Integer i1=1;

Integer i2=new Integer(1);

Int i3=1;

Byte b1=1;

Long g1=1L;

`i1 == i2` will return false because both are pointing to different object.

`i1 == i3` will return true because one operand is a primitive int and so the other will be unboxed and then the value will be compared.

`i1 == b1` will not even compile because type of i1 and b1 references are classes that are not in the same class hierarchy. So == knows at compile time itself that they can't point to the same object.

`i1.equals(i2)` will return `true` because both are Integer objects and both have the value 1.

`i1.equals(b1)` and `i1.equals(g1)` will return false because they are pointing to objects of different types.

Which of the following statements are true?

- **The modulus operator % can only be used with integer operands.**

(It can be used on floating points operands also. For example, 5.5 % 3 = 2.5)

- **& can have integral as well as boolean operands.**

(unlike &&, & will not "short circuit" the expression if used on boolean parameters.)

- **&& can have integer as well as boolean operands.**

(!, && and || operate only on booleans.)

- **~ can have integer as well as boolean operands.**

(~ Operates only on integral types)

Given :

```
//In file AccessTest.java

package a;

public class AccessTest {

int a;

private int b;

protected void c(){ }

public int d(){  return 0; }

}
```

```
//In file AccessTester.java

package b;

import a.AccessTest;


public class AccessTester extends AccessTest{

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

```
    AccessTest ref = new AccessTest();


  }
}
```

Idenfity the correct statements

CAVAB : Only d() can be accessed by ref.

Biraz casdirici sualdir.Cunki ilk baxsida belke de dusunerikki c metodu da accesabledir.Cunki bilirkki protected accese sahib metod eyni paketden ve ya diger paketdedirse torendiyi classdan yox subclass referansindan ell catandir! Yeni Burada AccessTest referansindan el catan deyil !!

```
public class StringTest {

    public static void main(String[] args) {

        String s1 = new String("pankaj");

        String s2 = new String("PANKAJ");

        System.out.println(s1 = s2);

    }

}
```

It's a simple yet tricky program, it will print "PANKAJ" **because we are assigning s2 String to s1.** **Don't get confused with == comparison operator.**

***** EGER = evezine == olsaydi false ekrana verecekdi !**

***** EGER String obyekt evezine sadece** String s1 = “pankaj” ve String s2 =”PANKAJ” yazsaydiq cavab **true olacaqdi cunki her ikisi string pool-u yeni eyni obyekti gosterir !!!**

```
String s1 = new String("Hello");

String s2 = new String("Hello");
```

Bu zaman arxada ne bas verir ?

1)Evvelce “Hello” obyekti string pool-da yaradilir
2)Heap-de yeni string obyekti “Hello” deyeri ile
3)Heap-de yeni string obyekt “Hello” deyeri ile (“Hello” deyeri tekrar olaraq string pool-dan gotuturlur yene de)

```
        final String s1="job";
        final String s2="seeker";
        String s3=s1.replace("j","R");
        String s4="jobseeker";
    System.out.println(s3==s4); // Output
```

Burada output false olacaq **. Cunki string1+string2 , replace , concat ve.s kimi emelyatlar geriye String qaytarir ve burada artiq s3 pool-da olmayacaq yeni obyekt kimi heap-de tutulacaq !!!**

```
String s1 = "Java";
String s2 = "Java";
StringBuilder sb1 = new StringBuilder();
sb1.append("Ja").append("va");

System.out.println(s1 == s2);
System.out.println(s1.equals(s2));
System.out.println(sb1.toString() == s1);
```

System.out.println(sb1.toString().equals(s1));

A. true is printed out exactly once.
B. true is printed out exactly twice.
C. true is printed out exactly three times.
D. true is printed out exactly four times.
E. The code does not compile.

CAVAB C

1 ve 2 true-dur cunki her referans olaraq (string pool) hem de deyer olaraq(Java) eynidirler.
3-cu ifadede toString() var o ise deyerleri muqayise edir ona gore false qayidir.
Son ifadede ise equals deyerleri yoxclayir true qayidir

What is the result of the following class? (Choose all that apply)
1: public class _C {
2: private static int $;
3: public static void main(String[] main) {
4: String a_b;
5: System.out.print($);
6: System.out.print(a_b);
7: } }

A. Compiler error on line 1.
B. Compiler error on line 2.
C. Compiler error on line 4.
D. Compiler error on line 5.
E. Compiler error on line 6.
F. 0null
G. nullnull

CAVAB :E

Lokal deyiskenler (metod daxilinde yerlesen deyiskenler (public static void main de daxil) mutleq deyer menimsedilemlidir

What is the output of the following code snippet?
13: System.out.print("a");
14: try {
15: System.out.print("b");
16: throw new IllegalArgumentException();
17: } catch (RuntimeException e) {
18: System.out.print("c");
19: } finally {
20: System.out.print("d");
21: }
22: System.out.print("e");

A. abe
B. abce
C. abde
D. abcde
E. The code does not compile.
F. An uncaught exception is thrown.

Cavab D :
Evvelce a ekrana verilir sonra b , sonra exception throw olur ve catch-a girir .Catch-de c ekrana verilir daha sonra system.exit() olmadigi muddetce hemise finally blogu ise dusur ve d ekrana verilir sonra ise try-dan cixilir ve e ekrana verlir .Netice : abcde

What is the output of the following code?

