```
length vs length()
==================
int[] x = new int[3];
 System.out.println(x);//[I@...
 System.out.println(x.length);//3
 System.out.println(x.length());//CE

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

Note: length is a property applicable only on array type varaibles whereas length() is a method applicable only on "String" objects.

Q>

```
int[][] arr =new int[2][4];
arr[0] =  new int[]{1,3,5,7};
arr[1] = new int[]{1,3};
for(int[] a : arr){
      for(int i:a){
            System.out.print(i+" " );
      }
      System.out.println();
}
```

```
A. Compilation fails
B. 1 3
   1 3
C. 1 3
   followed by AIOBE
D. 1 3
   1 3 0 0
E. 1 3 5 7
   1 3
F. 1 3 5 7
   1 3 0 0
```

Answer : E

For the above code predict how many objects are created and how many are eligible for garbage collection?

```
int[][] a=new int[3][2];
a[0]=new int[3];
a[1]=new int[4];
a=new int[4][3];
```

```
A. object crated = 3
    eligible for gc = 3
B. object crated = 11
    eligible for gc = 6
C. object crated = 10
    eligible for gc = 5
D.object crated = 10
    eligible for gc = 10
```

Answer: B

What is the nature of the following code?

```
class Test {
      int[] a; //a=null
public static void main(String[] args)
      {
            Test t1=new Test();
            System.out.println(t1.a);
            System.out.println(t1.a[0]);
      }
}
A. Compile Time Error
B.  0
      0
C. [I@...
      0
D. null
   ArrayIndexOutOfBoundsException
E. null
   NullPointerException
```

Answer: E

Q> Predict the nature of the following code

```
class Test {
      public static void main(String[] args) {
             int[] a;
            System.out.println(a);
            System.out.println(a[0]);
      }
}
A. Compile Time Error
B.  0
      0
C. [I@...
      0
D.  null
      ArrayIndexOutOfBoundsException
E. null
     NullPointerException
```

Answer: A

Note:
 variables in java are categorised into 3 types
      a. static varaible[memory in heap area,so default value will be given by jvm based on datatype]
      b. instance variable[memory in heap area,so default value will be given by jvm based on datatype]
      c. local variable[memory inside stack area,default value wil not be given, user should initialize before using the variable]

Q>
Predict the following output for the given code

```
public class TestApp{
      public static void main(String... args){
                  String[] arr[] ={{"%","***"},{"!!!!","@@@@@","######"}};

                  for(String str[] : arr){
                        for(String s:str)
                        {
                              System.out.println(s);
                              if(s.length()==4)
                                    break;
                        }
                        break;
                  }
      }
}
```
A. Compile Time Error
B. StringIndexOutOfBoundsException
C. %
```
   ***
```
D. %
```
   ***
   ######
```
E. ArrayIndexOutOfBoundsException
E. None of the above

Answer: C

Q>
Consider below code:
```
//Test.java
public class Test {
    public static void main(String [] args) {
      boolean flag = !true;
        System.out.println(!flag ? args[0] : args[1]);
    }
}
```
What will be the result of compiling and executing Test class using below commands?
```
javac Test.java
java Test AM PM
```

A. AM
B. PM
C. ExceptionInitalizerError while loading the .class file
D. CompilationError

Answer : A

++++++++++++++++
Ananomyous array
++++++++++++++++
 => sometimes we create an array without a name,such type of nameless array is refered as "Ananomyous array".
 => The main objective of ananomyous array is "just for instant use".

```
eg: int[] a ={10,20,30,40};
     System.out.println(a);
     System.out.println(a[0]);
     System.out.println(a.length);
```

eg::
```
public class Sample
{
      public static void main(String[] args)
      {
            System.out.println(sum(new int[]{10,20,30,40}));//100

      }
      public static int sum(int[] a)
      {
            int total = 0;
            for(int i : a)
                  total+=i;

            return total;
      }
}
```

```
+++++++++++++++++++++++
Command line arguments
+++++++++++++++++++++++
```

These are arguments which are passed in the command line by the programmer to jvm,where jvm will pass these values to main() by
creating an ananomyous array filled with the programmer supplied arguments.

