```
+++++++++++++++
static variables
+++++++++++++++
 => variables for whom the memory is given at the time of loading .class file is
called
    "static variable".
 => To create a static variable we need to use "static" keyword.
 => To access static variable/static methods we don't need "instance" of an
object,just with
    classname we can access "static variables/static methods".
 => static methods are called as "utility methods".
 => For static variables memory would be shared for all the objects,it means only
one copy
    would be created for all the objects of a particular class.
 => static variables are also called as "class variables".
 => For static variables also jvm will give default value based on the datatype.

public class Sample
{
      static int i;//memory will be given during loading of .class file

      static
      {
            System.out.println("loading of .class file");
      }
      Sample(){
            System.out.println("Object is instantiated");
      }
      public static void main(String[] args)
      {
            Sample s =new Sample();
            System.out.println(s.i);//0

            System.out.println(Sample.i);//0
            System.out.println(i);//0

      }
}

eg#2.
public class Sample
{
      static String s;
      public static void main(String[] args)
      {
            System.out.println(s);//null
      }
}

eg#3.
public class Sample
{
      int x= 10;
      static int x = 20;//CE
      public static void main(String[] args)
      {
            int x= 30;
            System.out.println(x);
      }
```

```
}
```

Question

```
1. public class BuildStuff {
2.    public static void main(String[] args) {
3.          Boolean test = new Boolean(true);//true
4.          Integer x = 343;
5.          Integer y = new BuildStuff().go(test, x);
6.          System.out.println(y);
7.    }
8.    int go(Boolean b, int i) {
9.          if(b)
              return (i/7);
10.         return (i/49);
11.   }
12.}
```

What is the result?

A. 7
B. 49
C. 343
D. Compilation fails.
E. An exception is thrown at runtime.

Answer: B

Given this code from Class B:

```
25. A a1 = new A();
26. A a2 = new A();
27. A a3 = new A();
28. System.out.println(A.getInstanceCount());
```

What is the result?

```
1. public class A{
2.
3. private int counter = 0;// instance variable
4.
5. public static int getInstanceCount() {//static area
6.    return counter;
7. }
8.
9. public A() {
10.    counter++;
11.}
12.
13.}
```

A. Compilation of class A fails.
B. Line 28 prints the value 3 to System.out.
C. Line 28 prints the value 1 to System.out.
D. A runtime error occurs when line 25 executes.
E. Compilation fails because of an error on line 28.

Answer: A

Q>
Given:

```
String[] elements = { "for", "tea", "too" };
String first = (elements.length > 0) ? elements[0] : null;
```

What is the result?
A. Compilation fails.
B. An exception is thrown at runtime.
C. The variable first is set to null.
D. The variable first is set to elements[0].//first = "for".

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

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

```
public class Breaker {
      static String o = "";
      public static void main(String[] args) {
            z:
                o = o + 2; // z:: refers to label(label can be declared without
datatype)
            for (int x = 3; x < 8; x++) {
                  if (x == 4)
                        break;
                  if (x == 6)
                        break z;
                  o = o + x;
            }
            System.out.println(o);
      }
}
```
What is the result?
A. 23
B. 234
C. 235
D. 2345
E. 2357
F. 23457
G. Compilation fails.

Given
1. public class KungFu {
2. public static void main(String[] args) {

```
3.    Integer x = 400;
4.    Integer y = x;
5.    x++;
6.    StringBuilder sb1 = new StringBuilder("123");
7.    StringBuilder sb2 = sb1;
8.    sb1.append("5");
9.    System.out.println((x == y) + " " + (sb1 == sb2));
10. }
11.}
```

What is the result?

A. true true
B. false true
C. true false
D. false false
E. Compilation fails.
F. An exception is thrown at runtime.

Answer: B

Given

```
class Converter {
public static void main(String[] args) {
      Integer i = args[0]; // line 13
      int j = 12;
      System.out.println("It is " + (j == i) + " that j==i.");
}
}
```

What is the result when the programmer attempts to compile the code and run it with the command

java Converter 12?

A. It is true that j==i.
B. It is false that j==i.
C. An exception is thrown at runtime.
D. Compilation fails because of an error in line 13.