```
1: public class Deer {
2: public Deer() { System.out.print("Deer"); }
3: public Deer(int age) { System.out.print("DeerAge"); }
4: private boolean hasHorns() { return false; }
5: public static void main(String[] args) {
6: Deer deer = new Reindeer(5);
7: System.out.println(","+deer.hasHorns());
8: }
9: }
10: class Reindeer extends Deer {
11: public Reindeer(int age) { System.out.print("Reindeer"); }
12: public boolean hasHorns() { return true; }
13: }
```

A. DeerReindeer,false
B. DeerReindeer,true
C. ReindeerDeer,false
D. ReindeerDeer,true
E. DeerAgeReindeer,false
F. DeerAgeReindeer,true
G. The code will not compile because of line 7.
H. The code will not compile because of line 12.

CAVAB : A
Bu cetin sual sayila biler.Demeli bilirikki subclassin constructorunu cagiranda parent constructoru ondan evvel ise dusur. Ona gore de evvelce Deer ekrana verilir.Daha sonra Reindeer . Ve daha sonra hasHorn metodu cagrilib

Which are true of the following code? (Choose all that apply)

```
1: import java.util.*;
2: public class Grasshopper {
3: public Grasshopper(String n) {
4: name = n;
5: }
6: public static void main(String[] args) {
7: Grasshopper one = new Grasshopper("g1");
8: Grasshopper two = new Grasshopper("g2");
9: one = two;
10: two = null;
11: one = null;
12: }
13: private String name; }
```

A. Immediately after line 9, no grasshopper objects are eligible for garbage collection.
B. Immediately after line 10, no grasshopper objects are eligible for garbage collection.
C. Immediately after line 9, only one grasshopper object is eligible for garbage collection.
D. Immediately after line 10, only one grasshopper object is eligible for garbage collection.
E. Immediately after line 11, only one grasshopper object is eligible for garbage collection.
F. The code compiles.
G. The code does not compile.

one

Cavab : C,D,F

two

Ilk veziyyet

One=two-dan sonra ise veziyyet

two

one

Bu veziyyetde ancaq g1 eligible-dir.Daha sonra her ikisi null verildiyi ucun her iki G1 ve G2 eligible olur.

What is the result of the following program?

```
1: public class Egret {
2: private String color;
3: public Egret() {
4: this("white");
5: }
6: public Egret(String color) {
7: color = color;
8: }
9: public static void main(String[] args) {
10: Egret e = new Egret();
11: System.out.println("Color:" + e.color);
12: }
13: }
```

A. Color:
B. Color:null
C. Color:White
D. Compiler error on line 4.
E. Compiler error on line 10.
F. Compiler error on line 11.

CAVAB : B

Burada ilk olaraq parametrsiz constructor cagrilir daha sonra o da this("white") ile digger constructoru ise salir.Ancaq burada color=color yazmaqla biz metod parametrini ozone menimsedirik instance color deyiskenine yox !!
Eger **this.color=color** yazsaydiq onda white alardiq/
Ancaq belede Instance variable String-in default deyeri null-dur!

Asagidaki kodda nece eded import gereksizdir ?

```
1: import java.lang.System;
2: import java.lang.*;
3: import java.util.Random;
4: import java.util.*;
5: public class ImportExample {
6: public static void main(String[] args) {
7: Random r = new Random();
8: System.out.println(r.nextInt(10));
```

9: }
10: }

CAVAB : 3

Java.lang yegane paketdirki avtomatik import olur . Kodda import yazmaq lazim deyil

java.lang avtomatik import olunur
util.* ise lazim deyil cunki Random teklkde elave olunubdur
Sadece 3-cu setir lazimdir
Cavab ise 3 eded gereksiz import var kodda

1: public class Chick {
2: private String name = "Fluffy";
3: { **System.out.println("setting field");** }
4: public Chick() {
5: name = "Tiny";
6: **System.out.println("setting constructor");**
7: }
8: public static void main(String[] args) {
9: Chick chick = new Chick();
10: **System.out.println(chick.name);** } }

CAVAB :
setting field
setting constructor
Tiny
Ilk once initialize bloklar daha sonra konstruktor ise dusur !!

```
public class InitTest{
   static String s1 = sM1("a");
   static{
      s1 = sM1("b");
   }
   static String s2 = sM1("c");
   public InitTest(){
      s1 = sM1("1");
```

```
   }
   String s3 = sM1("2");{
      s1 = sM1("3");
   }
   String s4 = sM1("4");

      public static void main(String args[]){
      InitTest it = new InitTest();
      }
      private static String sM1(String s){
      System.out.println(s);  return s;
      }
}
```

OUTPUT: a b c 2 3 4 1

Cunki ilk olaraq Static blok ve deyisenler ise dusur daha sonra instance deyisekenler ve bloklar ve en sonda constructor
Static blok ve deyiskende ardiciliiga baxilir.

