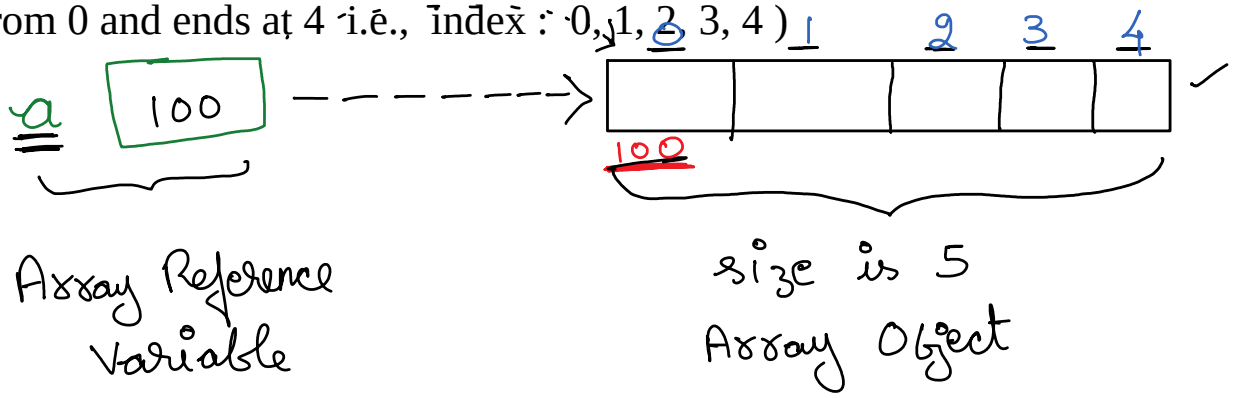
Tuesday, July 21, 2020 1:43 PM

Array :

Array is continuous block of memory, which is designed to store collection of same type of values.

Notes :

1. Array is fixed in its size, (we cannot increase or reduce the size of an array during runtime)
2. In array we can store multiple values, but it should be of same type. (Homogeneous)
3. We can access the elements present in an array with the help of an integer number starting from 0 up to the size of the array - 1, it is known as **index** or **subscript** of an array. (example if the size of the array is 5, then the index start from 0 and ends at 4 i.e., index : 0, 1, 2, 3, 4)



4. Array is a non-primitive type, it is represented with a pair of square brackets [].

To Create Array reference variable :

We can create an array reference variable with the help of array operator [ ] and the datatype.

Syntax :

```
datatype[]  identifier ;
datatype  identifier[]  ;
```

ex :

int[] a ;	// a is an integer array reference variable
char b [] ;	// a is char array reference variable
boolean[] c ;	// c is boolean array reference variable
Object [] obj ;	// obj is Object type array reference variable
String[] args ;	// args is String type array ref variable

What can we store in an Array Reference variable ?

1. we can store the default value null.
2. we can store the reference of the array object.

==========

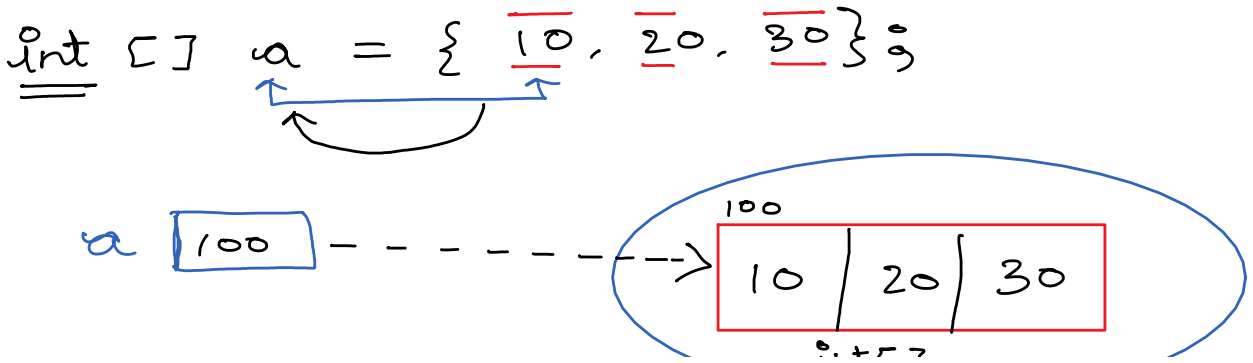
In Java, we can create array object in 2 ways :

