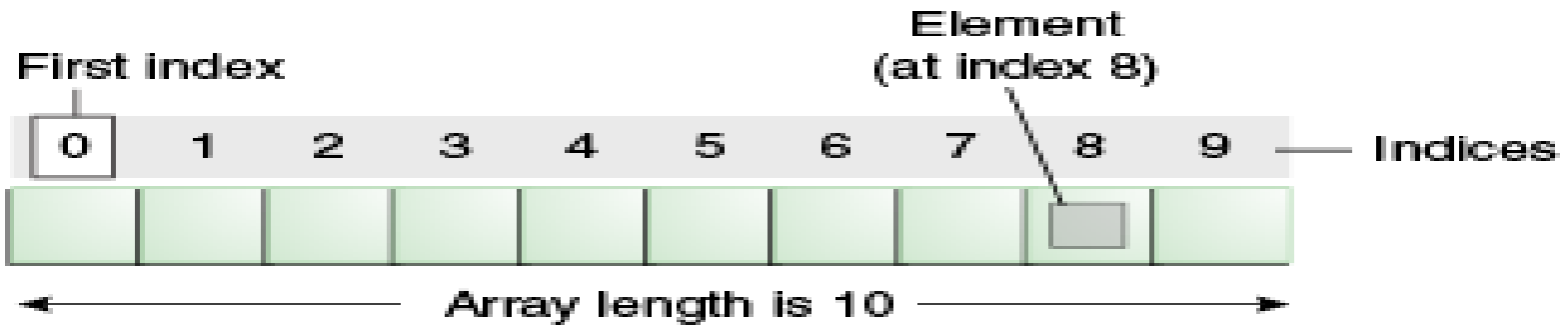
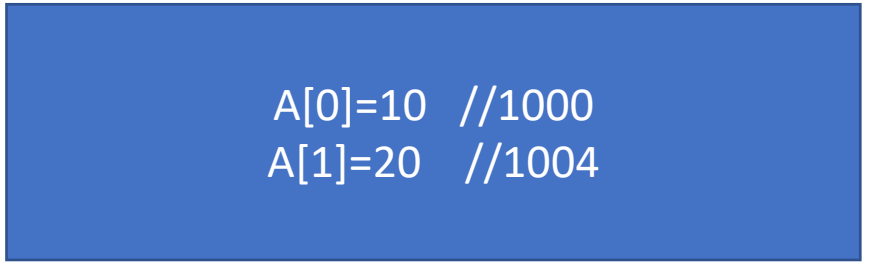
JAVA Strings and Arrays

Java Arrays

Normally, an array is a collection of similar type of elements which has contiguous memory location. **Java array** is an object which contains elements of a similar data type. Additionally, The elements of an array are stored in a contiguous memory location. It is a data structure where we store similar elements. We can store only a fixed set of elements in a Java array.





Advantages

•**Code Optimization:** It makes the code optimized, we can retrieve or sort the data efficiently.

•**Random access:** We can get any data located at an index position.

Disadvantages

•**Size Limit:** We can store only the fixed size of elements in the array. It doesn't grow its size at runtime. To solve this problem, collection framework is used in Java which grows automatically.

Types of Array in java

There are two types of array.

•Single Dimensional Array

•Multidimensional Array

Single Dimensional Array in Java

Syntax to Declare an Array in Java

```
dataType[] arr; (or)     int[] a;
dataType []arr; (or)     int []a;
dataType arr[];           int a[];
```

```
arrayRefVar=new datatype[size];
```

```
class Testarray{
public static void main(String args[]){
int a[]=new int[5];
a[0]=10;
a[1]=20;
a[2]=70;
a[3]=40;
System.out.print(a[0]+ " "+a[1])

for(int i=0;i<5;i++)
{
System.out.println(a[i]);
}
}
}}
```

```
String[] cars = {"Volvo", "BMW", "Ford", “Maruti"};
```

Change an Array Element
To change the value of a specific element, refer to the index number:

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
cars[0] = “Audi";
 System.out.println(cars[0]);
```

For-each Loop for Java Array

We can also print the Java array using **for-each loop**. The Java for-each loop prints the array elements one by one. It holds an array element in a variable, then executes the body of the loop.