```
public class JustLooping {
   private int j;
     void showJ(){
     while(j<=5){
          for(int j=1; j <= 5;){
               System.out.print(j+" ");
               j++;
          }
          j++;
     }
}
public static void main(String[] args) {
     new JustLooping().showJ();
     }
}
```

Cavab : 6 defe 12345 yazacaq
Cunki ilk olaraq global deyisekene default 0 deyeri verilir.
For -da teyin edilen j=1 ancaq for icerisinde aiddir ;
While ve kenradaki j ise global deyiskeni gorur.
Ona gore sehv salmayin 6 defe ekrana verilcek 12345

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

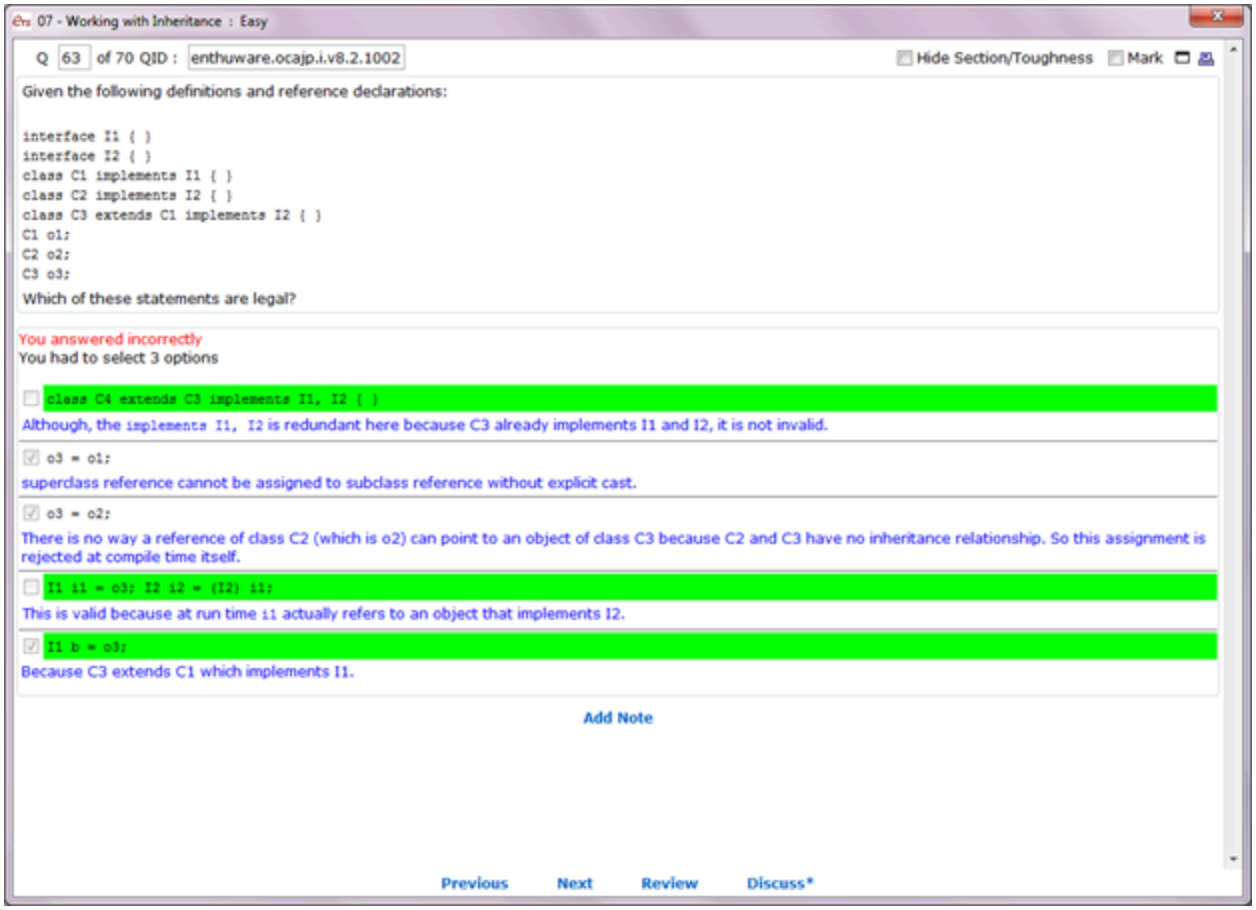
```
      int i = 0 ;
      int[] iA = {10, 20} ;
      iA[i] = i = 30 ;
         System.out.println(""+ iA[ 0 ] + " " + iA[ 1 ] + "  "+i)
;
      }
}
```
CAVAB : 30 20 30

```
public class Test{
      public static void main(String[] args){
      if (args[0].equals("open"))
      if (args[1].equals("someone"))
         System.out.println("Hello!");
      else System.out.println("Go away "+ args[1]);
      }
}
```
CAVAB : ArrayIndexOutOfArray

Cunki scopsuz if-de else ondan sonra gelene aiddir yeni burda else ilk if-in yox ikinci if-indir.

Bele olan halda birinci ifden sora kecir **if (args[1].equals("someone"))**

  Hissesine ve 1-ci element olmadigi ucun error verir




```
class A{
   A() {  print();   }
   void print() { System.out.println("A"); }
}
class B extends A{
   int i =   4;
   public static void main(String[] args){
      A a = new B();
      a.print();
   }
   void print() { System.out.println(i); }
}
```

What will be the output when class B is run ?

Cavab : 0,4

Note that method print() is overridden in class B. Due to polymorphism, the method to be executed is selected depending on the class of the actual object.

Here, when an object of class B is created, **first A's constructor is called, which in turn calls `print()`. Now, since the class of actual object is B, B's `print()` is selected. At this point of time, variable i has not been initialized (because we are still initializing A at this point), so its default value i.e. 0 is printed**.
This happens because the method `print()` is non-private, hence polymorphic.

Given the following definitions and reference declarations:

```
interface I1 { }
interface I2 { }
class C1 implements I1 { }
class C2 implements I2 { }
class C3 extends C1 implements I2 { }
C1 o1;
C2 o2;
C3 o3;
```

Which of these statements are legal?

`class C4 extends C3 implements I1, I2 { }`
Although, the implements I1, I2 is redundant here because C3 already implements I1 and I2, it is not invalid.

`o3 = o1;`
superclass reference cannot be assigned to subclass reference without explicit cast.

o3 = o2;
There is no way a reference of class C2 (which is o2) can point to an object of class C3 because C2 and C3 have no inheritance relationship. So this assignment is rejected at compile time itself.

I1 i1 = o3; I2 i2 = (I2) i1;
This is valid because at run time i1 actually refers to an object that implements I2.

I1 b = o3;
Because C3 extends C1 which implements I1.

```
class A{
   public int i = 10;
   public void  f(){}
   public void g(){ System.out.println("aaaaa"); }
}

class B extends A{
   public int i = 20;
   public void g(){ System.out.println("gggg"+super.i); }
}

public class C{

    public static void main(String[] args) throws Exception {
      A a = new A();//1
      A b = new B();//2
      b.g();
      System.out.println(b.i);
    }

}
```

Cavab : **gggg10 ve 10 olacaq**
Cunki variableller override edilmir metodlarda override vardir bunu qarsidirmayin.!!!

```
public class TestClass{
```

```
   public static void main(String[] args){  new TestClass().sayHello(); }   //1
   public static void sayHello(){ System.out.println("Static Hello World"); }  //2
   public void sayHello() { System.out.println("Hello World "); }  //3
}
```

What will be the result of compiling and running the class?

Cavab :
Compilation error at line 3.
Cunki eyni parametrleri ferqli olmadiqgi muddetce eyni adda 2 metod br java classda ola bilmez !
Ve compile error verecek ancaq 3-ci setirde cunki biz eyni adli metodu yaratnaq istediyimiz setirde error olacaq

```
 public static void main(String args[]) {
    String String = "";
    String : for(int i = 0; i< 10; i++)
    {
      for (int j = 0; j< 10; j++){
          if ( i+ j > 10 )  break String;
      }
       System.out.println( "hello");
    }
   }
```

Nece defe hello sozu ekrana verilecek ?
Cavab :2
For label ucun java keyword-lerden istifade etmek olmaz ancaq biz String istfde ede bilerik.
Burada 2 defe Hello ekrana verilcek. if (i+ j > 10) break String; -den evvel “ System.out.println("i="+i+",j="+j);” bele bir setir elave etsek output asagidaki kimi olar;

