Array of Object

Syntax:

```
Class_name obj[] = new Class_name[arr.length];
```

or

~~Class_name=~~

```
Class_name[] obj;
```

or

```
Class_name obj[];
```

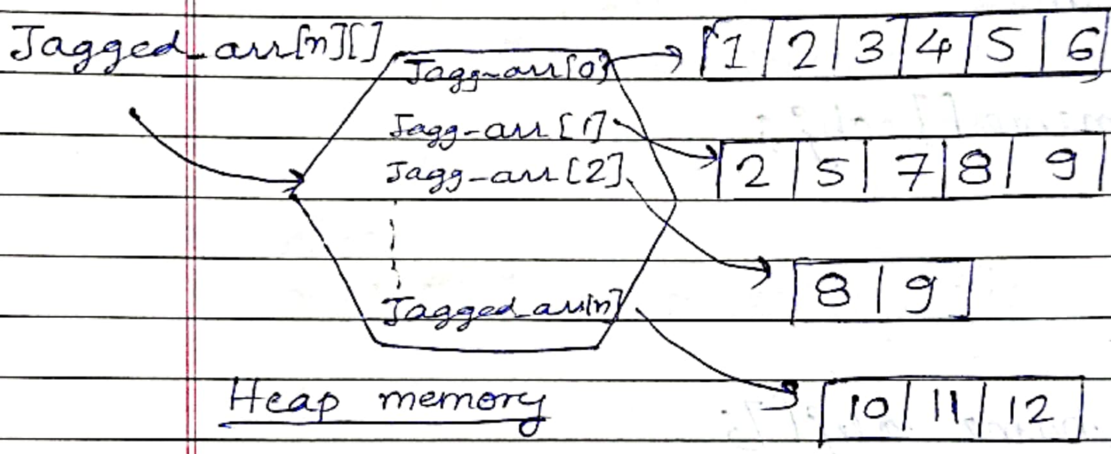
E.g.

```
Product[] obj = new Product[5];
obj[0] = new Product(23907, "Dell Laptop");
obj[1] = new Product(91240, "HP G30");
```

JAGGED ARRAY

A jagged array is an array of arrays such that ~~mem~~ member can be of diff. sizes i.e. we can create a 2-D array but with variable no. of columns in each row

Pictorial representation



Declaration: (Syntax)

```
data_type array_name[][]=
        new data_type[n][];
        //n=no. of rows
array_name[]=new data_type[n1] //n1=
                    //col. in row 1

array name[]=new data_type[n2]

 //n2= col. in row 2
.
:
array_name[]=new data_typ[nk];..
//nk= col. in row n
```

Alternative ways to initialize

```
int arr[][] = new int [][]{

  new int[]{10,20,30,40},
   "    "   "{50,60,70},
   "    "   "{110, 120}
        };
```

OR

```
int arr[][] = {
      new int[]{1, 2, 3, 4},
       "   "   "{5, 6, 7, 8, 9, 10};
       "   "   "{11, 12}
      };
```

~~in~~ OR

```
int arr [] [] = {
        {1, 2, 3, 4};
        {5, 6, 7, 8, 9},
        {11, 12}
          };
```

WRAPPER CLASS

→ Java treats objects differently from variables of Primitive ~~obj~~ types.
- Sometimes we need to treat int, char, float values as Objects.
- Java provides Wrapper class for each primitive type which wraps the value as an Object.

→ The following declaration creates an Integer object which is reference to an object with the Integer value 40.

```
Integer age = new Integer (40);
```

→ An object of wrapper class is used in situations where primitive value won't suffice
→ For e.g. some objects serve as containers of other objects
→ Primitive values could not be stored in such containers, but wrapper objects could be.

→ Wrapper class may contain static methods that help manage the associated type. For e.g, ~~we can convert~~ the Integer class contains a method to convert digits stored in a String to an int value.

```
num = Integer.parseInt(str);
```

→ Wrapper class contains useful constants
For e.g.
Integer class contains MIN_VALUE & MAX_VALUE

→ The Java. lang package contains a wrapper class that corresponds to each primitive type

Boxing

→ Converting from primitive to wrapper class is called as Boxing.

```
Integer intobj = new Integer(575);
```

UnBoxing

→ Converting wrapper to primitive is called as unboxing

```
int i = intobj.intValue();
```

Auto boxing

Auto boxing is automatic conversion of a primitive value to a corresponding wrapper object.

```
Integer obj;
int num = 42;
obj = num;
```

→ The assignment creates the appropriate Integer object wrapping a value of 42.
→ The reverse conversion (called unboxing) also occurs automatically as needed.

$$\begin{tikzcd}\boxed{\text{Integer intObj = new Integer(2);}} \arrow[d, Rightarrow, "\text{Equivalent}"] \\ \boxed{\text{Integer intObj = 2;}} \arrow[r, leftarrow] & \text{auto boxing}\end{tikzcd}$$

```
class autobox{

   p s v m (   ){
    Integer iOb=100   // autobox an int
    int i= iOb;      // at auto-unbox
    sout(i+" "+iOb);
   }
 }
```

o/p :

```
100   100
```

E.g. Code:

```
public class Main{
   p  s  v  m  (    ){
    Integer myInt =100;
    Double  mydoub= 5.99;
    Character myChar = 'A';
    sout(myInt.floatValue());
    sout(myDouble.intValue());
# sout(myChar.char Value());
// ~~sout (myChar.int~~ V
// sout(myChar.intValue()); } throws error
// sout(myInt.charValue());

String myString = myInt.toString();
 sout(myString);
 sout(myString.length);

     }
  }
```

o/p

```
100.0
5
A
100
3
```

type can be byte, int etc.

Methods	Description
typeValue()	Converts the value of the number object to the specified primitive data type returned
compareTo(arg)	Compares this numb. object to the argument
equals(arg)	Determines whether this no. object is equal to the argument
valueOf()	Returns an Integer object holding value of specified primitive data type

toString()	Returns a string object representing the value of specified Integer type argument
parseInt(arg)	Returns a Integer type value of specified Integer type argument
decode( )	decodes a string into an Integer
min(arg, arg)	Returns smaller value of comparison with the arguments
round()	Returns the closest ~~to~~ round of long or int value as per method return type.