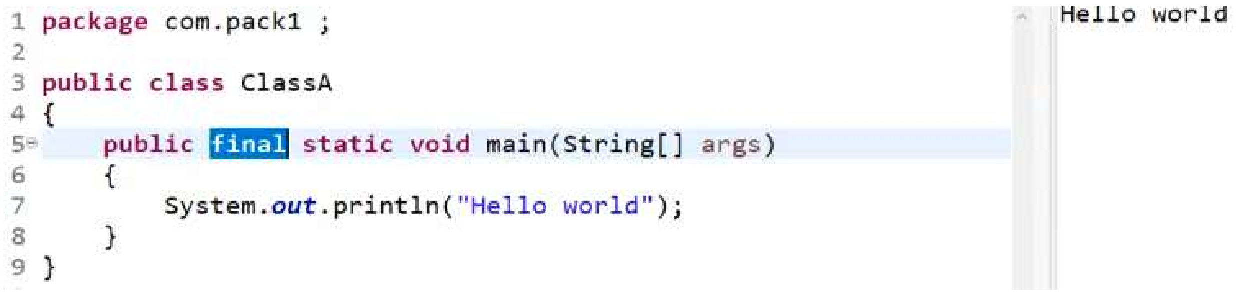
4.abstruction

Whenever you are declaring any method as static, statics method can't be overridden.

static is a key word

final is also a key word, if we want to stop the overriding then we can make the method as final.

The only use of the final method is we cannot override.



```
package com.pack1 ;

public class ClassA
{
    public final static void main(String[] args)
    {
        System.out.println("Hello world");
    }
}
```

```
Hello world
```

Whatever the final method can do, by default the static method is having that feature.

No need to declare a static method as final.

```
9) public final static void main(String[] args)//===> Valid
```

```
10) Public static void main(String[] args)//===> In-Valid
```

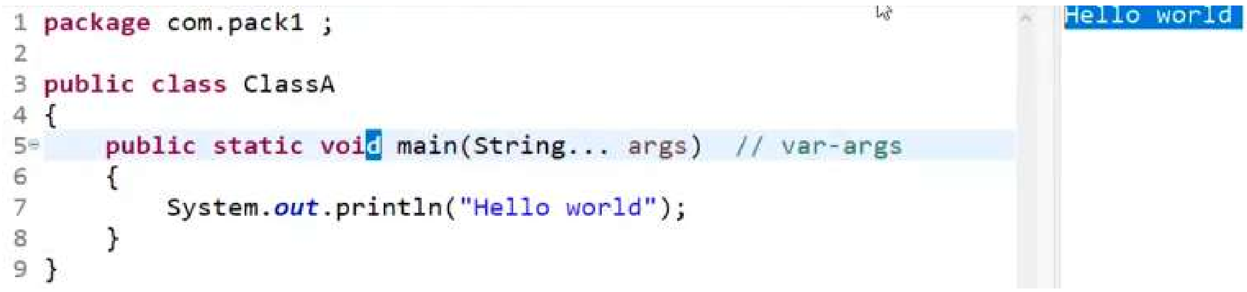
Because capital P in Public is invalid, every java keyword will start with a small letter.

```
11) final public static void main(String[] args)//===> Valid
12) Final public static void main(String[] args)//===> In-Valid
```

Because of capital f in final. Key words always start with small letters.

String... args this is a variable argument most known as var-args method.

Variable arguments will be working as an alternate for array.



```java
package com.pack1 ;

public class ClassA
{
    public static void main(String... args)  // var-args
    {
        System.out.println("Hello world");
    }
}
```

```
Hello world
```

```
13) public static void main(String… args)//===> Valid
```

Note after String three dots only valid (String ... args)

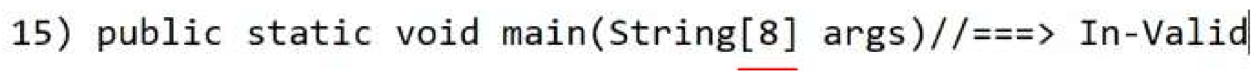
```
14) public static void mian(String[] args)//===> In-Valid
```

Invalid because of the main spelling wrong.