```
i=0,j=0
i=0,j=1
i=0,j=2
i=0,j=3
i=0,j=4
i=0,j=5
i=0,j=6
i=0,j=7
i=0,j=8
i=0,j=9
hello
i=1,j=0
i=1,j=1
i=1,j=2
i=1,j=3
i=1,j=4
i=1,j=5
i=1,j=6
i=1,j=7
i=1,j=8
i=1,j=9
hello
i=2,j=0
i=2,j=1
i=2,j=2
i=2,j=3
i=2,j=4
i=2,j=5
i=2,j=6
i=2,j=7
i=2,j=8
i=2,j=9
```

Ic-ice forlarda en ic for tam bitir daha sonra bir ust for counteri artrilir ve yene ic fora girilir tam bitene kimi ana forun counteri,Ve bu ardilciliqla gedir .

```
public class ChangeTest {

    private int myValue = 0;

    public void showOne(int myValue){
        myValue = myValue;
```

```
    }

    public void showTwo(int myValue){
        this.myValue = myValue;
    }
    public static void main(String[] args) {
        ChangeTest ct = new ChangeTest();
        ct.showOne(100);
        System.out.println(ct.myValue);
        ct.showTwo(200);
        System.out.println(ct.myValue);
    }
}
```

Cavab 0,200
Cunki biz instance variable-ni deyismis olmuruq myvalue=myvalue deyerek sadece parametre ozunu menimsedirik bele etmekle .Deyismek ucun this.myvalue istfde edilmelidr!

```
public class Test{
    public int luckyNumber(int seed){
        if(seed > 10) return seed%10;
        int x = 0;
        try{
            if(seed%2 == 0) throw new Exception("No Even no.");
            else return x;
            }
        catch(Exception e){
             return 3;
         }
          finally{
             return 7;
          }
    }
          public static void main(String args[]){
                int amount = 100, seed = 6;
                switch( new Test().luckyNumber(6) ){
                     case 3: amount = amount * 2;
                     case 7: amount = amount * 2;
                     case 6: amount = amount + amount;
                     default :
```

```
            }
            System.out.println(amount);
        }
}
```

Cavab : 400
Cunki burada evvelce return 3 uygun gelir ancaq finally her zaman sonda ise dusur ve return 7 edir. Switch case-de 7 hessblayanda 200 edir ancaq break istfde edimediyi ucun ondan bir sonrakilar da yerine yetirilir. Amount+amount 200+200=400.Eger break qoysaq her caseden sonra cavab 200 olardi.

```
public class LoopTest{
   public static void main(String args[]) {
      int counter = 0;
      outer:
      for (int i = 0; i < 3; i++) {
         middle:
         for (int j = 0; j < 3; j++) {
            inner:
            for (int k = 0; k < 3; k++) {
               if (k - j > 0) {
                  break middle;
               }
               counter++;
            }
         }
      }
      System.out.println(counter);
   }
}
```

Cavab :3

```
public class LoopTest{
   public static void main(String args[]) {
      int[] values = { 10, 30, 50 };
      for( int val : values ){
         int x = 0;
         while(x<values.length){
          System.out.println(x+" "+val);
           x++;
```

```
        }
      }
   }
}
```

Cavab :

```
0 10
1 10
2 10
0 30
1 30
2 30
0 50
1 50
2 50
```

```
public class TestClass{
   public static void main(String args[]){
      boolean b = false;
      int i = 1;
      do{
         i++ ;
      } while (b = !b);
      System.out.println( i );
   }
}
```

Cavab :3

Eger == olsa idi cavab 2 olard.Cunki == olanda refernaslar eyni obyektdirmi ya yox ona baxilir, Ancaq tek = ile deyeler muqayise edilir.burada ise b yeni false !b yeni true-dan ferqlidi

```
class TestClass{
      int x;
      public static void main(String[] args){
          // lot of code.
       }
}
```

Main clasinin x-i gormesi ucun x-i static etmek lazimdir. Ancaq this.x seklinde gormez
,Main classi icerisinde hec vaxt this istfde ede bilmerik !!!!

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
   public static void main(String args[]){
      int x  = 0;
      labelA:   for (int i=10; i<0; i--) {
         int j = 0;
         labelB:
         while (j < 10) {
            if (j > i) break labelB;
             if (i == j){
               x++;
               continue labelA;
             }
             j++;
           }
         x--;
      }
      System.out.println(x);
   }
}
```

Cavab :0

Bu cox sade ve casdirici sualdir. Burada cavab ona gore 0 olurki for dongusune girilmir!Cunki i<10 hec vaxt odenmir!!!

Consider the following two classes defined in two java source files.

```
//in file /root/com/foo/X.java
package com.foo;
public class X{
  public static int LOGICID = 10;
  public void apply(int i){
    System.out.println("applied");
  }
}
```

```
//in file /root/com/bar/Y.java
package com.bar;
//1 <== INSERT STATEMENT(s) HERE
public class Y{
   public static void main(String[] args){
      X x = new X();
      x.apply(LOGICID);
   }
}
```

What should be inserted at //1 so that Y.java can compile without any error?

Cavab :

```
import com.foo.*;
import static com.foo.X.*;
```

Diqqet etmeli oldugunuz yer odurki **static deyisken ve ya static class istifade eederken Import static yazilir !!!**

Syntax for importing static fields is: import static <package>.<classname>.*; or import static <package>.<classname>.<fieldname>;

Given:

```
public class FunWithArgs {
   public static void main(String[][] args) {
      System.out.println(args[0][1]);
   }
   public static void main(String[] args) {
      FunWithArgs fwa = new FunWithArgs();
      String[][] newargs = {args};
      fwa.main(newargs);
   }
}
```

The above program is compiled with the command line:
javac FunWithArgs.java
and then run with:
java FunWithArgs a b c

What will be the output?

