Date: YOUVA

Arrays:

An array is a collection of elements which can be primitives, object, complex DT.

Syntax:

datatype[] variable_name = new datatype[size];

Ex: Store 5 roll numbers

int[] rnos = new int[5];

↓ ref var rnos points to array object that contains Integer elements.

(or)

int[] rnos = {1,2,3,4,5};

1. int[] represents the type of elements stored in an array.
2. All the elements in an array should be same.

{1,2,3,4,5} ✓

{1,2,"Sri",4,5} ✗

How Array works:

int[] ros; → declaration of an array, ros is getting defined in the stack

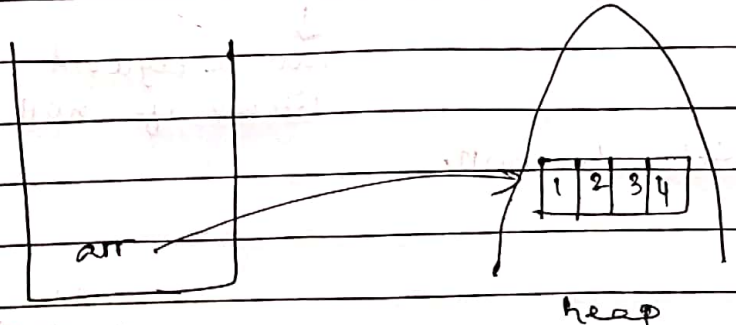
ros = new int[5]; → initialization, actually here object is getting created in the heap.

int[] arr = new int[5];

DT — ref var — obj in heap

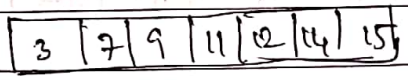
DT happens at compile time

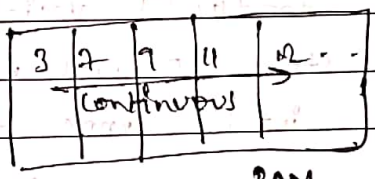
Mem is allocated at runtime (Dynamic Mem Allocation)



Internal representation of an array:

In general in c & c++ array is stored in continuous mem allocation which is not the case in Java.



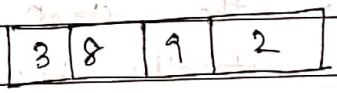


JVM decides whether to store an array in continuous or not. Because

1. objects are stored in heap.
2. In Java lang specification (JLS), it is mentioned that heap objects are not continuous.
3. Hence array objects in Java may not be continuous.

Index of an array:

positioned number of array starting from 0.



arr[0] = 3

arr[2] = ~~2~~ 10

arr[2] = 10

new:

Used to create an object

By default an integer array stores '0' . 0 (zero)

String array stores null

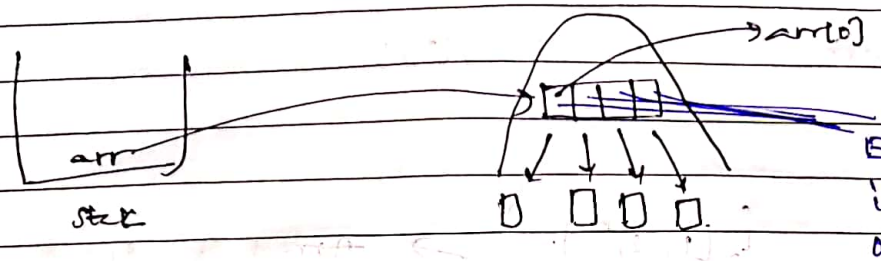
↓

Not a keyword

literal of null type

By default ref var point to null.

String [] arr = new String [5];



Each particular element in an array is an object stored in seperate Mem.

By dfault for every reference variable if there is no value, its null.

Array Input:

```
psvm{
    Scanner in = new Scanner(System.in);
    int[] arr = new int[5];
    for(int i=0 ; i<arr.length ; i++){
        sout(arr[i]);
    }
```

