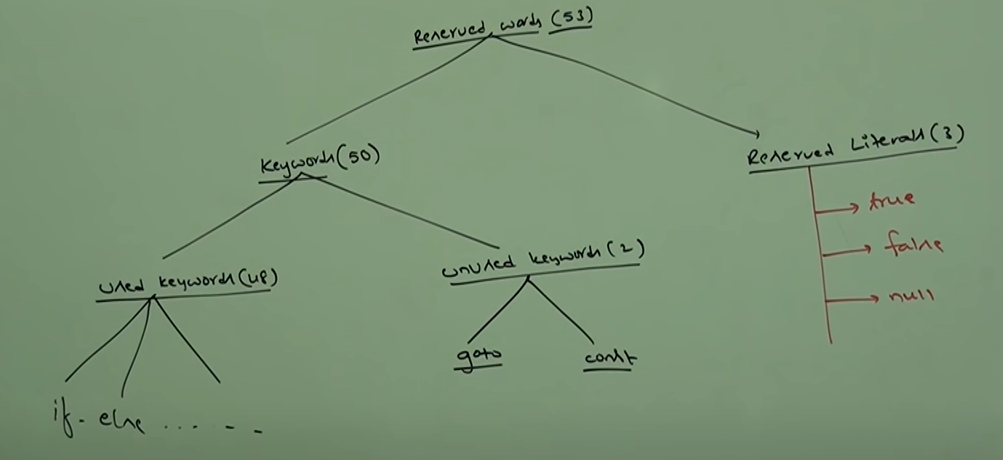
**Identifier** - A name in Java program is called identifier, which can be used for identification purposes. It can be a method name, class name, variable name or label name.   
Rules For defining a Java identifier –

* Allowed list of characters we can use as in identifier are (a-z, A-Z, 1-9, $, \_). Using characters other than this list of character we will get an compile time error
* identifier cannot start with a digit. E.g. – ‘name123’ is a valid Identifier but ‘123name’ is not a valid identifier.
* Java identifiers are case sensitive since Java language is treated as case sensitive language. So we can differentiate with respect to case.
* There is no length limit for Java identifier but it is not recommended to take lengthy identifiers.
* We cannot use reserve words as identifiers if we use it we will get compile time error.
* all predefined Java class names or method names can we use as identifiers but it is highly not recommended.

**Reserve Word** - In Java some words are reserved to represent some meaning our functionality search type of words are called reserve words there are 53 reserved words in Java.



\*In Java return type is mandatory if a method does not return anything we should declare it as void.

\*If we use keyword ‘goto’ or ‘const’ then we will get a compile time error.

\* null - Default value for object reference

\*We can use ‘enum’ to define a group of named constants.

\*In Java new keyword is there to create an object but no delete keyword is present to remove or destroy object because removing or destroy an object is taken care by Garbage Collector

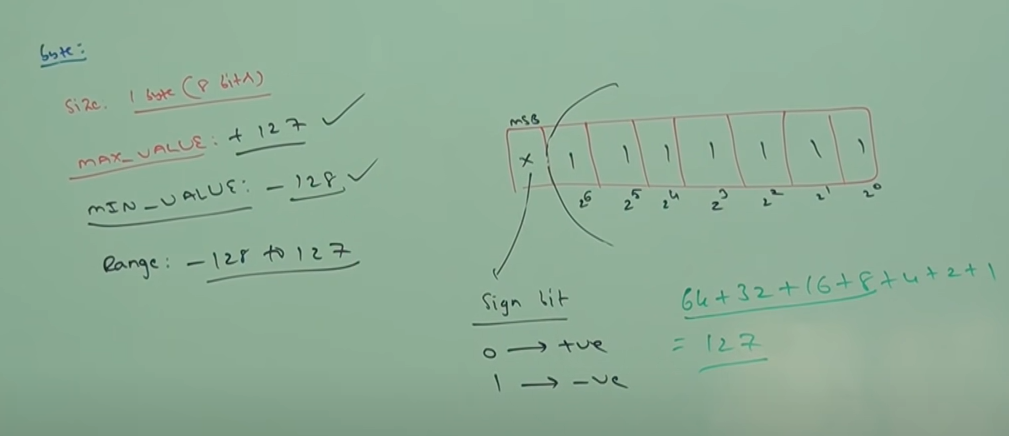
**DATA TYPES :-**

In Java every variable and every expression has some type. Each and every data type it's clearly defined .every assignment should be checked by compiler for type compatibility .  
Because of above reasons we can conclude Java is strongly typed programming language.

