Access modifiers in java
++++++++++++++++++++++
1. private
2. public
3. protected
4. static[if we mark method as static, then that method can be called without creating the object]
5. strictfp
6. synchronized
7. final
8. abstract
9. native
10.transient
11.volatile

++++++++++++++++++++++++++++++++
length property vs length() method
++++++++++++++++++++++++++++++++
length: It is property which belongs to Arrays.
 => publi final int length;
 => This property would return the no of elements present inside the array.

```
eg#1.
int[] arr = {10,20,30};
System.out.println(arr);//[I@...
System.out.println(arr.length);//3
System.out.println(arr.length());//CE

eg#2.
int[][] arr = new int[6][3];
System.out.println(arr.length);//6
System.out.println(arr[0].length);//3
```

length() : It is available inside "String" class in java.
 => public int length();
 => This method would return the no of characters present in the given String.

```
eg#1.
String name = "sachin";
System.out.println(name);//sachin
System.out.println(name.length());//6
System.out.println(name.length);//CE
```

Ananomyous Array
================
 => An array without a name is called Ananomyous Array.
 => These type of array is created just for instance use.
 => Creation of Ananomyous Array

```
      new int[]{10,20,30,40};
      new String[]{"sachin","kohli","dhoni"};
      new int[][]{{10,20,30},{40,50},{60}}

eg#1.
class Test
{
      public static void main(String[] args)
      {
            System.out.println("The sum is :: "+sum(new int[]{10,20,30,40,50}));
```

```
	}
	static int sum(int[] arr)
	{
		int total = 0;
		for(int data : arr)
		{
			total+=data;
		}
		return total;
	}
}
```

Output

```
D:\OctBatchMicroservices>javac Test.java
D:\OctBatchMicroservices>java Test
The sum is :: 150
```

CommandLineArguments

++++++++++++++++++++

=> These are the arguments passed in the command line from the programmer to the main().

=> Any type of arguments can be passed from the command line like int,float,char,double,String.....

=> JVM will collect the arguments passed by the programmer and creates an Ananomyous array of type String.

=> JVM will call main() by passing Ananmyous array as the argument.

=> Signature of main()

public => jvm should access the main() without any authorization and authentication so make it as public.

static => jvm should not create an object of the class which contains main(), it should directly call main,so mark main() as static

void => jvm will not return anything to o.s so we need to mark the return type as void

main(String[] args) => String[] args :: it refers to command line arguments which the jvm will use to store the arguments sent by the user.

```
eg: java Test sachin ramesh tendulkar
      |
    Test.main(new String[]{"sachin","ramesh","tendulkar"})

eg:: java Test sachin 10 true
      |
     Test.main(new String[]{"sachin","10","true"});

eg:: java Test 54.5 true 10
      |
      Test.main(new String[]{"54.5","true","10"})


eg#1.
class Test
{
	//Pre-Defined Method[Entry point/Driving Code]
	public static void main(String[] args)
	{
		System.out.println(args);
		System.out.println("The length of command line arguments is "+args.length);

		for(String data:args)
```

```
            {
                  System.out.print(data+"\t");
            }
            System.out.println();
      }
}
```

```
Output
D:\OctBatchMicroservices>javac Test.java
D:\OctBatchMicroservices>java Test
[Ljava.lang.String;@76ed5528
The length of command line arguments is 0

D:\OctBatchMicroservices>java Test sachin ramesh tendulkar
[Ljava.lang.String;@76ed5528
The length of command line arguments is 3
sachin  ramesh  tendulkar

D:\OctBatchMicroservices>java Test 10 53.4 true 180000
[Ljava.lang.String;@76ed5528
The length of command line arguments is 4
10      53.4    true    180000
```