Initialization means passing some values into an array.

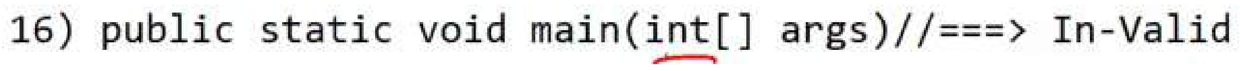
Instantiation means reserving some memory location so we can pass the value later.

We should never write the size of an array, at the time of its declaration.

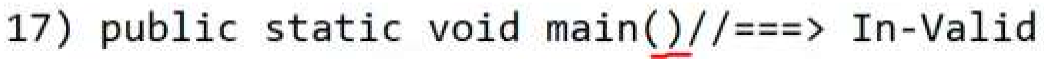


```
15) public static void main(String[8] args)//===> In-Valid
```

The parameter of the main method should be string array (String [] args)

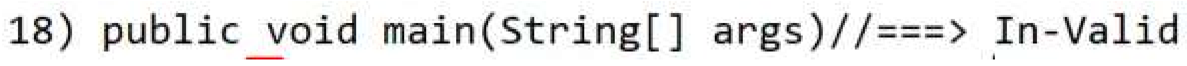


```
16) public static void main(int[] args)//===> In-Valid
```



```
17) public static void main()//===> In-Valid
```

Invalid because there is no parameter.



```
18) public void main(String[] args)//===> In-Valid
```

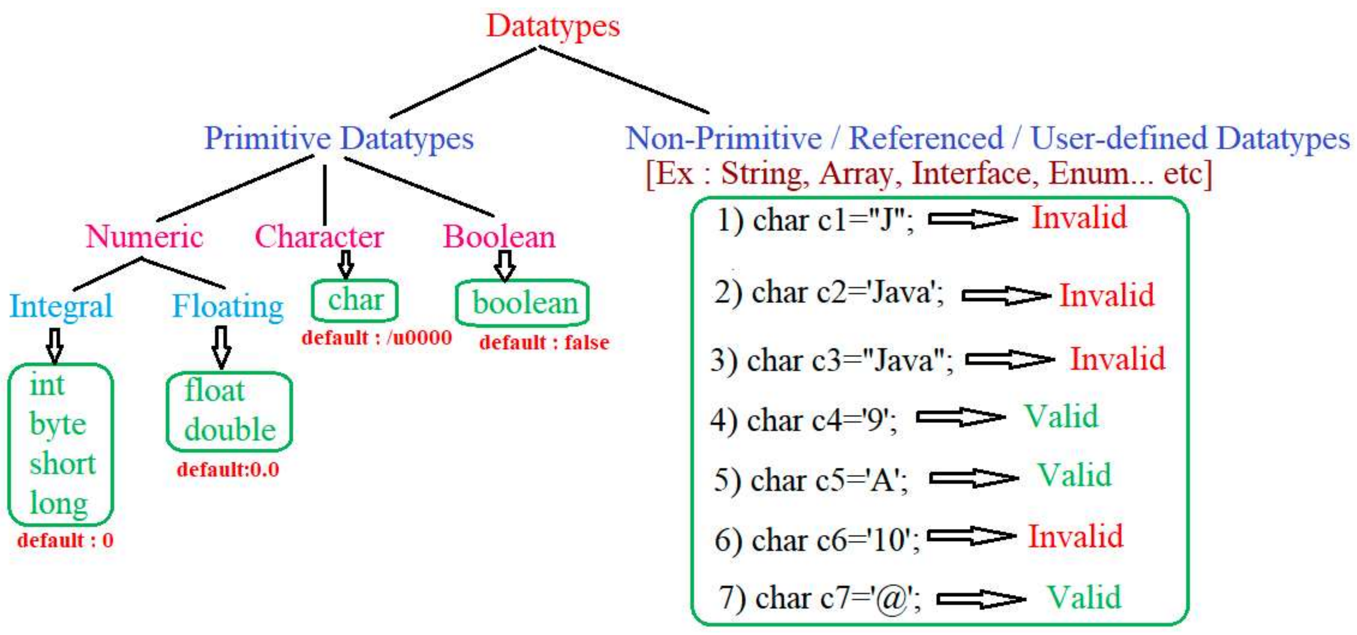
invalid because there is no static key word.

```
19) public static void Main(String[] args) //===> In-Valid
```