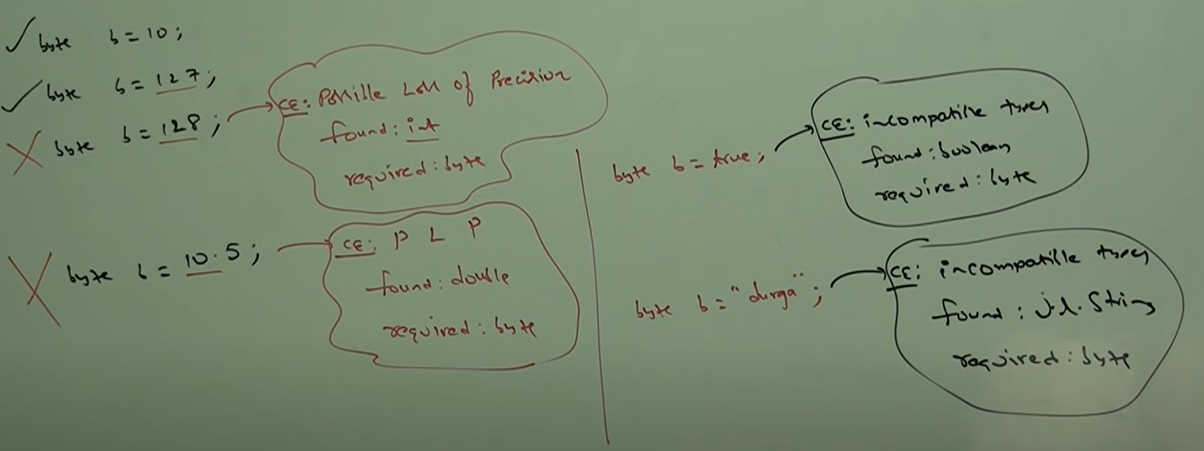
JAVA is not considered as a pure object oriented programming language because several OOP features are not supported by Java like operator overloading, multiple inheritance and dependency on primitive data type.

Integral data types (byte, short, int, long)

Byte 🡪

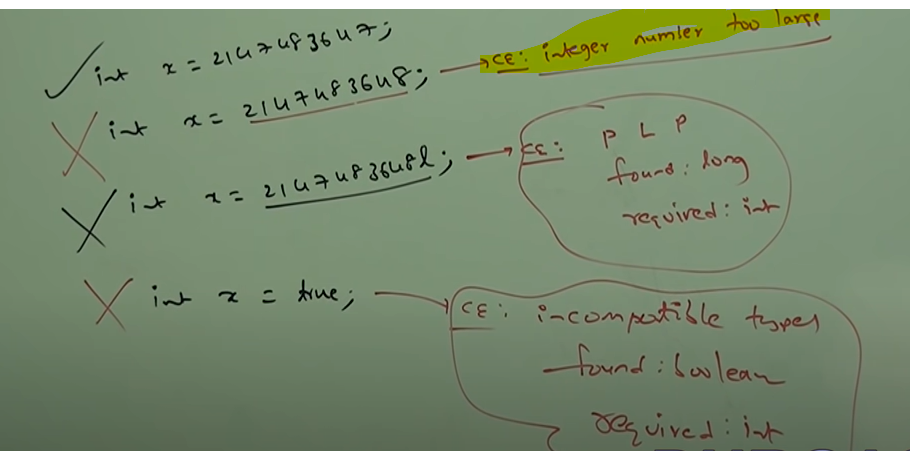


Byte is the best choice if we want to handle data in terms of streams either from file or from network (file supported form or network supported form is byte )



short 🡪 Size – 2 bytes (16 bits)

int 🡪 size – 4 bytes (32 bits)

We get the compile time error as ‘integer number too large’ because by default any integer number is taken as data type int

long 🡪 size – 8 bytes (64 bytes)