Cavab : b

There is no problem with the code. The main method is just overloaded.
When it is run, the main method with String[] will be called. This method then calls the main with String[][] with an argument { {"a", "b", "c"} }
Thus, args[0][1] refers to "b", which is what is printed.

```
class A{
  protected int i;
  A(int i) {    this.i = i;    }

}
// 1 : Insert code here
```

Setir 1-e class B {} elave ede bileeik ancaq class B extends A {} elave ede bilmerik! Cunki SubClass contructoru ana clasin default constructorunu cagirir her zaman.A classinda ise default constructor yoxdur deye sehv verer !

What will the following class print when executed?

```
class Test{
   static boolean a;
   static boolean b;
   static boolean c;
   public static void main (String[] args){
     boolean bool = (a = true) || (b = true) && (c = true);
    System.out.print(a + ", " + b + ", " + c);
 }
}
```

Cavab :
True,false,false
a=true menimsetmedir b=true ve c=true da . Ve sert soldan saga yoxlanilir.
Ola biler deyeceksiniz cavab 3 dene true olmali idi. Xeyir cunki burda maraqli bir sey var Evvelce ve ya emelyatina baxilir a=true artiq bir teref true olduqda diger terefe baxilmir ve yada /Yeni menimsetme emelyatlari yerine yetirilmir ve defauit deyer false-dir !!!

```
public class InitTest{
   public static void main(String[] args){
      int a = 10;
      int b = 20;
      a += (a = 4);
      b += b + (b = 5);
      System.out.println(a+ ",  "+b);
   }
}
```

Cavab : 14, 45

a=4 menimsetmedir . b=5 de hemcinin. Ancaq burada toplama ederken a=a+4 demis oluruq yeni 10+4=14 eyni forma b ucun nde aiddir

```
class D { public static void main(String args[]) { Integer b2=128; Integer b3=128; System.out.println(b2==b3); } }
```

Output:

```
false
```

```
class D { public static void main(String args[]) { Integer b2=127; Integer b3=127; System.out.println(b2==b3); } }
```

Output:

```
true
```

Have a look at the Integer.java, if the value is between -128 and 127, it will use the cached pool,

```
public class TestClass{
  public static void main(String args[]){
    int x = Integer.parseInt(args[0]);
    switch(x){
      case x < 5 :   System.out.println("BIG"); break;
      case x > 5 :   System.out.println("SMALL");
      default :    System.out.println("CORRECT"); break;
   }
  }
 }
```

Cavab : It will not compile

Casdirici sualdir. Unutmayin switch deyiskeni ile case sertleri eyni tip olmalidir. Burada x int-dir ancaq x<5 x>5 boolendir /!!!! Ona gore de kod calismayacaq!!

```
class Wrapper{
       int w = 10;
}

public class TestClass{

    static Wrapper changeWrapper(Wrapper w){
        w = new Wrapper();
        w.w += 9;
        return w;
    }


    public static void main(String[] args){
        Wrapper w = new Wrapper();
        w.w = 20;
        changeWrapper(w);
        w.w += 30;
        System.out.println(w.w);
        w = changeWrapper(w);
        System.out.println(w.w);
     }
}
```

Cavab : 50,19

Cunki changeWrapper(w); her hansi obyekt referansinba menimsedilmeyib.ona gore w.w hele de 20-dir ve w.w+=30 50 edir. Ikincide menimsedilib ancaq changeWrapper metodu icerisinde biz yeni wrapper yaradiriq orada ise w deyeri 10-dur ve uzerine 9 gelemnde edir 19 !!!

What will the following code print?

```
 List s1 = new ArrayList( );
 s1.add("a");
 s1.add("b");
 s1.add("c");
s1.add("a");
if(s1.remove("a")){
   if(s1.remove("a")){
    s1.remove("b");
   }else{
     s1.remove("c");
  }
}
System.out.println(s1);
```

Cavab : [c]

ArrayList's `remove(Object )` method removes the first occurence of the given element and returns true if found. It does not remove all occurences of the element. In this case, the first call to `s1.remove("a");` will remove only the first "a" and return true, the second call to `remove("a")` will remove the second "a" and return true. Finally, the call to remove("b") will remove "b". Therefore, only "c" will be left in the list.

What will the following code print when run?

```
public class TestClass{
    public static Integer wiggler(Integer x){
       Integer y = x + 10;
       x++;
       System.out.println(x);
       return y;
    }
```

```
    public static void main(String[] args){
       Integer dataWrapper = new Integer(5);
       Integer value = wiggler(dataWrapper);
       System.out.println(dataWrapper+value);
    }
}
```

Cavab :6 20

Eger 6 515 dusunmusdunuzse bir seye diqqet edin.System.out.println(dataWrapper+value);
Iki deyiseni toplayib cemini ekrana verir ;) string kimi baxmir onlara.Bir de wiggler metodunda x++ datawrapper deyisekenini umumi artirmir her hansi instance deyiseni deyil ve ya deyer deyisib return edilmirki nese deyissin.

```
public class Test {
  static String s = "";

   public static void m0(int a, int b) {
      s += a;
    m2();
       m1(b);
  }

   public static void m1(int i) {
      s += i;
    }

   public static void m2() {
     throw new NullPointerException("aa");
   }

   public static void m() {
     m0(1, 2);
    m1(3);
  }

   public static void main(String args[]) {
      try {
       m();
```

```
      } catch (Exception e) {
      }
    System.out.println(s);
  }
}
```

Cavab : 1

Ola biler deyeceksiniz nullpinter exception throw edir axi m2. Ancaq nezere alsaqki m0-da her hansi try catch vasitesile exceptionu handle elemirik demeli davam edecek proses.

Which of these for statements are valid?

1. for (int i=5; i=0; i--) { }

2.
```
int j=5;
 for(int i=0, j+=5;  i<j ; i++) {  j--;  }
```

3.
```
int i, j;
 for (j=10; i<j; j--) { i += 2; }
```

4.
```
int i=10;
 for ( ; i>0 ; i--) { }
```

5. for (int i=0, j=10; i<j; i++, --j) {;}

Cavab: 4,5

No 1.

uses '=' instead of '==' for condition which is invalid. The loop condition must be of type boolean.