```
for(data_type variable:array)
{
//body of the loop
}
```

```
int arr[]={33,3,4,5};

for(int i:arr)
System.out.println(i);
}}
```

Array Length

To find out how many elements an array has, use the `length` property:

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
 System.out.println(cars.length);
```

Java Program to sort the elements of an array in ascending order

Original array:

5	2	8	7	1

Array after sorting:

1	2	5	7	8

Algorithm

Tmp=0

- **STEP 1:** START
- **STEP 2:** INITIALIZE arr[] ={5, 2, 8, 7, 1}.
- **STEP 3:** SET temp =0
- **STEP 4:** PRINT "Elements of Original Array"
- **STEP 5:** REPEAT STEP 6 UNTIL i<arr.length
  //for(i=0; i<arr.length; i++)
- **STEP 6:** PRINT arr[i]
- **STEP 7:** REPEAT STEP 8 to STEP 9 UNTIL i<arr.length
  //for(i=0; i<arr.length; i++)
- **STEP 8:** REPEAT STEP 9 UNTIL j<arr.length
  //for(j=i+1;j<arr.length;j++)
- **STEP 9:** if(arr[i]>arr[j]) then
  temp = arr[i]
  arr[i]=arr[j]
  arr[j]=temp
- **STEP 10:** PRINT new line
- **STEP 11:** PRINT "Elements of array sorted in ascending order"
- **STEP 12:** REPEAT STEP 13 UNTIL i<arr.length
  //for(i=0;i<arr.length;i++)
- **STEP 13:** PRINT arr[i]
- **STEP 14:** END

Multidimensional Arrays

A multidimensional array is an array containing one or more arrays.

To create a two-dimensional array, add each array within its own set of **curly braces**:

```
int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };
```

```
public class MyClass
{
public static void main(String[] args)
{
int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };
 for (int i = 0; i < myNumbers.length; ++i)
 {
      for(int j = 0; j < myNumbers[i].length; ++j) {
      System.out.println(myNumbers[i][j]);

} } } }
```

Passing Array to a Method in Java

We can pass the java array to method so that we can reuse the same logic on any array. Let's see the simple example to get the minimum number of an array using a method.

```
class Testarray{

static void min(int arr[]){
   int min=arr[0];
   for(int i=0;i<arr.length;i++)
   if(min>arr[i])
       min=arr[i];
   System.out.println(min);
}
 public static void main(String args[]){
   int a[]={33,3,4,5};

min(a);
}
```

Java Array Exercise Questions

1. Print a Matrix
2. Add 2 Matrices
3. Subtract 2 matrices
4. Program to copy all elements of one array into another array
5. Java Program to print the sum of all the items of the array
6. Java Program to print the array in reverse order

1 3 5 7 9 =1+3+5+7+9=25

Java String

In Java, string is basically an object that represents sequence of char values. An array of characters works same as Java string. For example:

```
String greeting = "Hello";
```

```
char[] ch={'H','e','l','l','o'};
```

String Length

A String in Java is actually an object, which contain methods that can perform certain operations on strings. For example, the length of a string can be found with the `length()` method:

```
String txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
System.out.println("The length of the txt string is: " + txt.length());
```

More String Methods

There are many string methods available, for example `toUpperCase()` and `toLowerCase()`:

```
String txt = "Hello World";
System.out.println(txt.toUpperCase()); // Outputs "HELLO WORLD"
System.out.println(txt.toLowerCase()); // Outputs "hello world"
```