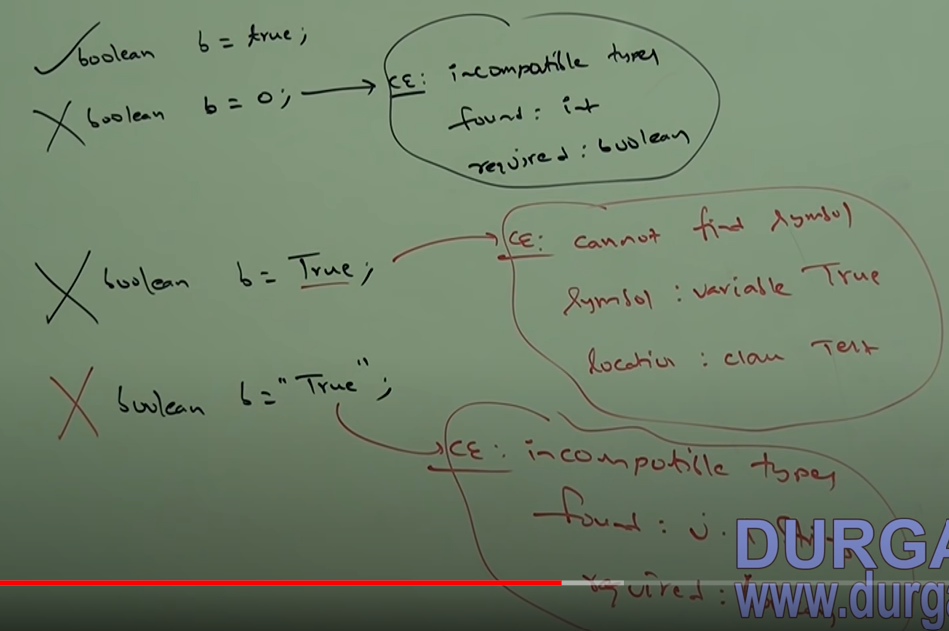
Floating Point data types (float & double) –

Double provides more precision than float, double provides precision up to 14 to 15 digits after decimal point whereas float provides precision up to 5 to 6 digits after decimal point.