invalid because there is capital M in main, main method is a predefined method.

Data types

- Data types specify size and type of values that can be stored in an identifies.
- Data types represent what type of data to be allowed and how much memory is required to hold the data.
- In java we have 2 types of data types
  1. Primitive data types
  2. Non primitive or referenced or user-defined data types.



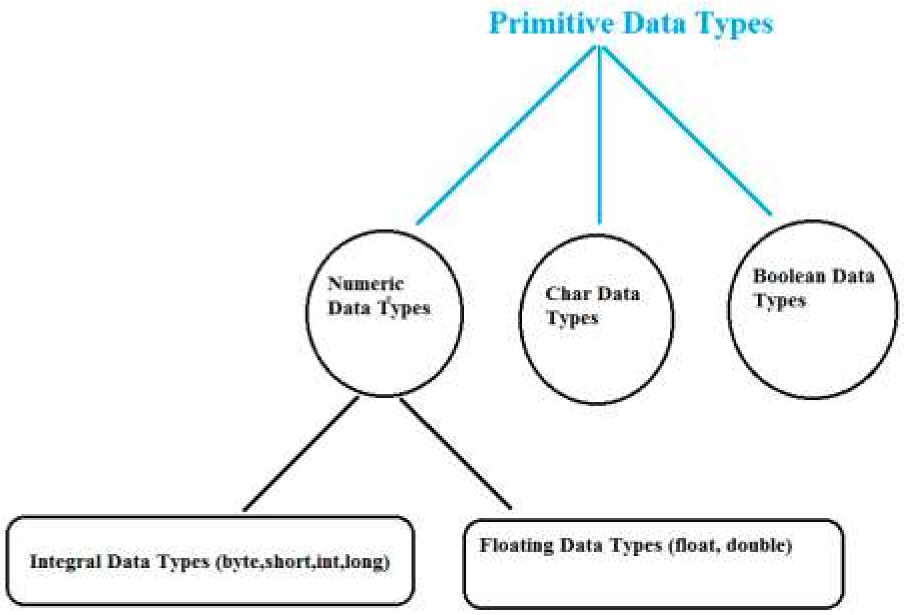
Java Data Types

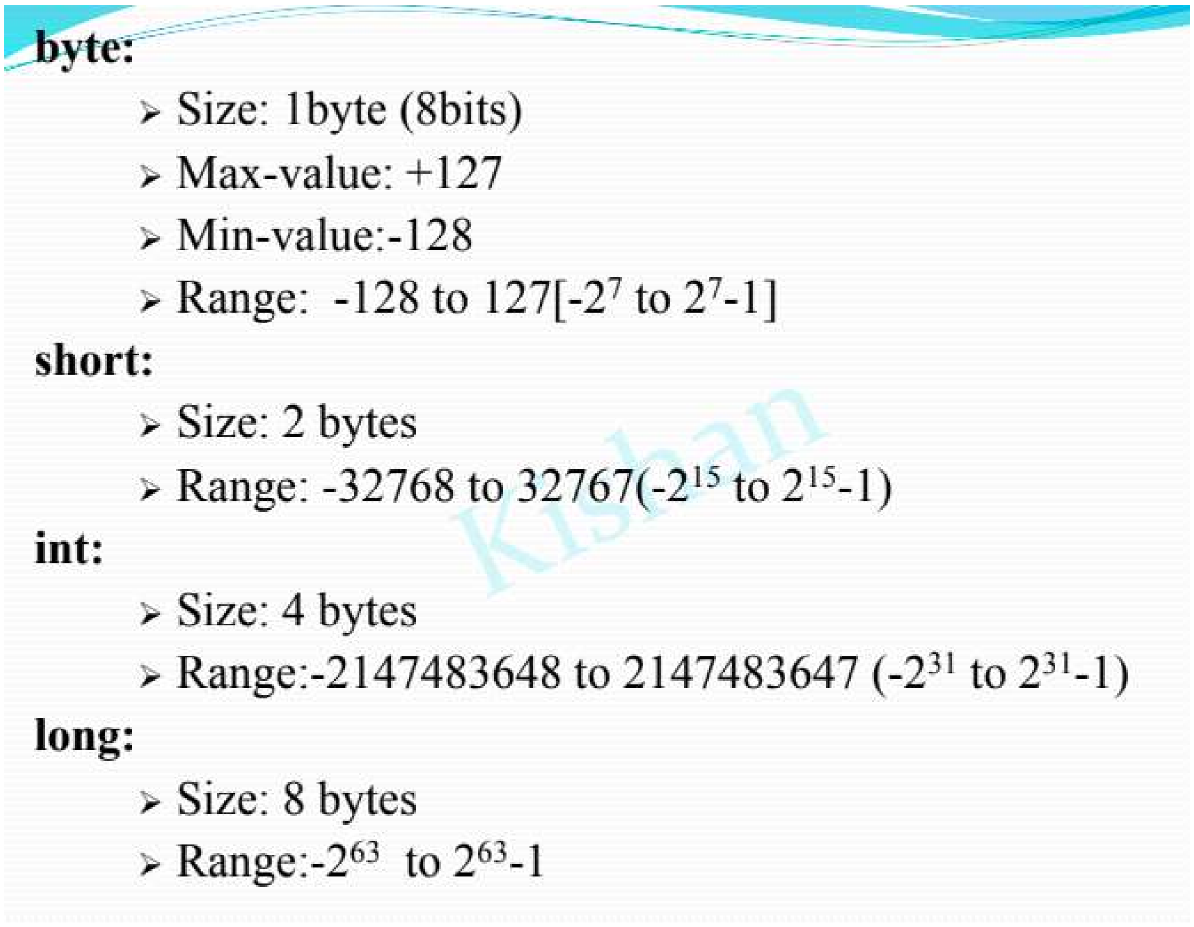
- Data type specifies the size and type of values that can be stored in an identifier
- They are usefull to represent how much memory is required to hold the data.
- Represents what type of data to be allowed.
- Java data types are classified in to 2 types

--->Primitive Data types

---> User Defined Data types (Reference)

(String, Array, class, abstract class, interface…etc)


Primitive Data Types
Numeric Data Types
Char Data Types
Boolean Data Types
Integral Data Types (byte,short,int,long)
Floating Data Types (float, double)



byte:

- Size: 1byte (8bits)
- Max-value: +127
- Min-value:-128
- Range: -128 to 127[-2^7 to 2^7-1]

short:

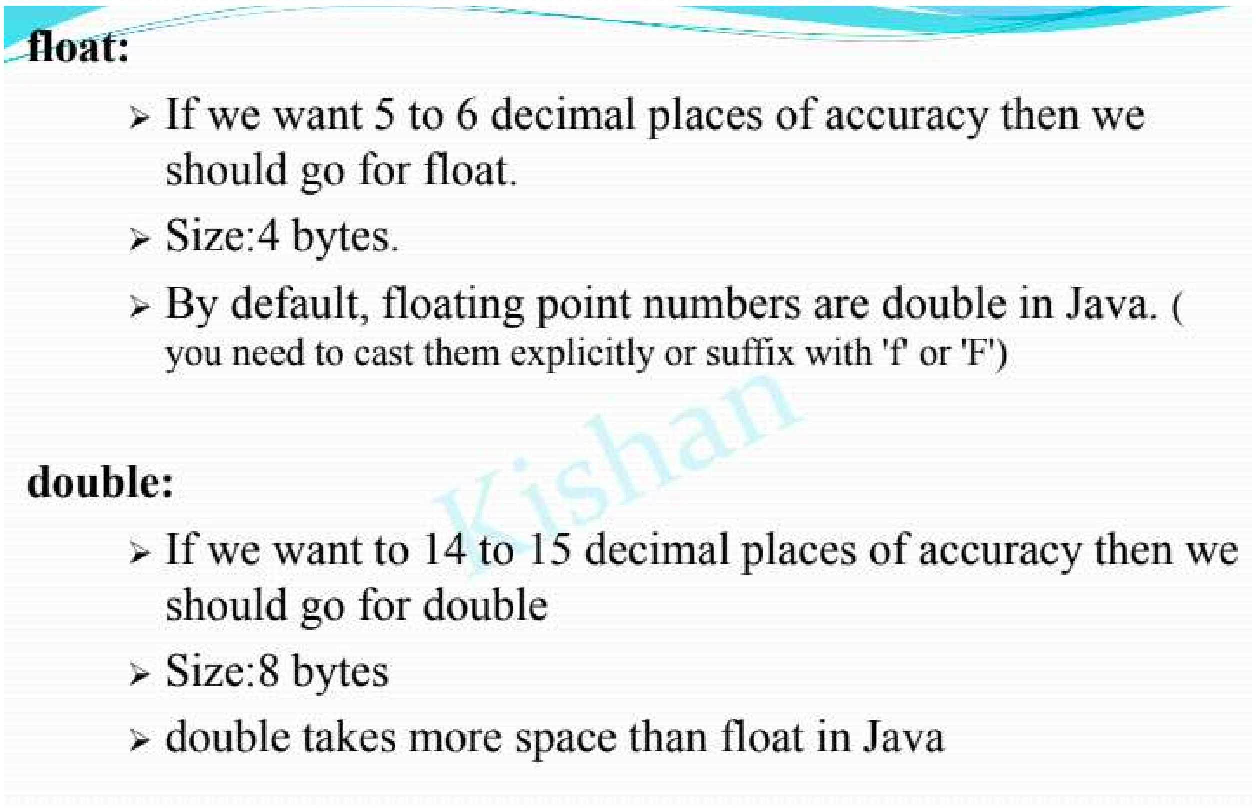
- Size: 2 bytes
- Range: -32768 to 32767(-2^{15} to 2^{15}-1)

int:

- Size: 4 bytes
- Range:-2147483648 to 2147483647 (-2^{31} to 2^{31}-1)

long:

- Size: 8 bytes
- Range:-2^{63} to 2^{63}-1

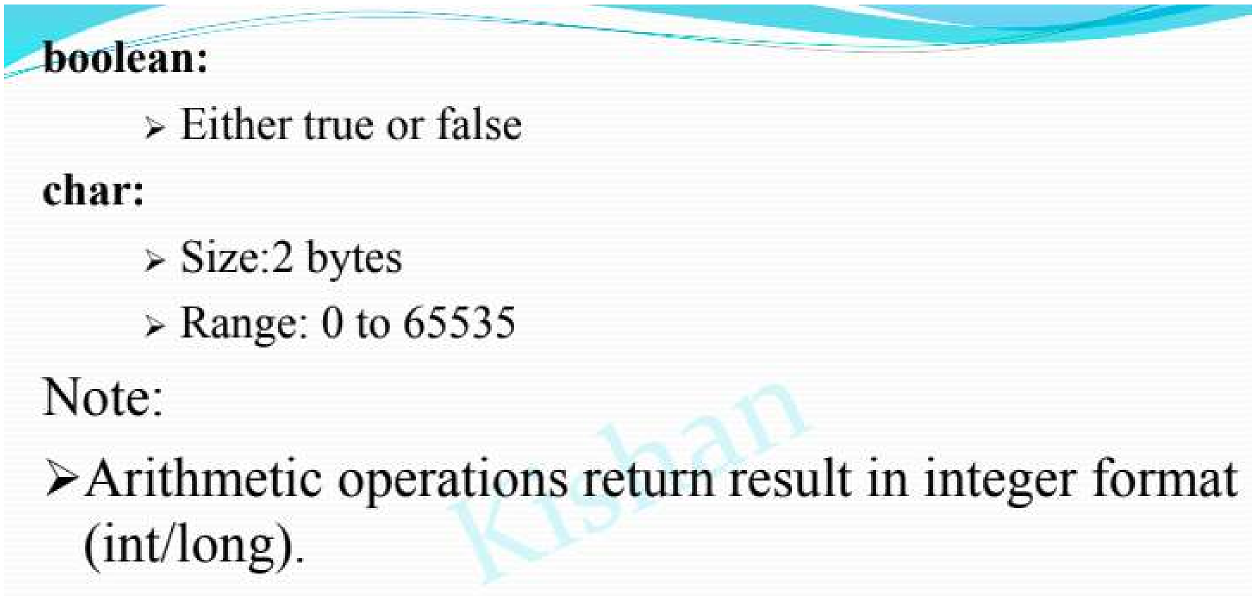


float:

- If we want 5 to 6 decimal places of accuracy then we should go for float.
- Size:4 bytes.
- By default, floating point numbers are double in Java. (you need to cast them explicitly or suffix with 'f' or 'F')

double:

- If we want to 14 to 15 decimal places of accuracy then we should go for double
- Size:8 bytes
- double takes more space than float in Java



boolean:

- Either true or false

char:

- Size:2 bytes
- Range: 0 to 65535

Note:

- Arithmetic operations return result in integer format (int/long).

Char data type

Char data type will accept only one character.

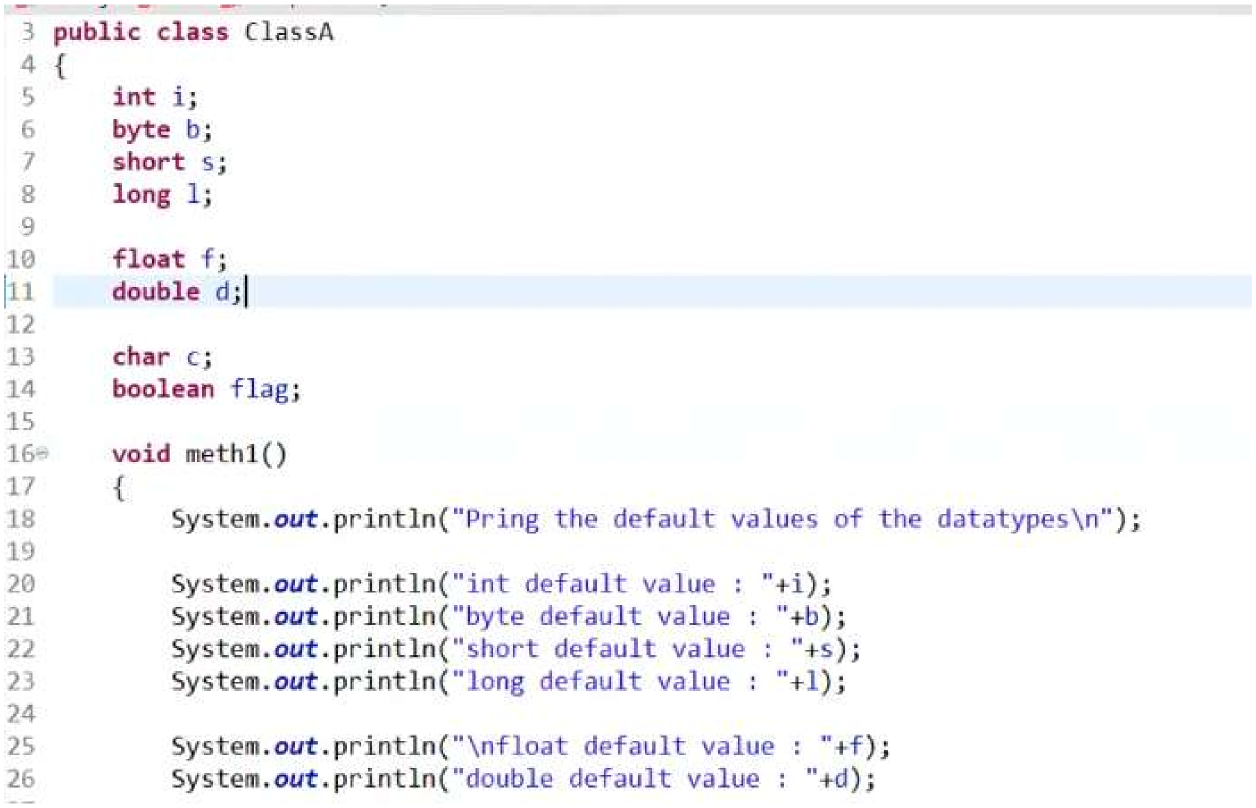
That single character can be any alphabet, number, special character, anything but only one.