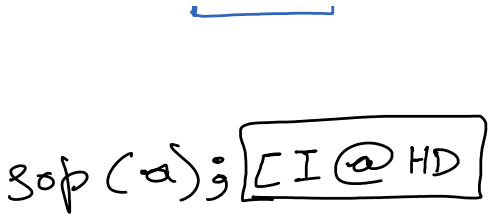
1. **Array declaration and initialization statement**
2. **with the help of new operator**

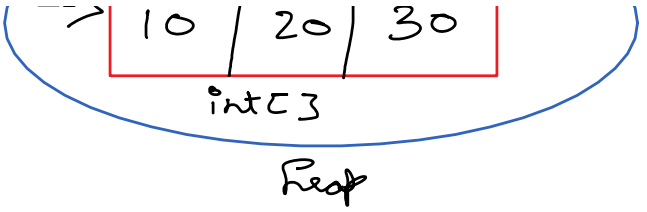
1. Using declare & initialization statement :

Syntax:

```
datatype[]  identifier    =  { element1 , element2, …..} ;
datatype    identifier[]  =  { element1 , element2, …..} ;
```







For example refer:

workspace / arrays / src / pack1 / A6.java - A8.java

Note :

in this syntax, the array length is same as the number of elements passed in the initialization.

2. TO Create an Array object using new operator :

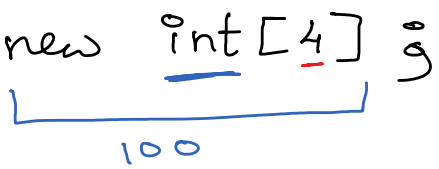
syntax :

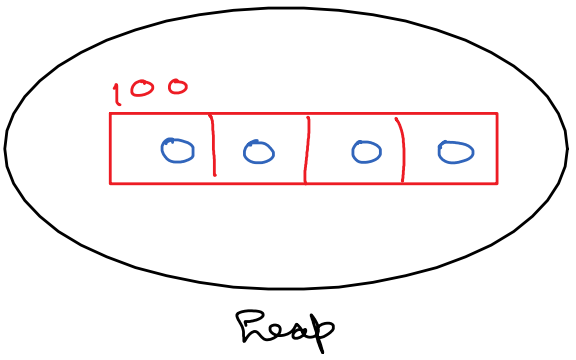
```
new  datatype[  size ]
```

Note :

1. new returns reference of array object created.
2. the array object created will be assigned with the default value of its type

ex:1

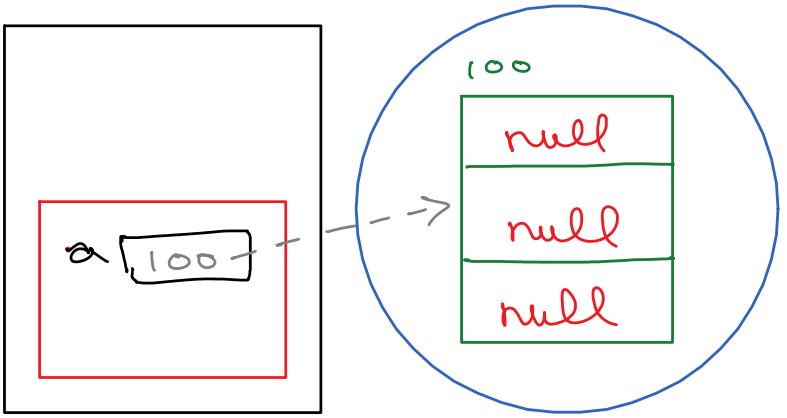




ex:2

Exc 2 :

```
String[] a = new String[3];
Sop (a.length); // 3
```



To Access the elements of an Array :