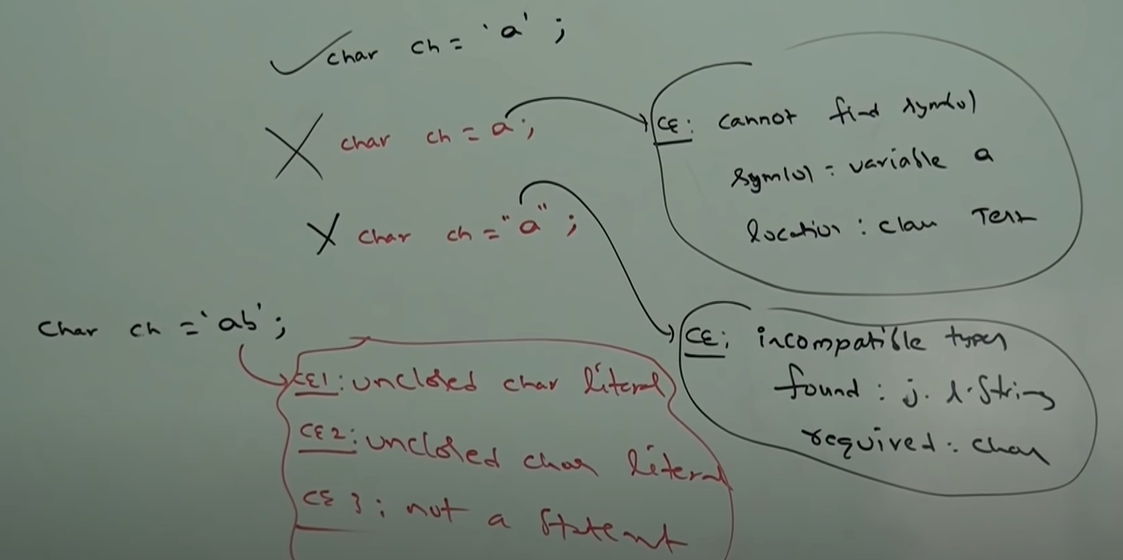
float 🡪 size – 4 bytes

double 🡪 size – 8 bytes

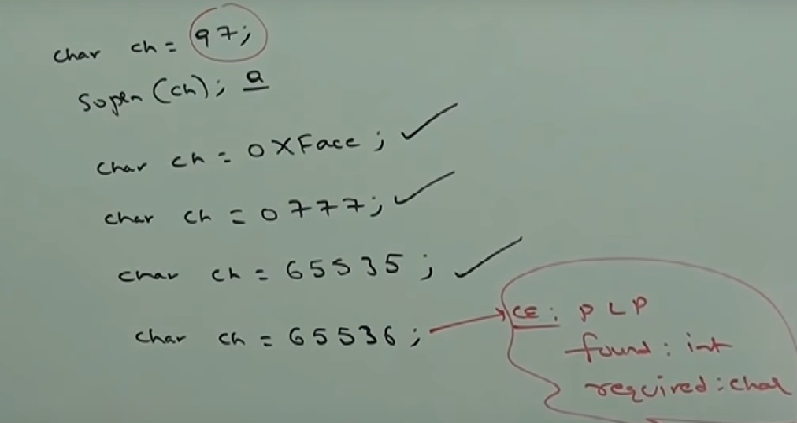
boolean 🡪 true or false



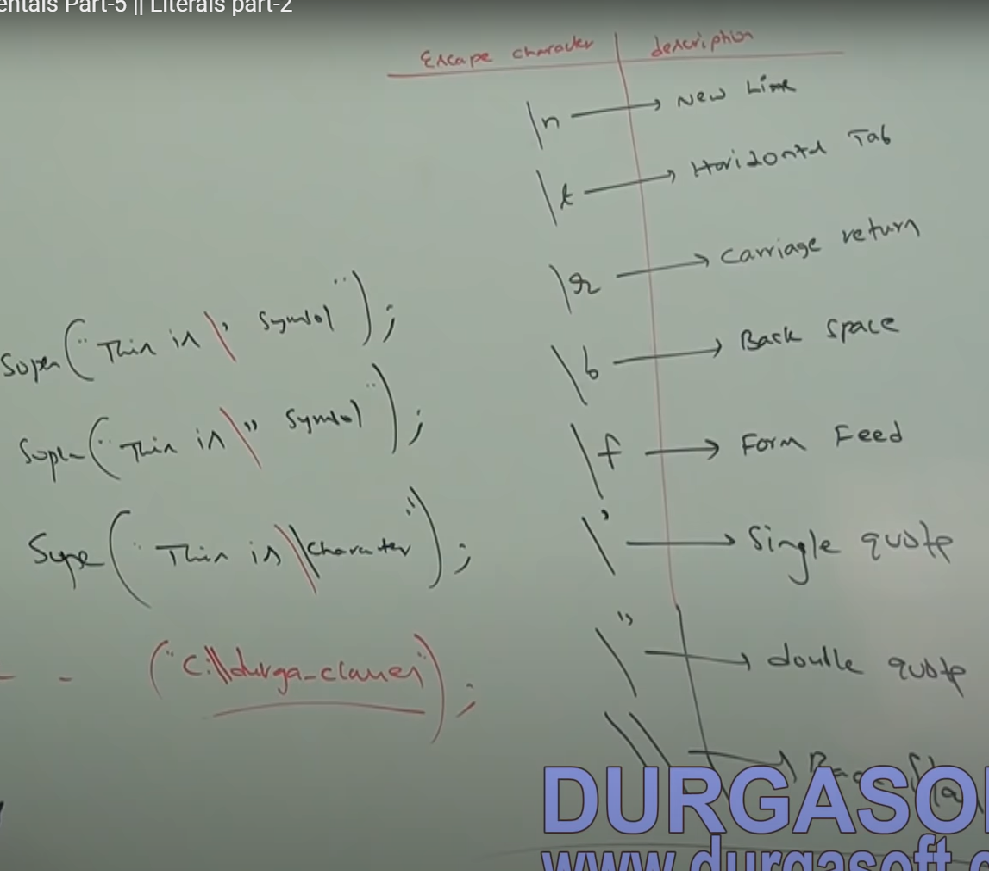
char 🡪 Old languages (c++/c) are ASCII code based, under that the number of different allowed ASCII code characters are <= 256. To the represent these 256 characters 8 bits are enough, hence the size of char in old languages is 8 bits but Java is Unicode based, under the number of different allowed Unicode characters add <= 65,536. to the present these many characters 8 bits may not be enough and so the **size of char in Java is made to be 16 bits or 2 bytes.**