Char data should be in single codes' '

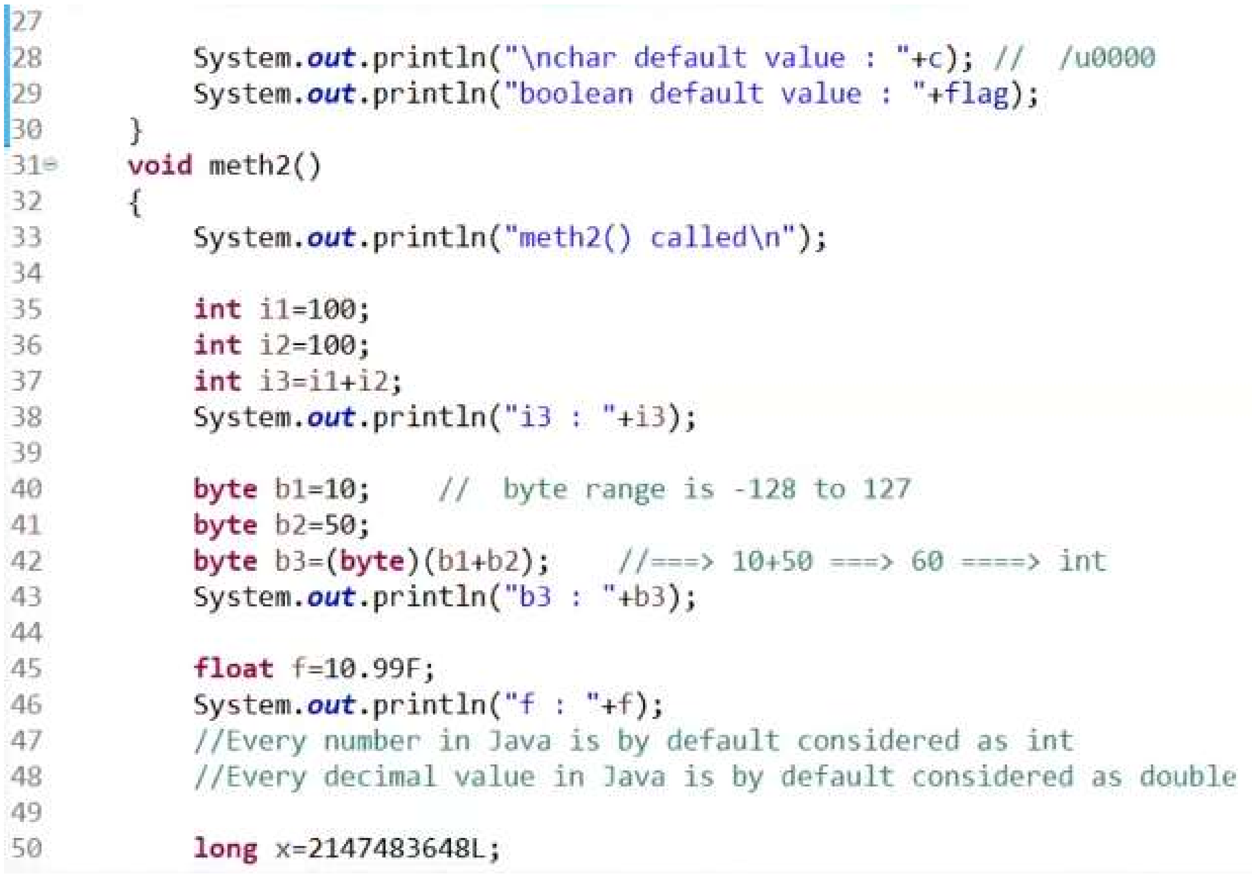
Default value for char is NPC (non-printable character) /u0000