No 2.

uses 'j +=5'. Now, this statement is preceded by 'int i=0,' and that means we are trying to declare variable j. But it is already declared before the for loop. If we remove the int in the initialization part and declare i before the loop then it will work. But if we remove the statement int j = 5; it will not work because compiler will try to do j = j+5 and you can't use the variable before it is initialized. Although the compiler gives a message 'Invalid declaration' for j += 5, it really means the above mentioned thing.

No 3. i is uninitialized.

No 4. is valid. It contains empty initialization part.

No 5.
This is perfectly valid. You can have any number of comma separated statements in initialization and incrementation part. The condition part must contain a single expression that returns a boolean.
All a for loop needs is two semi colons :-
for(; ;) {} This is a valid for loop that never ends. A more concise form for the same is : for(; ;);

Consider the following class :

```
public class Parser{
   public static void main( String[] args){
      try{
          int i = 0;
          i =  Integer.parseInt( args[0] );
      }
      catch(NumberFormatException e){
         System.out.println("Problem in " + i );
      }
   }
}
```

What will happen if it is run with the following command line:
java Parser one

Cavab : It will not even compile.
Bezileriniz dusune bilerki run olacaq ve cavab Problim in one yazlacaq ancaq xeyir cunki o o zaman olardiki i deyiskeni try icerisinde tanimalnmasin. I deyiskeni try icerisinde tanimandigi ucun catch-de i -ni gormur !! Ve netcede kod compile bele ola bilmez !

```
public class Parser{
static String str;
  public static void main(String[] args){
     System.out.println(str);
  }
```

}

Cavab : null
Unutmayin String obyekt tipdi javada ve default deyer nulldur.Ancaq her hansi menimsetme varsa default deyer “” bosluq kimi goturulecek ancaq deyer menimsedilmediyi halda null-dur

Given the following LOCs:

```
  int rate = 10;
 XXX amount = 1 - rate/100*1 - rate/100;
```

What can XXX be?
Cavab : int, long, float or double

Bu cox casdirici sualdir !!! Cunki dusune bilersizki 10/100*1 ve ya 10/100 size 0.1 vercecek.Ancaq javada bu bele deyil eger biz 10.0/100 yazsaq 0.1 vererdi .
Javada bir qayda varki eger emelyat zamani deyiskenler int byte short char -dan ibaretdirse netice bize int tipinde qayidir !!! yeni amount deyiseknimiz int olmalidi. Ancaq javada int-iozunden boyuk olan deyisen tiplerine yeni long,double,floa-a da vere bilerik ona groe cavab yuxaridaki kimidir !!!

```
class TestClass{
   void probe(Object x) { System.out.println("In Object"); } //3

   void probe(Number x) { System.out.println("In Number"); } //2

   void probe(Integer x) { System.out.println("In Integer"); } //2

   void probe(Long x) { System.out.println("In Long"); } //4

   public static void main(String[] args){
      double a = 10;
      new TestClass().probe(a);
   }
}
```

What will be printed?

Cavab : in Number
Cunki double primitivinin obyekt tipi Double Numberden toreyir.Hemcinin diger number tipler de. Ona gore Object yox Number olmalidir cavab !

Which line, if any, will give a compile time error ?

```
void test(byte x){
  switch(x){
     case 'a':   // 1
     case 256:   // 2
     case 0:     // 3
     default :   // 4
     case 80:    // 5
  }
}
```

Cavab : Line 2 as 256 cannot fit into a byte.

Every case constant expression in a switch block must be assignable to the type of switch expression. Meaning :

```
byte by = 10;
switch(by){
    case 300 :  //some code;
    case 56 :   //some code;
}
```

This will not compile as 300 is not assignable to 'by ' which can only hold values from -128 to 127. This gives compile time error as the compiler detects it while compiling. The use of break keyword is not mandatory, and without it the control will simply fall through the labels of the switch statement.

Consider :

```
class A {  public void perform_work(){}  }
class B extends A {  public void perform_work(){}  }
class C extends B {  public void perform_work(){}  }
```

How can you let perform_work() method of A to be called from an instance method in C?

Cavab : It is not possible.

The method in C needs to call a method in a superclass two levels up. But `super` is a keyword and not an attribute so `super.super.perform_work( )` strategy will not work. There is no way to go more than one level up for methods.
Remember that this problem doesn't occur for instance variables because variable are never overridden. They are hidden (by instance variables of a subclass) or shadowed (by local variables or method parameters of a method). So to access any of the super class's variable, you must unhide it using a cast. For example, `((A) c).data;` This will give you the data variable defined in A even if it is hidden in B as well as in C.

What will the following code print?

```
int x = 1;
int y = 2;
int z = x++;
int a = --y;
int b = z--;
 b += ++z;
 int answ = x>a?y>b?y:b:x>z?x:z;
 System.out.println(answ);
```

Cavab : 2
a=--y ve.s zamani a deyerinden elave y-in de deyerinin deyisdiyini unutmayin!!
This is a simple but frustratingly time consuming question. Expect such questions in the exam.
For such questions, it is best to keep track of each variable on the notepad after executing each line of code.

The final values of the variables are as follows -
`x=2 y=1 z=1 a=1 b=2`

The expression `x>a?y>b?y:b:x>z?x:z;` should be grouped as -
`x > a ? (y>b ? y : b) : (x>z ? x : z);`

```
try {
    s = new ServerSocket(3030);
}
catch(Exception t){ t.printStackTrace(); }
catch(IOException e) {
```

```
    s = new ServerSocket(4040);
}
catch(Throwable t){ t.printStackTrace();  }
```

Cavab : INVALID

You can have any number of catch blocks in any order but each catch must be of a different type. Also, a catch for a subclass exception should occur before a catch block for the superclass exception. Here, IOException is placed before Throwable, which is good but Exception is placed before IOException, which is invalid and will not compile.

```
public class InitClass{
   public static void main(String args[ ] ){
      InitClass obj = new InitClass(5);
   }
   int m;
   static int i1 = 5;
   static int i2 ;
   int  j = 100;
   int x;
   public InitClass(int m){
      System.out.println(i1 + "  " + i2 + "   " + x + "  " + j + "  " + m);
   }
  { j = 30; i2 = 40; }  // Instance Initializer
   static { i1++; }      // Static Initializer
}
```

Cavab : The code will compile without error and will print 6 40 0 30 5 when run.