```
public class Demo
{
      public static void main(String[] args)
      {

      }
}

javac Demo.java ===> Demo.clas

case1::
java  Demo =====> Demo.main(new String[]{})

case2::
java Demo 10 20 30======> Demo.main(new String[]{"10","20","30"})

case3::
java Demo sachin ramesh tendulkar ====> Demo.main(new String[]
{"sachin","ramesh","tendulkar"});
```

```
+++++++++
Snippets
+++++++++

byte ====> short =====> int ======> long =====> float ====> double
                  ^
                  |
                 char

int[] a= new int[10];
a[0] = 96;

a[1] = 'a';

byte b= 10;
a[2] = b;

short s =20;
a[3] = s;
```

```
a[4] = 10L;//invalid

eg2::
 Object[] obj = new Object[5];
 obj[0] = new Integer(10);
 obj[1] = new Object();
 obj[2] = new String("sachin");

 Number[] num =new Number[3];
  num[0] = new Integer(10);
  num[1] = new Double(10.5);
  num[2] = new String("sachin");//CE

eg3::
 Runnable[] r =new Runnable[3];
  r[0] = new Thread();
  r[1] = new String("sachin");//CE


eg4::
 int[] a = {10,20,30};
 char[] ch= {'a','b','c'};
 int[] b= a;
 int[] c= ch;//CE

eg5::
  String[] s ={"sachin","dhoni","kohli"};
  Object[] o = s;

eg6::
 int[] a= {10,20,30,40,50};
 int[] b= {60,70};
  a = b;
  b = a;

eg::
 int[][] a =new int[3][];//2D -> 1D + element
 a[0] = new int[4][5];//invalid
 a[0] = 10;//invalid
 a[0] = new int[5];//valid

eg::
 int[][] a= new int[3][2];
   a[0] = new int[3];//valid
   a[1] = new int[4];//valid
   a = new int[4][3];//valid


Q>
What will be the result of compiling and executing Test class?
public class Test {
    public static void main(String[] args) {
        String fruit = new String(new char[] {'M', 'a', 'n', 'g', 'o'});//Mango
        switch (fruit) {
            default:
                System.out.println("ANY FRUIT WILL DO");
            case "Apple":
                System.out.println("APPLE");
            case "Mango":
                System.out.println("MANGO");
            case "Banana":
                System.out.println("BANANA");
                break;
        }
    }
}
A. ANY FRUIT WILL DO
B. MANGO
C. MANGO
   BANANA
D. ANY FRUIT WILL DO
   APPLE
   MANGO
   BANANA
```

Answer: C

Q>What will be the result of compiling and executing Test class?

```
public class Test {
    public static void main(String[] args) {
        String fruit = "mango";
        switch (fruit) {
            default:
                System.out.println("ANY FRUIT WILL DO");
            case "Apple":
                System.out.println("APPLE");
            case "Mango":
                System.out.println("MANGO");
            case "Banana":
                System.out.println("BANANA");
                break;
        }
    }
}
A. ANY FRUIT WILL DO
B. MANGO
C. MANGO
   BANNANA
D. ANY FRUIT WILL DO
   APPLE
   MANGO
   BANNANA
```

Answer: D

Q>
What will be the result of compiling and executing Test class?

```
public class Test {
    public static void main(String[] args) {
        System.out.println("Output is: " + 10 != 5);
    }
}
A. Output is : true
B. Output is : false
C. Compilation error
D. Output is : 10 !=5
```

Answer: C(String object we can't use equality operator)

Q>
What will be the result of compiling and executing Test class?

```
public class Test {
    public static void main(String[] args) {
        System.out.println("Output is: " + (10 != 5));
    }
}
```

A. Output is : true
B. Output is : false
C. Compilation error
D. Output is : 10 !=5

Given

```
1. class Zippy {
2.    String[] x;
3.    int[] a [] = {{1,2}, {1}};
4.    Object c = new long[4];//Address of any object can be stored in reference
variable of type Object
5.    Object[] d = x;
6. }
```

What is the result?
A. Compilation succeeds.
B. Compilation fails due only to an error on line 3.
C. Compilation fails due only to an error on line 4.
D. Compilation fails due only to an error on line 5.
E. Compilation fails due to errors on lines 3 and 5.
F. Compilation fails due to errors on lines 3, 4, and 5.

Answer: A

Q>
Given

```
 class Dims {
2. public static void main(String[] args) {
3.    int[][] a = {{1,2,}, {3,4}};
4.    int[] b = (int[]) a[1];
5.    Object o1 = a;
6.    int[][] a2 = (int[][]) o1;
7.    System.out.println(b[1]);
8. }
}
```

What is the result?
A. 2
B. 4
C. An exception is thrown at runtime
D. Compilation fails due to an error on line 4.
E. Compilation fails due to an error on line 5.
F. Compilation fails due to an error on line 6.