In some laptops we can get ?, in some we get space, in some we can get square

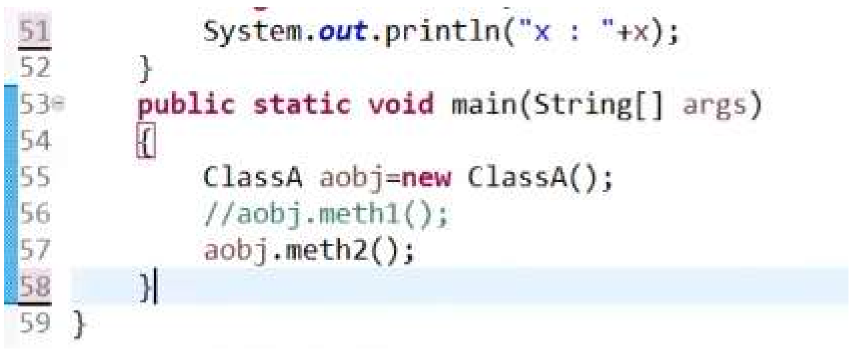
Arithmetical operations will be giving the result either in int or in long.



```java
public class ClassA
{
    int i;
    byte b;
    short s;
    long l;

    float f;
    double d;

    char c;
    boolean flag;

    void meth1()
    {
        System.out.println("Pring the default values of the datatypes\n");

        System.out.println("int default value : "+i);
        System.out.println("byte default value : "+b);
        System.out.println("short default value : "+s);
        System.out.println("long default value : "+l);

        System.out.println("\nfloat default value : "+f);
        System.out.println("double default value : "+d);
```



```java

        System.out.println("\nchar default value : "+c); //  /u0000
        System.out.println("boolean default value : "+flag);
    }
    void meth2()
    {
        System.out.println("meth2() called\n");

        int i1=100;
        int i2=100;
        int i3=i1+i2;
        System.out.println("i3 : "+i3);

        byte b1=10;    //  byte range is -128 to 127
        byte b2=50;
        byte b3=(byte)(b1+b2);    //===> 10+50 ===> 60 ====> int
        System.out.println("b3 : "+b3);

        float f=10.99F;
        System.out.println("f : "+f);
        //Every number in Java is by default considered as int
        //Every decimal value in Java is by default considered as double

        long x=2147483648L;
```



```
        System.out.println("x : "+x);
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        //aobj.meth1();
        aobj.meth2();
    }
}
```