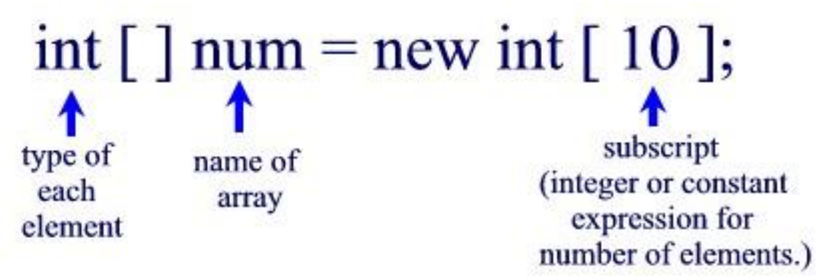
One Dimensional Arrays

An **array** is a collection of data storage locations, each of which holds the **same type of data.** An array is storage in memory which can store multiple values of the same data type. Typically used to store large amounts of data.

1. Declaration and terminology

Example: int [] num = {7,6,5,8,3,9,2,6,10,2};

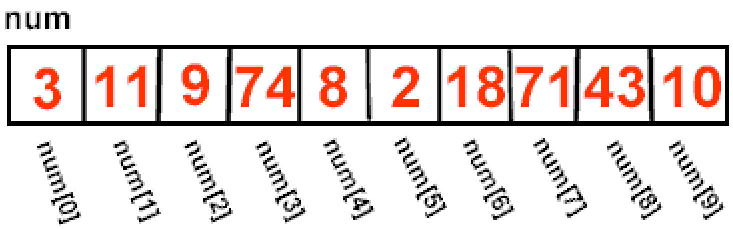
3	11	9	74	8	2	18	71	43	10



```
int [ ] num = new int [ 10 ];
```

type of each element
name of array
subscript (integer or constant expression for number of elements.)

2. Indices and values

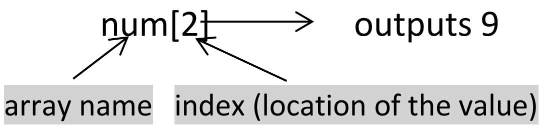


num

num[0]	num[1]	num[2]	num[3]	num[4]	num[5]	num[6]	num[7]	num[8]	num[9]
3	11	9	74	8	2	18	71	43	10

3. Array elements access

a) Single element



num[2] → outputs 9

array name index (location of the value)

b) All elements access- loop

```
for(int a = 0; a<num.length; a++)
{
  num[a];
}
```

c) **For each** loop access

```
int [] num = {7,6,5,8,3,9,2,6,10,2};

int sum = 0;
for (int b : num) {  // b gets successively each value in num
```

Iterator   array name

```
 sum += b;  // adding all values to a sum variable
}
```

4. Errors and other details

num [0]	always OK
num [9]	OK (given the above declaration)
num [10]	IndexOutOfBound (no such index)
num [-1]	No such index (illegal)
num [3.5]	Index MUST be an integer (illegal)

a) If the value of an index for an array element **is negative, a decimal**, or greater than **or equal to** the length of the array (remember that the last subscript is array length - 1), an error message will be *ArrayIndexOutOfBoundsException*.

b) length vs length()

num.length	**String_value.length( )**
is used to find the length of an array named num.	is a method used to find the length of a **String** named value(**not an array**)