Write a java program to peform addition of 2 numbers by taking inputs from command line?

```
class Test
{
      //Pre-Defined Method[Entry point/Driving Code]
      public static void main(String[] args)
      {
            System.out.println(args);
            System.out.println("The length of command line arguments is
"+args.length);

            int firstOperand  = Integer.parseInt(args[0]);
            int secondOperand = Integer.parseInt(args[1]);
            int result = firstOperand + secondOperand;
            System.out.println("The sum is :: "+result);

      }
}
```

```
Output
D:\OctBatchMicroservices>javac Test.java

D:\OctBatchMicroservices>java Test 10 20
[Ljava.lang.String;@76ed5528
The length of command line arguments is 2
The sum is :: 30

D:\OctBatchMicroservices>java Test 100 200
[Ljava.lang.String;@76ed5528
The length of command line arguments is 2
The sum is :: 300

D:\OctBatchMicroservices>java Test 1000 2000
[Ljava.lang.String;@76ed5528
The length of command line arguments is 2
The sum is :: 3000
```

```
eg#2.
class Test
{
	//Pre-Defined Method[Entry point/Driving Code]
	public static void main(String[] args)
	{
		String[] argh= {"A","B"};
		args = argh;

		System.out.println(args.length);//2
		for (int i=0;i<args.length ;i++ )
		{
			System.out.println(args[i]);
		}

		for(String data : args)
		{
			System.out.println(data);
		}
	}
}
//java Test X Y  ---> Test.main(new String[]{"X","Y"})
output :: 2 A B A B
//java Test      ---> Test.main(new String[] {})
output :: 2 A B A B

eg#3.
class Test
{
	//Pre-Defined Method[Entry point/Driving Code]
	public static void main(String[] args)
	{
		String[] names = {"sachin","saurav","dhoni","kohli"};//names is Array
of String
		System.out.println(names.length);//4
		System.out.println(names[0].length());//6
		System.out.println(names[2].length());//5
	}
}

++++++++++++++++++
Types of Variables
++++++++++++++++++
 What is variable?
  => It is an identifier or name given to memory location which holds our data.

How many type of variables are there in java language?
 There are 2 types of variables
	a. Based on the type of value variable holds
	b. Based on the behaviour and position of its declaration

Based on the type of value variable holds
  -> We have 2 types
	a. primitive type
	b. reference type

Primitive type
 => These are the variables which holds primitive values
```

eg: int x= 10; boolean isMarried = false; double avg=53.5;

Reference type
 => These are the variables which holds the address of the objects, or these varaible are used to refer the objects
eg: Student std = new Student();
Employee emp = new Employee();

Based on the behaviour and position of its declaration
-> we have 3 types
a. instance variable
b. static variable
c. local variable

a. instance variable
These are varaibles which are written inside the class,but outside the method.
instance variables are such variables whose value changes from object to object.
instance variables are created at the time of object creation and destroyed at the time of object destruction(GC)
instance variables will be stored in the heaparea of the object[seperate copy for each object].
instance varaibles can be accessed directly from instance area in instance method.
instatnce variables are accessed through reference from static area in static method.

eg#1.
```
class Student
{
	//instance variables
	String name;
	int age;
	char gender;

	public void dispStdDetails()
	{
		System.out.println("Name   is :: "+name);
		System.out.println("Age    is :: "+age);
		System.out.println("Gender is :: "+gender);
	}
}
class Test
{
	//Pre-Defined Method[Entry point/Driving Code]
	public static void main(String[] args)
	{
		Student s1 = new Student();
			s1.name = "sachin";
			s1.age  = 51;
			s1.gender='M';
			s1.dispStdDetails();

			System.out.println();

		Student s2 = new Student();
			s2.name="SmritiMandana";
```

```
                s2.age = 27;
                s2.gender='F';
                s2.dispStdDetails();
        }
}
Output
D:\OctBatchMicroservices>javac Test.java

D:\OctBatchMicroservices>java Test
Name   is :: sachin
Age    is :: 51
Gender is :: M

Name   is :: SmritiMandana
Age    is :: 27
Gender is :: F

eg#2.
class Test
{
        int i = 100;
        //Pre-Defined Method[Entry point/Driving Code]
        public static void main(String[] args)
        {
                System.out.println(i);//CE
                new Test().disp();
        }
        public void disp()
        {
                System.out.println(i);//100
        }
}

eg#3.
class Test
{
        boolean isMarried;

        //Pre-Defined Method[Entry point/Driving Code]
        public static void main(String[] args)
        {
                System.out.println(new Test().isMarried);//false
        }

}

eg#4.
class Test
{
        //Array declaration
        int[] arr;

        //Pre-Defined Method[Entry point/Driving Code]
        public static void main(String[] args)
        {
                System.out.println(new Test().arr);//null
                System.out.println(new Test().arr[0]);//NPE
        }
}
```