Finding a Character in a String

The `indexOf()` method returns the **index** (the position) of the first occurrence of a specified text in a string

```
String txt = "Please locate where 'locate' occurs!";
 System.out.println(txt.indexOf("locate"));
```

The + operator can be used between strings to combine them. This is called **concatenation**:

```
String firstName = “Meera";
 String lastName = “K R";
 System.out.println(firstName + " " + lastName);
```

You can also use the `concat()` method to concatenate two strings:

```
System.out.println(firstName.concat(lastName));
```

Special Characters

Because strings must be written within quotes, Java will misunderstand this string, and generate an error:

```
We are the so-called “Aliens" from the Space.
```

```
String txt = "We are the so-called \”Aliens\" from the Space.";
```

```
String txt = "It\'s alright.";
```

Java String Class Methods

No.	Method	Description
1	char charAt(int index)	returns char value for the particular index
2	int length()	returns string length
3	String substring(int beginIndex)	returns substring for given begin index.
4	String substring(int beginIndex, int endIndex)	returns substring for given begin index and end index.
5	String replace(CharSequence old, CharSequence new)	replaces all occurrences of the specified CharSequence.
6	String trim()	removes beginning and ending spaces of this string.

Java StringBuffer class

Java StringBuffer class is used to create mutable (modifiable) string. The StringBuffer class in java is same as String class except it is mutable i.e. it can be changed.

```
StringBuffer sb=new StringBuffer("Hello ");
```

Important Constructors of StringBuffer class

Constructor	Description
StringBuffer()	creates an empty string buffer with the initial capacity of 16.
StringBuffer(String str)	creates a string buffer with the specified string.
StringBuffer(int capacity)	creates an empty string buffer with the specified capacity as length.

Important methods of StringBuffer class

The append() method concatenates the given argument with this string.

```
class StringBufferExample{
public static void main(String args[]){
StringBuffer sb=new StringBuffer("Hello ");
sb.append("Java");
System.out.println(sb);
}
}
```

2) StringBuffer insert() method

The insert() method inserts the given string with this string at the given position.

```
StringBuffer sb=new StringBuffer("Hello ");
sb.insert(1,"Java");
```

3) StringBuffer replace() method

The replace() method replaces the given string from the specified beginIndex and endIndex.

```
StringBuffer sb=new StringBuffer("Hello");
sb.replace(1,3,"Java");
```

4)StringBuffer delete() method

The delete() method of StringBuffer class deletes the string from the specified beginIndex to endIndex.

```
StringBuffer sb=new StringBuffer("Hello");

sb.delete(1,3);
```

5) StringBuffer reverse() method

The reverse() method of StringBuilder class reverses the current string.

```
sb.reverse();
```

Garbage Collection

Garbage Collection in Java is a process by which the programs perform memory management automatically. The Garbage Collector(GC) finds the unused objects and deletes them to reclaim the memory

Garbage collection in Java happens automatically during the lifetime of the program, eliminating the need to de-allocate memory and thereby avoiding memory leaks.

Advantage of Garbage Collection

- It makes java **memory efficient** because garbage collector removes the unreferenced objects from heap memory.
- It is **automatically done** by the garbage collector(a part of JVM) so we don't need to make extra efforts.

How can an object be unreferenced?

There are many ways:

- By nulling the reference
- By assigning a reference to another
- By anonymous object etc.

gc() method

The gc() method is used to invoke the garbage collector to perform cleanup processing. The gc() is found in System and Runtime classes.

```
public static void gc(){}
```

finalize() method

The finalize() method is invoked each time before the object is garbage collected. This method can be used to perform cleanup processing. This method is defined in Object class as:

```
public void finalize(){}
```

```
public class TestGarbage1{
 public void finalize(){
System.out.println("object is garbage collected");
}
 public static void main(String args[]){
  TestGarbage1 s1=new TestGarbage1();
  TestGarbage1 s2=new TestGarbage1();
  s1=null;
  s2=null;
  System.gc();
 }
}
```