We can specify char literals as integral literal which represents the Unicode value of the character and the integral literal can be specified either in decimal octal or hexadecimal form but allowed range is 0 to 65535

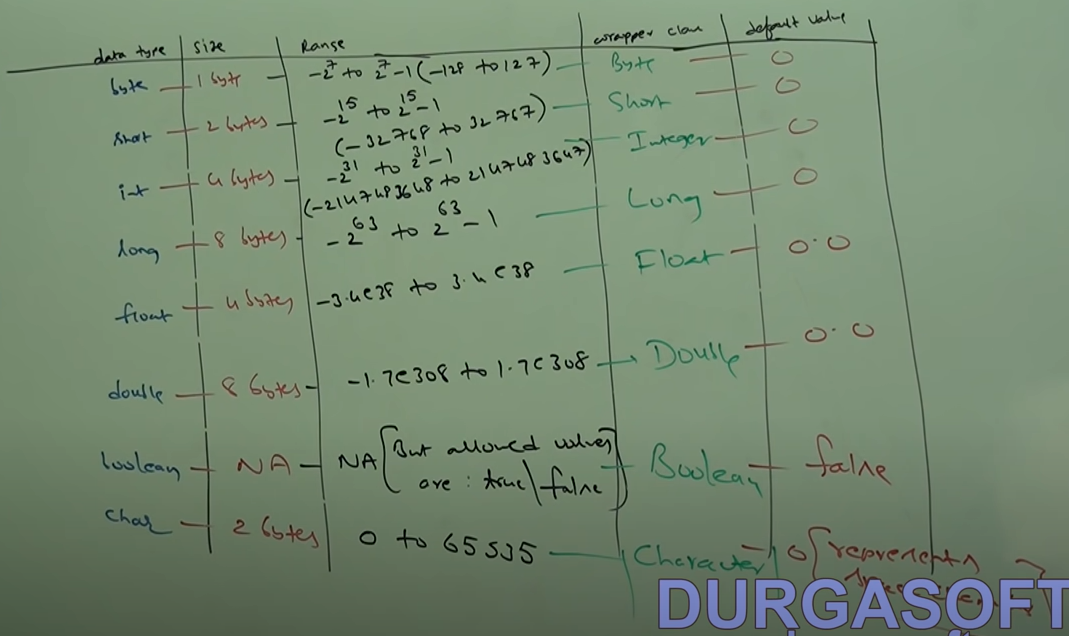


There are 8 escape characters in total and every escape character is a ‘char’ literal only :



String 🡪Any sequence of character within double quotes is treated as string literal.

Datatype size&range :



**JAVA LITERALS :**

What is a literal 🡪 Any constant value which can be assigned to the variable is called a literal.   
E.g. int x = 10; here value 10 is considered as a “Constant value/Literal”

Integral Literals 🡪 For integral data types like byte, short, int, long we can specify literal value in the following basis

* decimal literals - int x = 10; (mostly what is used)
* octal literals - int x = 010; (in java if a number is prefixed by 0, it is considered as octal literal)
* hexa decimal literals - int x = 0X10; or int x =0x10; (case does not matter)

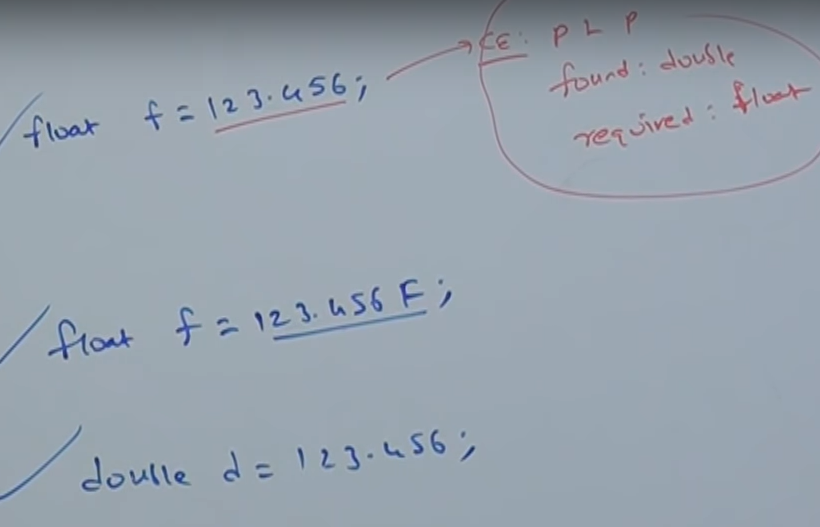
These are only possible way to specify literal values for integral data types. A programmer can define a value in any of the above ways but JVM will always provide the value in decimal literal i.e. while printing any variable may it be decimal or octal or hexa it will be printed in decimal form only.