Answer:D

Q>

Given

```
1. public class Venus {
2.    public static void main(String[] args) {
3.          int[] x = { 1, 2, 3 };
4.          int y[] = { 4, 5, 6 };
5.          new Venus().go(x, y);
6.     }
7.
8.    void go(int[]... z) {
9.          for (int[] a : z)
10.               System.out.print(a[0]);
11.   }
12.}
```

What is the result?

A. 1
B. 12
C. 14
D. 123
E. Compilation fails.
F. An exception is thrown at runtime.

Answer: C

Q>
Given

```
 public class Test{
      Boolean b[] = new Boolean[2];
      public static void main(String... args){
            Test t= new Test();
            System.out.println(t.b[0] + ":" +t.b[1]);
      }
 }
```

A. NullpointerException
B. false:false
C. true:true
D. null:null
E. RunTimeException other than NullPointerException

Answer: D

Given:

```
1. public class GC {
2.    private Object o;
3.    private void doSomethingElse(Object obj) { o = obj; }
4.    public void doSomething() {
5.          Object o = new Object();
6.          doSomethingElse(o);
7.          o = new Object();
8.          doSomethingElse(null);
9.          o = null;
10.   }
11.}
```

When the doSomething method is called, after which line does the Object created in line 5 become available for garbage collection?
A. Line 5
B. Line 6
C. Line 7
D. Line 8
E. Line 9
F. Line 10

Answer: D

What is methodOverloading?
 => Methods with samename but change in the argument type is refered as "MethodOverloading".
 => MethodOverloading reduces the complexity of the programming.
 => Overloading is refered as "CompileTime" polymorphism.
 => Binding of the method call will be based on the reference time,but not on the runtime object,so it is called as
    CompileTimeBinding/eager binding/earlybinding.

eg#1.

```
public class Sample
{
      public void m1(){
```

```
            System.out.println("No arg method");
      }
      public void m1(int i){
            System.out.println("int arg method");
      }
      public      void m1(double d){
            System.out.println("double arg method");
      }
      public static void main(String[] args)
      {
            Sample s  =new Sample();
            s.m1();//no-arg
            s.m1(10);//int-arg
            s.m1(10.5);//double-arg
      }
}

eg#2.
public class Sample
{
      public void m1(int i){
            System.out.println("int arg method");
      }
      public      void m1(float d){
            System.out.println("float arg method");
      }
      public static void main(String[] args)
      {
            Sample s  =new Sample();
            s.m1('a');//int-arg
            s.m1(19L);//float-arg
            s.m1(10.5);//CE
      }
}
eg#3.
public class Sample
{
      public void m1(int i){
            System.out.println("int arg method");
      }
      public void m1(byte b){
            System.out.println("byte arg method");
      }
      public      void m1(Integer i){
            System.out.println("Integer arg method");
      }
      public void m1(Number n){
            System.out.println("Number arg method");
      }
      public static void main(String[] args)
      {
            Sample s = new Sample();
            Byte b1 = 25;
            s.m1(b1);//Number arg method
      }
}
sequence of binding
==================
1.primitive -> exact match-> not found perform type casting
```

2.wrapper   -> exact match -> bind -> not found -> type cast to child/parent -> child/parent not found-> work with primtive and
  perform type casting.

```
// byte---> short--->int ---> long ---> float ---> double
//                     ^
//                     |
//                    char
```

eg#3.

```
public class Sample
{
	public void m1(String s){
		System.out.println("String version");
	}
	public void m1(Object o){
		System.out.println("Object version");
	}


	public static void main(String[] args)
	{
		Sample s = new Sample();
		s.m1("sachin");//string-version
		s.m1(new Object());//object-version
		s.m1(null);//null (Default value for any type of object)//string-version
	}
}
```

eg#5.

```
public class Sample
{
	public void m1(String s){
		System.out.println("String version");
	}
	public void m1(Object o){
		System.out.println("Object version");
	}
	public void m1(StringBuilder o){
		System.out.println("StringBuilder version");
	}
	public static void main(String[] args)
	{
		Sample s = new Sample();
		s.m1("sachin");//string-version
		s.m1(new Object());//Object-version
		s.m1(new StringBuilder("sachin"));//StringBuilder-version
		s.m1(null);//default value for any type of object[CE]
	}
}

// Object(p) --> String and StringBuilder child classes
// String and StringBuilder are siblings
```