How many keywords are there in java?
  Ans. 53

Reserve words meant for datatypes

a. byte,short,int,long
b. float,double
c. char
d. boolean

Reserve words for access modifiers(11)

a. public
b. private
c. protected
d. static
e. strictfp
f. synchornized
g. abstract
h. native
i. transient
j. volatile
k. final

What is Object?
 It is an instance of a class.

What is class?
 It is a blueprint as to how object should look upon creation.

Keywords related to object
 a. new

How many ways are available to write a method in java program?
 Ans. 4 ways

1. Method with no input and no output[void m1(){}]
2. Method with input and no output [void m1(XXXXX a, XXXX b){}]
3. method with input and output[XXXX m1(XXXX a, XXXX y){ return XXXX;}]
4. method without input and with output [XXXX m1(){ return XXXX;}]

What is the difference b/w Arguments and Parameter?
 Ans. Arguments => When we are calling a method, if we pass data , those data we call as "Arguments".
      Parameters => When we write a method body, the varaibles used to collect the value is called "Parameters".

What is the skeletal structure of a java program?

```
 class .......
 {
      public static void main(String[] args)
      {
            //method body
      }
 }
```

+++
Execution of Java Program(behind the scenes)
+++

```
class Student
{
      //properties/fields/attributes
      String name;
      int age;

      //methods/behaviours
      public void dispStdDetails()
      {
            System.out.println(name);
            System.out.println(age);
      }
}
class Test
{
      public static void main(String[] args)
      {
            Student s = new Student();
             s.dispStdDetails();
      }
}

D:\OctBatchMicroservices>javac Test.java
 Upon compilation of Test.java file 2 .class files will be loaded
  a. Test.class[main()]
  b. Student.class

D:\OctBatchMicroservices>java Test
  JVM will load Test.class file by searching in the CWD[D:\OctBatchMicroservices>]
  JVM found Test.class file which contains main() so JVM started the execution
  JVM will split JRE into 3 regions
      a. MethodArea[To keep .class file details]
      b. StackArea[To keep the method body for execution]
      c. HeapArea[To keep object data with default values for properties based on
datatype]

  Upon loading the main(), JVM started the execution
      a. loaded main() body to StackArea.
      b. Student s = new Student();
                  |=> JVM will go to heap area, create an object.
                  |=> Load Student.class file by searching in CWD[D:\
OctBatchMicroservices>] to MethodArea
                  |=> Depending upon the datatypes of properties, supply the
default values by giving a memory inside heaparea.
                  |=> Address of the object will be returned to the user.
      c. s.dispStdDetails()
                  |=> JVM will call the method
                  |=> Body of dispStdDetails() will be loaded into stack area
                  |=> Execution of statements present inside dispStdDetails will
happen
                              S.o.p(name); S.o.p(age);
                  |=> Print the details of name and age into console[monitor]
      d. Remove stackarea of dispStdDetails()
      e. Remove stackarea of main()
      d. Since Student Object doesn't have refernce, it becomes garbage object[Call
GarbageCollector[internally done by JVM]]
      f. Unload the .class files from MethodArea[Test.class,Student.class]
      g. Remove JRE /JVM from RAM.

Note: JVM by default will always search for main() with the following signature
            public static void main(String[] args)
```

```
+++++++
Arrays
+++++++
1. Introduction to Arrays
2. Array Declaration
3. Array Construction
4. Array Initalization
5. Array declaration,construction,initalization in single line
5. length vs length() method
6. Annomynous array
7. Array element assignments
8. Array variable assignments

Need of Arrays : To resolve the problem of remembering multiple variable names.

Arrays
   => It refers to index collection of fixed no of homogenous[all elements should
belong to same datatype] data elements.
   => Single variable holding mulitple values which imporves readability of the
program.

Disadvntages
  1. Once we create the size cannot be increased/decreased.
  2. It stores only homogenous data elements.


Array declarations
==================
  Single Dimension Array

Declaration of array
====================
 int[] a;//recomended to use as variable is seperated from type.
 int a[];
 int []a;

int[6] a; // compile time error. we cannot specify the size.


Array Construction
==================
   Every array in java is an object hence we create using new operator.

example
   int[]  a;
   a=new int[5];

      or
   int[] a =new int[5];


eg#1.
class Test
{
      public static void main(String[] args)
      {
            //Array declaration
            int[] a = null;
            System.out.println(a);//null
            //Array Construction
            a=new int[5];
            System.out.println(a);//[I

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

            //Array initalization
             a[0]=10;
             a[1]=20;
             a[2]=30;
             a[3]=40;
             a[4]=50;

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

      }
}
```

Output

```
D:\OctBatchMicroservices>javac Test.java
D:\OctBatchMicroservices>java Test
null
[I@76ed5528
Before initalization
0
0
0
0
0
After initalization
10
20
30
40
50
```