By default every integral literal is of int type but we can specify it explicitly as long type by suffixing with small L or capital L.   
E.g. : long x = 10L;

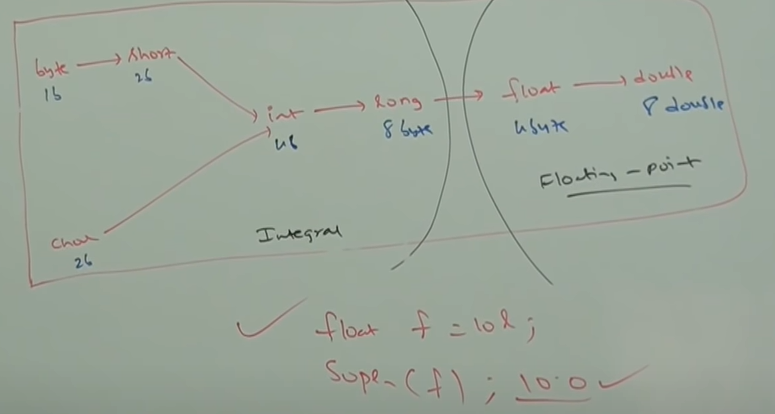
There is no direct way to specify byte and short literal explicitly but indirectly we can specify whenever we are assigning integral literal to byte variable and if the value is within the range of byte then the compiler treats it automatically as a byte literal, similarly for short literal also.

By default every floating type variable is of type double.

Same as long type, if we want to specify float type variable explicitly, we need to suffix the variable with small F or capital F.  
E.g : float f = 123.456F;



\*\*We can specify floating point literals only in decimal form, we cannot specify floating point literal in octal or hexadecimal forms, if we do compiler will throw ‘malformed floating point literal’.



8 byte long value we can assign to 4 byte float variable because both are following different memory representations internally

**ARRAY**

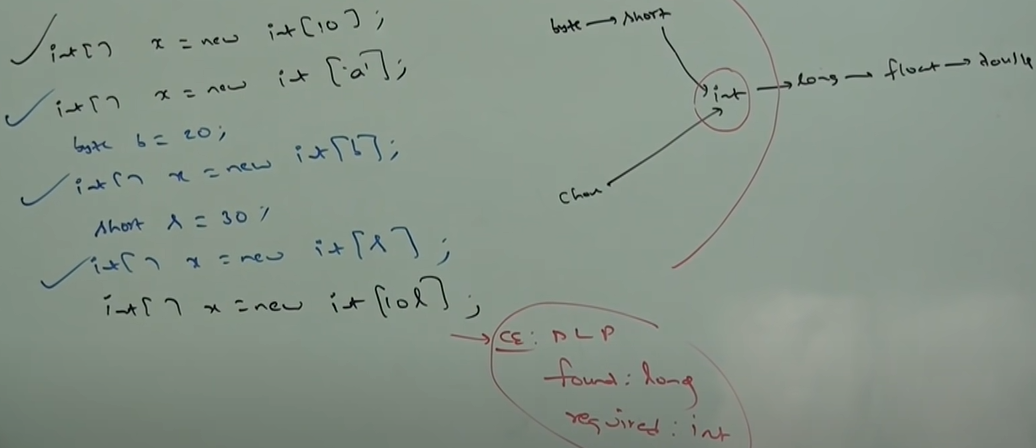
Array is an indexed collection of fixed number of homogeneous data type.

Array creation : int[] a=new int[sizeofarray];

Every array in Java is an object hence we can create array using new operator and it is compulsory to specify the size of array.   
We can specify the size of array as 0 i.e. int[] a=new int[0]; 🡪valid.