toString() Method:

```
System.out.println(Arrays.toString(arr));
```

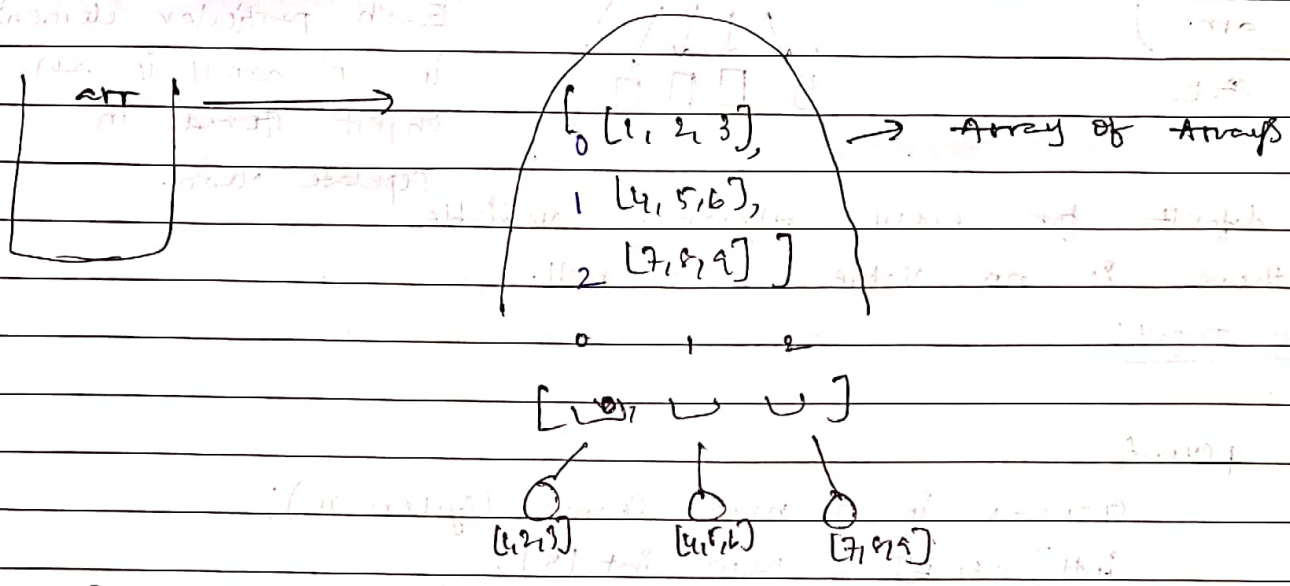
↓ converts arr to string

Array of Strings:

```
String[] str = new String[4];
for(int i=0; i<str.length; i++){
    str[i] = sc.next.in.next();
}
sout(Arrays.toString(str));
```

Multi Dimension Arrays:

```
int [] [] arr = new int[3][3];
```



arr[1] → [4,5,6]

↓

arr[1][0] → 4

No. of rows must be specified but not columns. Since each object is another individual array their size doesn't matter

```
{
    { 1, 2, 3}, // 0th index
    {4, 5}, // 1st index
    { 6, 7, 8, 9} //2nd index
};
```

2D Array Input:

```
int [][] arr = new int[3][2];
for(int row = 0; row < arr.length; row++){
    for(int col = 0; col < arr[row].length; col++){
        arr[row][col] = in.nextInt();
    }
}
```

Array output:

```
for (int row = 0; row < arr.length; row++) {
    for (int col=0; col < arr[row].length; col++) {
        System.out.print(arr[row][col]);
    }
    System.out.println();
}
```

Arraylist:

Used when we do not know the size of the array.

```
ArrayList<Integer> list = new ArrayList<Integer>();
list.add(65);
list.add(100);
list.add(100);
.
.
.
list.add(150);
System.out.println(list);
```

intial capacity: 5

add as many as req

Array functions:

```
System.out.println(list.contains(65));
```

↓
true

```
list.set(0, 99)
```

↳ 0th index changed to 99

```
list.remove(2);
```

↳ 100 will be removed from above array

to get an item

```
Sout(list.get(2));
```

```
for (int i = 0; i<5; i++){
    list.add(in.nextInt());
}
```

Multiple Arraylist:

```
Scanner in = new Scanner(System.in);
ArrayList<ArrayList<Integer>> list = new ArrayList<>();
// initialization
for (int i=0; i<3; i++){
    list.add(new ArrayList<>());
}
// add elements
for (int i=0; i<3; i++){
    for (int j=0; j<3; j++){
        list.get(i).add(in.nextInt());
    }
}
System.out.println(list);
```

Question 1:

Swap two values in array:

```
{ int [] arr = {1, 2, 3, 4, 5};
swap(arr, 1, 3);
System.out.println(Arrays.toString(arr));
}

static void swap(int[] arr, int in1, int in2){
    int temp = arr[in1];
    arr[in2] = arr[in];
    arr[in] = temp;
}
```

2. Max value of an array:

```
psvm{
  int[] arr = {1, 3, 9, 11, 23, 18};
  System.out.println(max(arr));
}
  static int max(int[] arr){
      int maxVal = arr[0];
      for(int i=0; i<arr.length; i++){
          if(arr[i] > maxVal){
              maxVal = arr[i];
          }
      }
      return maxVal;
  }
```

3. Reverse the array:

```
psvm{
  int arr[] = {1, 3, 5, 7, 9, 11};
    reverse(arr);
  sout(Arrays.toString(arr));}
static void reverse(int[] arr){
    int start = 0;
    int end = arr.length-1;
    while(start < end){
        swap(arr, start, end);
        start++;
        end--;
    }
static void swap(int[] arr, int start, int end){
    int temp = arr[start];
    arr[start] = arr[end];
    arr[end] = temp;
}
```