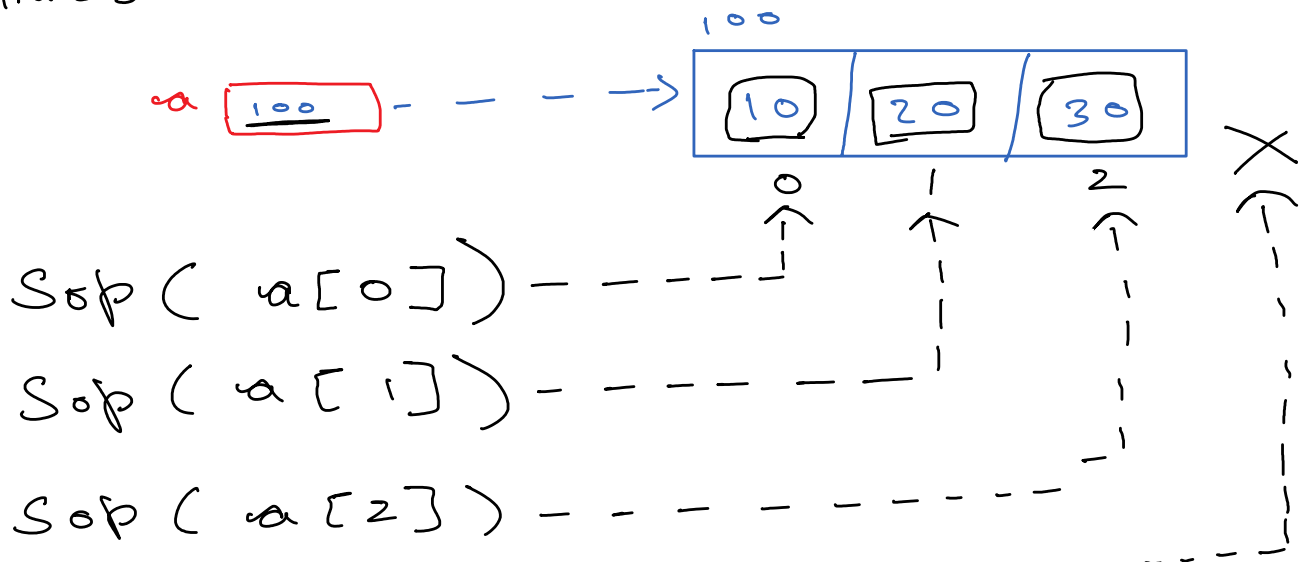
1. We can access the elements of the array with help of index.
2. The index of the array is always **int type**.
3. The index start from 0.
4. The index end at array size /length -1.

Syntax :

```
array_reference [  index ]
```

Exc :

```
int[] a = {10, 20, 30}
```



Sop (a[2]) - - - - - - - -

Sop (a[3]) - - - - - - - - Exception

// ArrayIndexOutOfBoundsException

Note :

If array index is less than zero or, greater than or equal to length of the array we get AIOOBE.

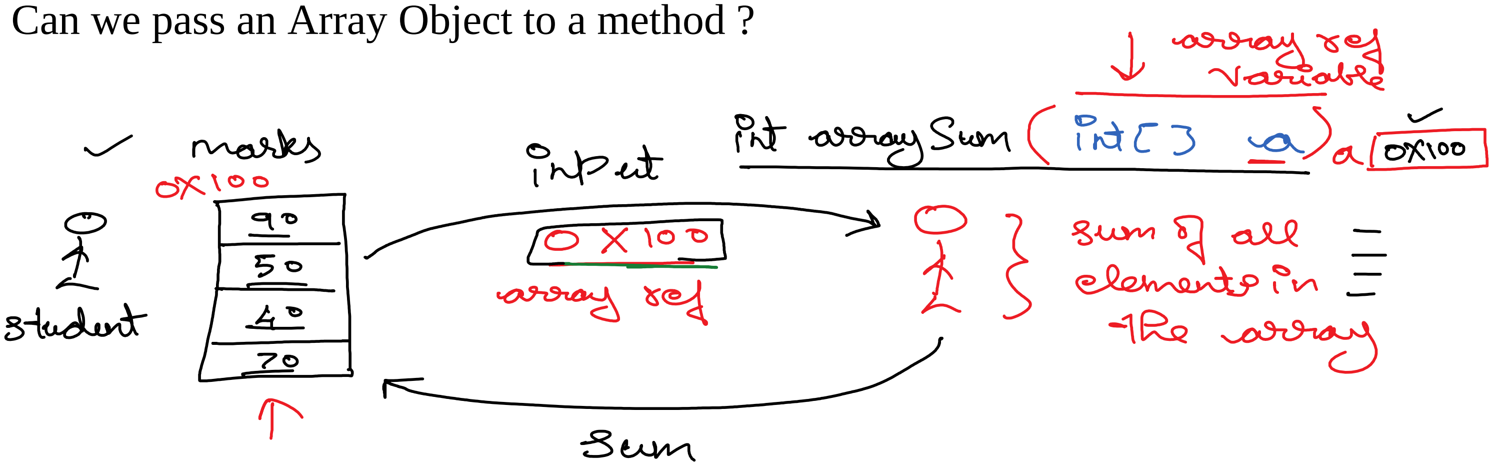
Assignment1 :

1. WAJP to read names of 5 cities from the user, store them in the array, and display the city names along with their length.