If we're trying to specify array size with some negative integer value, then we will get **Runtime exception** saying **NegativeArraySizeException.** E.g. int[] a=new int[-3]; 🡪 RE :  **NegativeArraySizeException.**

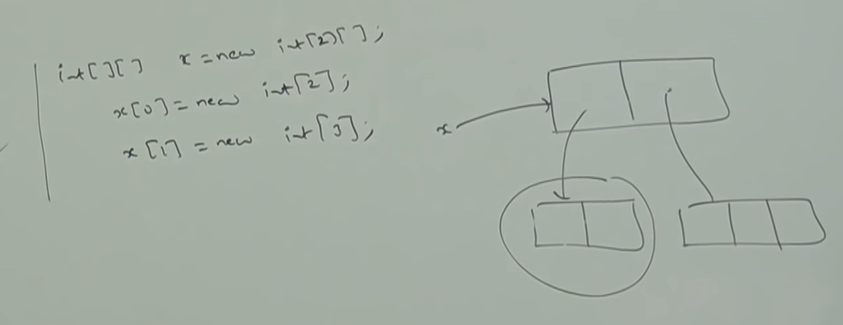
To specify array size the allowed data types are byte short char int ; if we're trying to specify any other data type then we will get compile time error



The maximum value of array size is the upper limit of int data type i.e. 2147483647. If we specify value like double or float then we will gat a compile time error saying **possible loss of precision** : found double required int

2-Dimensional Array Creation :

In Java 2D array may not be implemented using matrix representation rather Array Of Arrays approach is beneficial to be followed for multi-dimensional array creation. The main advantage of this approach is memory utilization will be improved.  
E.g. -



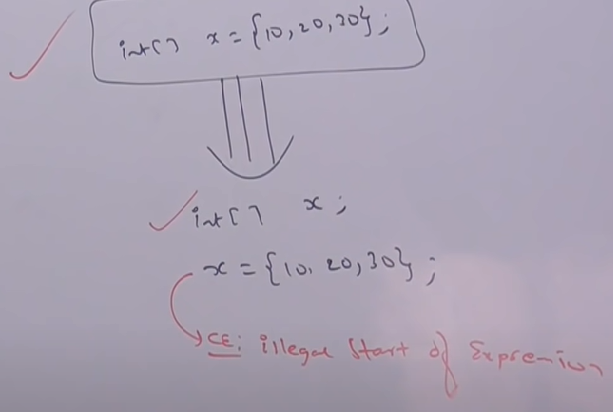
When we create an array, every element is initialized with default value i.e. 0.

Whenever we're trying to print any reference variable, internally the two string method will be called which is implemented by default to return the string in the following form : classname@hashcode\_in\_hexadecimalform.

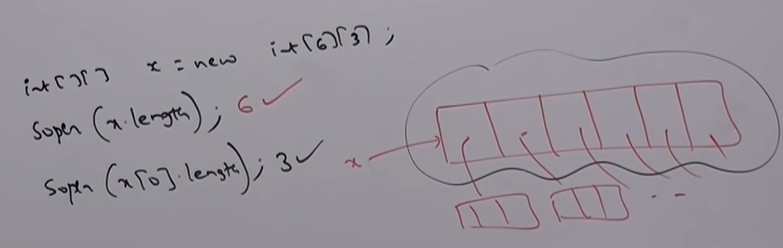
If we try to perform any operation on null we will get null pointer exception.

If we're trying to access array element with out of range index (either positive or negative integer value) then we will get runtime exception sing edit index out of bound exception.

We can create, declare and initialize an array in a single line like : int[] x = {1,2,3}; int[][] x = {{10,20},{30,40,50}};  
But for using this shortcut we need to specify everything in a single line. If we try to divide it we will get CE : Illegal start of expression



Length of Array :

length is a final variable applicable for array only, length variable represents the size of the array.  
E.g. - int[] x = {1,2,3}; S.o.pln(x.length); 🡪 3  
In multidimensional arrays length variable represents only base size but not the total size of the array.  
  
There is no direct way to get the total length of a multidimensional array but we can calculate it like :  
x[0].length + x[1].length + x[2].length …..

Anonymous Arrays :

Sometimes we can declare an array without specifying the name, such type of nameless arrays are called anonymous arrays and the main purpose of anonymous area is just for instant use one time usage.   
We can create anonymous arrays as follows -  
 >> new int[]{10,20,30,40};

While creating anonymous arrays we cannot specify the size otherwise we will get compile time error.  
Multidimensional anonymous array 🡪 new int[][]{{10,20},{30,40}};