Note::
For every type corresponding classes are available but these classes are part of java language but not applicable at the programmer level.
These classes are also called as "proxyClasses".

```
    int[]       [I
    float[]     [F
    double[]    [D
    byte[]      [B
    boolean[]   [Z]
    short       [S
    String      [java.lang.String...
```

Rule1::
   At the time of Array construction compulsorily we should specify the size.
   example::

```
         int[] a=new int[5];
         int[] a =new int[];//ce:: array dimension is missing.

Rule2::
  It is legal to have an array with size zero.
     example::
          int[] a =new int[0];
          System.out.println(a.length);

Rule3::
   If we declare an array with negative size it would result in Negative Array size
exception.
       example::
            int[] a=new int[-5]; //NegativeArraySizeException.

Rule4::
   The allowed datatypes to specify the size are byte,short,int,char.
        example::
             int[] a =new int[5];

             byte b=10;
             int[] a =new int[b];//valid

             short s=25;
             int[] a =new int[s];//valid

             char c='A';
             int[] a=new int[c];//valid

           int[] a=new int[10L];//CE
           int[] a=new int[3.5f];//CE

Rule5:: The maximum allowed array size in java is maximum value of int size.
        int[] a=new int[2147483647]; //but valid:: OutOfMemoryError
        int[] a=new int[2147483648]; //CE

Note: During the execution, if JVM is not able to create the memory for the objects
due to insufficient memory space we call it as
      "OutofMemoryError".
      During the execution, if JVM is not able to create a memory because of
invalid size, we say such problems as "Exception".

ArrayInitialisation
===================
  Since arrays are treated as objects,internally based on the type of data we keep
inside array
  JVM will keep default values.
     eg::int[] a =new int[5];
         System.out.println(a);//[I@....
         System.out.println(a[0]);//0

     eg2:: int[] a=new int[4];
           a[0]=10; a[1]=20; a[2]=30;
           System.out.println(a[3]); //0
           System.out.println(a[4]); //ArrayIndexOutOfBoundsException.
           System.out.println(a[-4]);//ArrayIndexOutOfBoundsException.

++++++++
Snippets
++++++++
hint: Don't create object of Feline, directly call foo() because it is static.
class Feline {
      public static void main(String[] args) {
            Long x = 42L;
            Long y = 44L;
            System.out.print(" " + 7 + 2 + " "); // 72 foo425 86foo
            System.out.print(foo() + x + 5 + " ");
            System.out.println(x + y + foo());
}
      static String foo() {
             return "foo";
      }
}
What is the result?
A. 9 foo47 86foo
B. 9 foo47 4244foo
C. 9 foo425 86foo
D. 9 foo425 4244foo
E. 72 foo47 86foo
F. 72 foo47 4244foo
G. 72 foo425 86foo//Answer
H. 72 foo425 4244foo
I. Compilation fails.

Q>
class Sixties {
      public static void main(String[] args) {
            int x = 5;
            int y = 7;
            System.out.print(((y * 2) % x));// 14%5 = 4
            System.out.print(" " + (y % x));// 7%5  = 2
      }
}
What is the result?
A. 1 1
B. 1 2
C. 2 1
D. 2 2
E. 4 1
F. 4 2 //Answer
G. Compilation fails.
H. An exception is thrown at runtime.

Q>
Given
int a= 8,b=15,c=4;
System.out.println( 2 * ((a%5) * (4+(b-3)/(c+2))));
What is the output?
A. 30
B. 36//Answer
C. 32
D. 35

// 2 * ( (8%5) * ( 4 + (15-3)/(4+2))))
   2 * ( (3)   * ( 4 + (12)/(6))))
   2 * ( (3)   * ( 4 + 2))
   2 * ( (3)   * 6)
   2  * ( 18)
```

36

Q>
Given

```
class TestOR {
2. public static void main(String[] args) {
3. if ((isItSmall(3)) || (isItSmall(7))) {
4.    System.out.println("Result is true");
5. }
6. if ((isItSmall(6)) || (isItSmall(9))) {
7.    System.out.println("Result is true");
8. }
9. }
10.
11. public static boolean isItSmall(int i) {
12.   if (i < 5) {
13.         System.out.println("i < 5");
14.         return true;
15.   } else {
16.         System.out.println("i >= 5");
17.         return false;
18.   }
19.   }
20. }
```

What is the result?
A. Compilation Error at line 3
B. Compilation Error at line 6
C. i<5
   Result is true
D. i<5
   Result is true
   i>=5
   Result is true
E. i<5
   Result is true
   i>=5
   i>=5
   Result is true
F. i<5
   Result is true
   i>=5
   i>=5