++

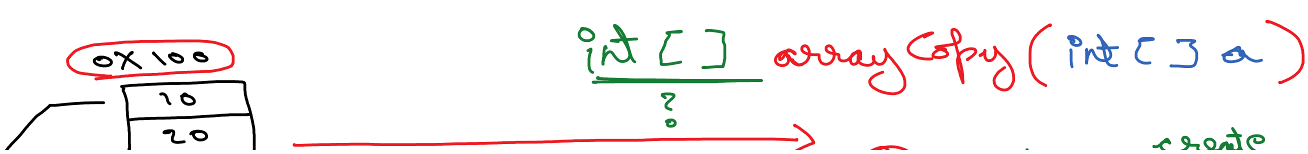
Passing an Array to a Method :

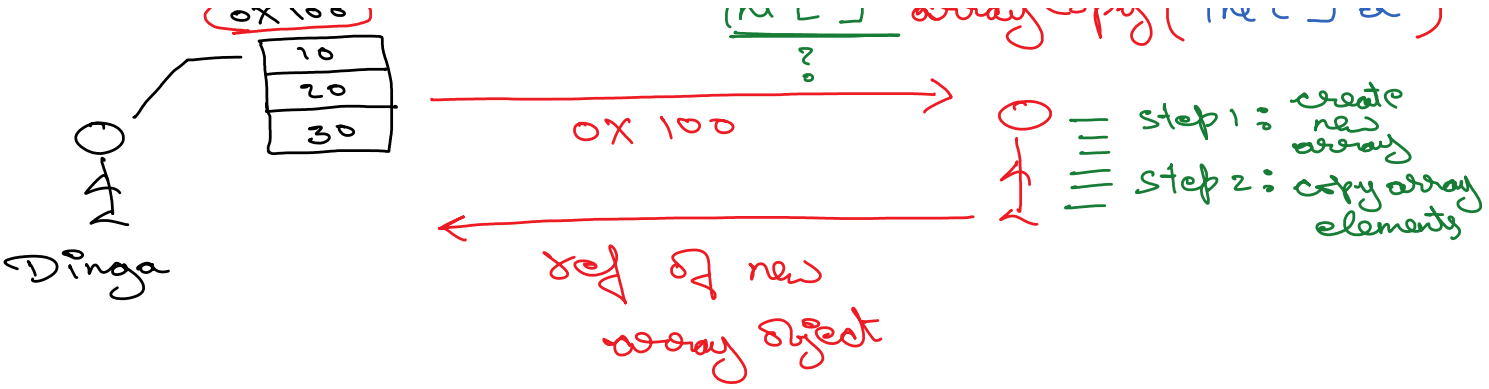
Can we pass an Array Object to a method ?



1. We can pass the reference of an array object to a method.
2. The method must be declared with an array reference variable as the formal argument only then it can accept the array.

Can a method return an Array :





1. A method can return the array reference to the caller.
2. To achieve this design the return type of the method must be an array type.

Task1 : Design a method to return a clone of an integer array object

==========

Note :

In java we have a built in class called as **Arrays in util package**. there the fully qualified name is java.util.Arrays, which has lot of built in methods to perform array operations such as :
copy array, sort array etc…

Example to sort the Array :

1. in java.util.Arrays we have a static method sort() which can accept the reference of an array and returns the sorted array in ascending order.

   example refer : **workspace/arrays/src/pack1/A13.java**

Task2 :

WAJP to do the following task :

1. ask the user total number of best friends he has got in his life time
2. read the names of all the best friends and store them in the array
3. display his best friend names in the ascending order.

Assignment 3 :

1. Create Employee class (attributes : eid , name, salary)
    1. achieve data hiding
    2. override toString, equals and hashCode
2. Create Company class.

    Attributes :
    1. company name
    2. an array of size 5 to store Employees

    design following methods (Features)

    1. a method to add employee
    2. a method to get employee based on id
    3. a method to search employee based on id
    4. a method to remove employee based on id
    5. a method to remove employee based on salary (same salary only first occurrence to be removed)

Sorting arrays using built in sort() method :

1. in util package we have a built-in class called Arrays.
2. in java.util.Arrays class, we have a built-in static method called sort which can accept a **reference of an array** and sorts the array in ascending order by default.

Note:

1. sort method can sort the array only if all the objects present in the array is comparable type.
2. If the object is not comparable type we get ClassCastException

Note :

1. We can sort a non-primitive array, using sort() method, but the objects stored in the array must be made comparable type.

Steps to make an Object Comparable Type :

Step1 : The class must implement java.lang.Comparable interface, then the objects of that class or its subclass is considered as comparable type.