static variable
	These are varaibles which are written inside the class,but outside the method with an access modifer called "static".
	 static variables are such variables whose value does not changes from object to object[unique copy].
	static variables are created at the time of loading the .class file and destroyed at the time of unloading the .class file.
	static variables will be stored in the methodArea[Common copy for all the objects of the class].
	 static varaibles can be accessed directly inside instance or static area.
	static variables should be accessed using classname or object name,but good practise is through "classname".

eg#1.

```
class Student
{
	//static variable
	static String nationality = "INDIAN";

	//instance variables
	String name;
	int age;
	char gender;

	public void dispStdDetails()
	{
		System.out.println("Name         is :: "+name);
		System.out.println("Age          is :: "+age);
		System.out.println("Gender       is :: "+gender);
		System.out.println("Nationality  is :: "+nationality);
	}
	public static void dispNationality()
	{
		System.out.println("Nationality  is :: "+nationality);
	}
}

class Test
{
	//Pre-Defined Method[Entry point/Driving Code]
	public static void main(String[] args)
	{
		Student s1 = new Student();
			s1.name = "sachin";
			s1.age  = 51;
			s1.gender='M';
			s1.dispStdDetails();

			System.out.println();

		Student s2 = new Student();
			s2.name="SmritiMandana";
			s2.age = 27;
			s2.gender='F';
			s2.dispStdDetails();

		System.out.println("***********Static Area*************");
```

```
            System.out.println("Nationality is :: "+Student.nationality);
            System.out.println("Nationality is :: "+s1.nationality);
            System.out.println("Nationality is :: "+S2.nationality);

      }
}
Output
D:\OctBatchMicroservices>javac Test.java
D:\OctBatchMicroservices>java Test
Name         is :: sachin
Age          is :: 51
Gender       is :: M
Nationality  is :: INDIAN

Name         is :: SmritiMandana
Age          is :: 27
Gender       is :: F
Nationality  is :: INDIAN

***********Static Area*************
Nationality is :: INDIAN
Nationality is :: INDIAN
Nationality is :: INDIAN

Q>
class Test{
      static int i =10;
      public static void main(String[] args)
      {
            System.out.println(i);
            System.out.println(new Test().i);
            System.out.println(Test.i);
      }
}
Answer: 10  10 10

Q>
class Test{
      static String name;
      public static void main(String[] args)
      {
            System.out.println(name);
      }
}
Answer : null

Q>
class Test{
      int x =10;
      static int y= 20;
      public static void main(String[] args)
      {
            Test t = new Test();
            t.x = 888;
            t.y = 999;
            Test t2 =new Test();
            System.out.println(t.x + " "  + t.y);
            System.out.println(t2.x + " " + t2.y);
      }
```

```
}
```

Q>
Given:

```
public class Yippee {
      public static void main(String [] args) {
            for(int x = 1; x < args.length; x++) {
                  System.out.print(args[x] + " ");
            }
      }
}
```

and two separate command line invocations:

```
java Yippee
java Yippee 1 2 3 4
```

What is the result?

A. No output is produced.
   1 2 3
B. No output is produced.
   2 3 4
C. No output is produced.
   1 2 3 4
D. An exception is thrown at runtime.
   1 2 3
E. An exception is thrown at runtime.
   2 3 4
F. An exception is thrown at runtime.
   1 2 3 4

Answer: java Yippee 1 2 3 4

```
           |
          Yippee.main(new String[]{"1","2","3","4"})
            args[0] = "1"
            args[1] = "2"
            args[2] = "3"
            args[3] = "4"
```

Answer: B

Q>
Given:

```
1. class Alligator {
2.    public static void main(String[] args) {
3.          int[] x[] = { { 1, 2 }, { 3, 4, 5 }, { 6, 7, 8, 9 } };
4.          int[][] y = x;
5.          System.out.println(y[2][1]);
6.    }
7. }
```

What is the result?

A. 2
B. 3
C. 4
D. 6
E. 7
F. Compilation fails.

Answer: E