Note:

1. Comparable interface has only one abstract method called compareTo().

   Method declaration is :

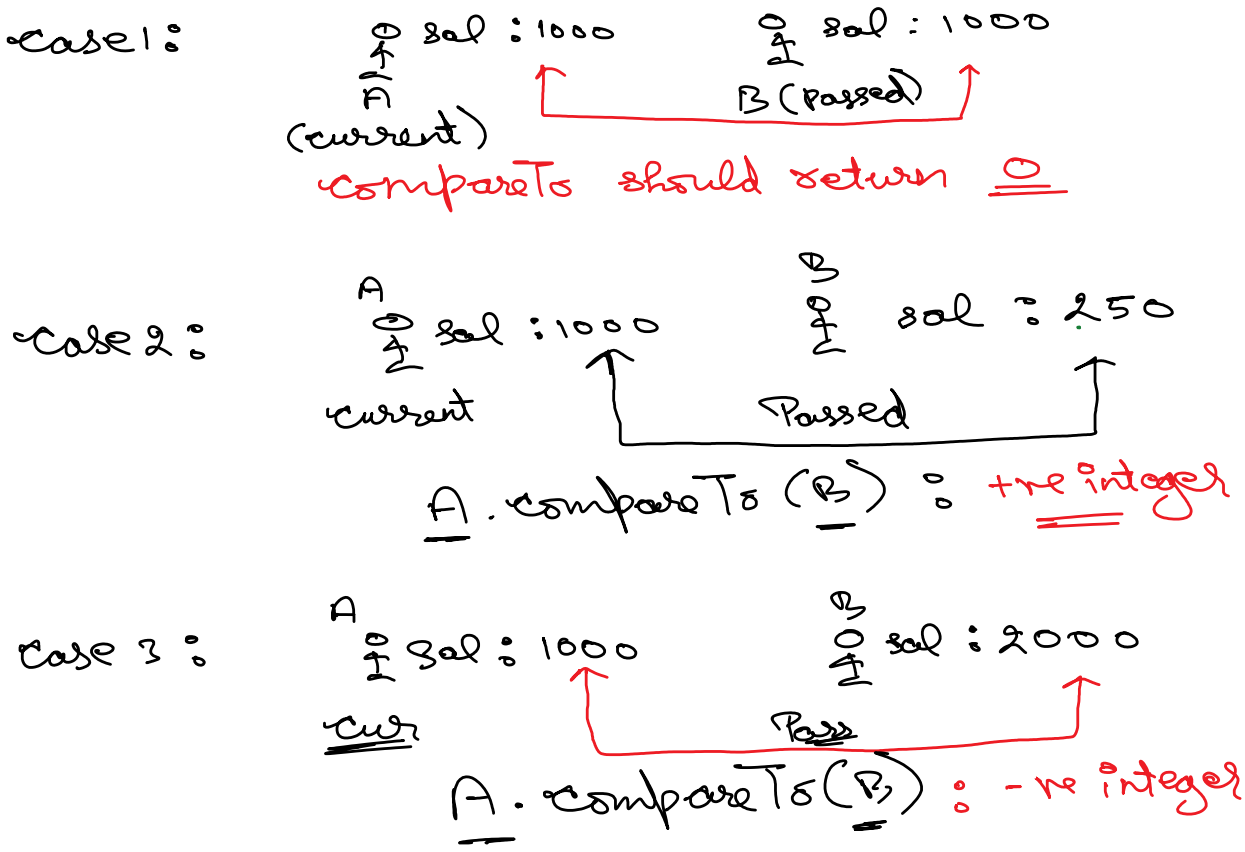
   public int compareTo (Object obj) ;

Step2: The implementing class must override and provide implementation to the abstract method of the Comparable interface.

Design tip to override compareTo():

1. compareTo method is used to compare 2 objects.
2. it can accept only one object as argument.
3. compareTo method compares the current object with the object passed as argument.
4. return type of compareTo method is int, Therefore

    1. it should return 0, if the state of both the objects is same.
    2. it should return +ve integer, if state of current object is higher than state of object passed.
    3. it should return -ve integer, if the state of current object is lower than the state of passed object.



Example : **refer workspace/arrays/src/pack3**

Question:

1. **Why a class must implement Comparable interface ?**

Ans: We have to make the class Comparable type in order to sort the objects

in the array or Collection using java built-in sort methods.

Task 1 :

ST1 :

1. **Design a blueprint for Laptop, make the class comparable type.**
2. **properties of latop(ram_size , hard_disk, retail_price)**
3. **create suitable constructors**
4. **override toString, hashCode and equals method.**

ST2 :

1. **Create a driver class, in the driver class create an array to store 3 laptop objects and sort them based on their retail price.**