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
 public static void main(String args[ ] ){
   Object a, b, c ;
  a = new String("A");
  b = new String("B");
   c = a;
  a = b;
   System.out.println(""+c);
 }
}
```

Cavab : The program will print A

c=a yeni c referansi artiq a ile eyni yeni A Stringini gosterir
a=b yeni a referansi artiq b ile yeni yeni B Stringini gosterir
a=b o demek deyilki c a ile yni obyekti vaxtile gosterirdi indi artiq c de b-ni gostermlidi.bele bir sey yoxdur !!! Hal hazirki veziyetde c- A -ni a - B ni b B-ni gisterir

Which of the following declarations is/are valid:

1. bool b = null;
2. boolean b = 1;
3. boolean b = true|false;
4 bool b = (10<11);
5. boolean b = true||false;

Cavab : 3ve 5

1 4 sehvdir cunki Javada bool yoxdur boolen var
2 boolen true ve false ala bilir ancaq.

Which of the following statements can be inserted at // 1 to make the code compile without errors?

```
public class InitTest{
  static int si = 10;
  int  i;
  final boolean bool;
  // 1
}
```

Bu sualda cavab variantlari arasinda diqqet etmeli oldugunuz sey final deyisenin mutleq deyer menimsedilmeli ve menimsedildikden sonra deyeri deyise bilmerik !!!!!

```
public class CrazyMath {
    public static void main(String[] args) {
       int i = 0;
       int j = 1;
       if( (i++ == 0) & (j++ == 2) ){
           i = 12;
       }
       System.out.println(i+" "+j);
    }
}
```

Cavab : 1 2
Sert odenmese de sertde istfde edilen i++ j++ deyisenin deyerini yenileyir.Ve krana yeni deyeelr verilir.

```
System.out.println("1" + 2 + 3);
```

Cavab : 123
Coxunuz dusune bilersizki cavab 15 olacaq ancaq ele deyil!!
Cunki emelyat soldan saga aparilir ve ilk “1”+2 sonra alinan netice +3 hesablanir !!!

What will be printed by the following code if it is run with command line: java TestClass -0.50 ?

```
 public class TestClass{
  public static double getSwitch(String str){
    return Double.parseDouble(str.substring(1, str.length()-1) );     }     public static void main(String args []){
    switch(getSwitch(args[0])){
      case 0.0 : System.out.println("Hello");
       case 1.0 : System.out.println("World"); break;
        default : System.out.println("Good Bye");
   }
  }
}
```

Cavab : None of the above.

Observe that the method getSwitch() has been declared to return a double. Its return value is being used in the switch statement. Therefore, the program will not even compile because **double/float/long/boolean cannot be used in a switch statement.**

```
public class TestClass{
   public static void main(String args[]){
        int x = 10;
        do{
         x--;
         System.out.println(x);  // 1
        }while(x<10);
   }
}
```

How many times will the line marked //1 be called in the following code?

Cavab : -2147483648 -dek

Cunki do while sert odenmese bele bir defe ise dusur while sonda yerlesdiyi ucun bu zaman x 9 olur ve artiq do while her defe isleyir cunki her deyeri 10-dan kicikdir.int -in MIN_VALUE-sine kimi gedecek

```
public class TestClass {
      public static void main(String[] args){
     int k = 2;
     while(--k){
      System.out.println(k);
     }
   }
}
```

What will the following code print when compiled and run:

Cavab : It will not compile.
In Java, a while or do/while construct takes an expression that returns a boolean. The expression --k is an integer, which is invalid and so the compilation fails.

You could change it to: `while( --k>0 ){ ... }`. In this case, --k>0 is a boolean expression and is valid.

```
public class InitTest{
   static String s1 = sM1("a"); {
      s1 = sM1("b");
   }
   static{
      s1 = sM1("c");
   }
   public static void main(String args[]){
      InitTest it = new InitTest();
   }
   private static String sM1(String s){
      System.out.println(s);  return s;
   }
}
```

Cavab : a,c,b

Evvelce staticler sonra instanceler sonra constructor sirasi ;)

```
public class LoadTest{
```

```
   public static void main(String[] args) throws Exception {
        LoadTest t = new LoadTest();
        int i = t.getLoad();
        double d = t.getLoad();
        System.out.println( i + d );
    }

    public int getLoad() {
       return 1;
    }

    public double getLoad(){
       return 3.0;
    }

}
```

Cavab : The code will not compile.
Eyni adda yalniz return tipine goreferlqenen iki metod ola bilmez !

Consider the following code:

```
class A{
   A() {  print();   }
   void print() { System.out.println("A"); }
}
class B extends A{
   int i =   4;
   public static void main(String[] args){
      A a = new B();
      a.print();
   }
   void print() { System.out.println(i); }
}
```

Cavab : It will print 0, 4

Note that method print() is overridden in class B. Due to polymorphism, the method to be executed is selected depending on the class of the actual object.
Here, when an object of class B is created, first B's default constructor (which is not visible in the code but is automatically provided by the compiler because B does not define any constructor explicitly) is called. The first line of this constructor is a call to super(), which invokes A's constructor. A's constructor in turn calls print(). Now, print is a non-private instance method and is therefore polymorphic, which means, the selection of the method to be executed depends on the class of actual object on which it is invoked. Here, since the class of actual object is B, B's print is selected instead of A's print. At this point of time, variable i has not been initialized (because we are still in the middle of initializing A), so its default value i.e. 0 is printed.

What is the output of the following snippet, assuming a and b are both 0?
3: try {
4: return a / b;
5: } catch (RuntimeException e) {
6: return -1;
7: } catch (ArithmeticException e) {
8: return 0;
9: } finally {
10: System.out.print("done");
11: }
A. -1
B. 0
C. done-1
D. done0
E. The code does not compile.
F. An uncaught exception is thrown.

Cavab :E.
The order of catch blocks is important because they’re checked in the order they appear after the try block. Because ArithmeticException is a child class of Runtime-Exception, the catch block on line 7 is unreachable. (If an ArithmeticException is thrown in try try block, it will be caught on line 5.) Line 7 generates a compiler error because it is unreachable code.

Consider the following code:

```
interface Flyer{ }
class Bird implements Flyer { }
class Eagle extends Bird { }
class Bat { }

public class TestClass {

   public static void main(String[] args) {
      Flyer f = new Eagle();
      Eagle e = new Eagle();
      Bat b = new Bat();

      if(f instanceof Flyer) System.out.println("f is a Flyer");
      if(e instanceof Bird) System.out.println("e is a Bird");
      if(b instanceof Bird) System.out.println("b is a Bird");
   }
}
```

What will be printed when the above code is compiled and run?

Cavab : It will not compile.
b points to an object of class Bat, which does not extend from Bird. Now, it is possible for b to point to an object of any subclass of Bat. However, it is not possible for that sub class to extend Bird (because a class can at most extend from only one class). Therefore, it is not possible for b to point to an object of a class that "is a" Bird. The compiler figures out this fact at compile time itself and so the code fails to compile.
Eger Bat extends Bird olsa compile error vermezdi !

Consider the following code:

```
interface Flyer{ }
class Bird implements Flyer { }
class Eagle extends Bird { }
class Bat { }

public class TestClass {
```

```
    public static void main(String[] args) {
        Flyer f = new Eagle();
        Eagle e = new Eagle();
        Bat b = new Bat();

        if(f instanceof Bird) System.out.println("f is a Bird");
        if(e instanceof Flyer) System.out.println("e is a Flyer");
        if(b instanceof Flyer) System.out.println("b is a Flyer");
    }
}
```

What will be printed when the above code is compiled and run?

Cavab : f is a Bird e is a Flyer

At run time, f points to an object of class Eagle. Now, Eagle extends Bird so f instanceof Bird returns true.
e points to an object of class Eagle. Eagle extends Bird, which in turn implements Flyer. So an Eagle is a Flyer. Therefore, e instanceof Flyer will also return true.

b points to an object of class Bat, which does not extend from Bird. Therefore, b instanceof Flyer returns false.
B instanceof Bird yazsaq umumiyyetle complie time error vererdi!

Which letters will be printed when the following program is run?

```
 public class TestClass{
  public static void main(String args[]){
  B b = new C();
   A a = b;
   if (a instanceof A) System.out.println("A");
   if (a instanceof B) System.out.println("B");
   if (a instanceof C) System.out.println("C");
   if (a instanceof D) System.out.println("D");
  }
 }
class A { }
class B extends A { }
class C extends B { }
class D extends C { }
```

Cavab : A ,B,C will be printed
The program will print A, B and C when run. The object denoted by reference a is of type C. The object is also an instance of A and B, since C is a subclass of B and B is a subclass of A. The object is not an instance of D.

An overriding method must have a same parameter list and the same return type as that of the overridden method.

Cavab :False
This would have been true prior to Java 1.5. But from Java 1.5, an overriding method is allowed to change the return type to any subclass of the original return type, also known as covariant return type. This does not apply to primitives, in which case, the return type of the overriding method must match exactly to the return type of the overridden method.

3 down vote

yes It is possible.. returns type can be different only if parent class method return type is

a super type of child class method return type..

means

```
class ParentClass { public Circle() method1() { return new Cirlce(); } }
class ChildClass extends ParentClass { public Square method1() { return
new Square(); } } Class Cirlce { } class Square extends Circle { } ---
```

If this is the then different return type can be allowed...

Which of these methods are not a part of the String class?

A trim
B length
C concat
D hashcode
E reverse

Cavab : reverse
The String class has no reverse( ) method but StringBuffer (and StringBuilder) do have this method.

What will be the result of attempting to compile and run the following class?

```
 public class TestClass{
  public static void main(String args[ ] ){
    int i, j, k;
    i = j = k = 9;
    System.out.println(i);
  }
}
```

Cavab :9

What will the following code print?

```
public class Test{
    public static void testInts(Integer obj, int var){
        obj = var++;
        obj++;
    }
    public static void main(String[] args) {
        Integer val1 = new Integer(5);
        int val2 = 9;
        testInts(val1++, ++val2);
        System.out.println(val1+" "+val2);
    }
}
```

Cavab : 6 10
Cunki testInts-de her hansi intance variable falan deyismir return de yoxdur nese.sadece gonderilen parameter deyisdirilir.
PSVM-de ise ++ istfde edildiyi ucun parametr gondermezden evvel deyerler 1 artirilir
Cavab 6 ve 10

```
interface Bozo{   int type = 0;    public void jump(); }

public class Type1Bozo implements Bozo{
    public Type1Bozo(){
        type = 1;
    }
    public void jump(){
        System.out.println("jumping..."+type);
    }
    public static void main(String[] args){
        Bozo b = new Type1Bozo();
        b.jump();
    }

}
```

Cavab : cannot assign a value to final variable type .
Unutmayinki Interface deyisenleri ve metodlari biz yazmasaq bele arx aterefde stativ ve finaldir. Final deyiskeen ise soradan deyerini deyismek olmaz !

What will the following code print when compiled and run?

```
class ABCD{
  int x = 10;
  static int y = 20;
}
class MNOP extends ABCD{
  int x = 30;
  static int y = 40;
}

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

```
    System.out.println(new MNOP().x+", "+new MNOP().y);
  }
}
```

Cavab : 30,40

Cunki javada variableler polymorphic deyiller, override edilmirler !

Consider the following method...

public int setVar(int a, int b, float c) { ...}

Which of the following methods correctly overload the above method?

```
public int setVar(int a, float b, int c){
  return (int)(a + b + c);
}
```

```
public int setVar(int a, float b, int c){
  return this(a, c, b);
}
```

this(...) can only be called in a constructor and that too as a first statement.

```
public int setVar(int x, int y, float z){
  return x+y;
}
```

It will not compile because it is same as the original method. The name of parameters do not matter.

```
public float setVar(int a, int b, float c){
  return c*a;
}
```

It will not compile as it is same as the original method. The return type does not matter.

```
public float setVar(int a){
  return a;
}
```

Consider the contents of following two files: //In file A.java

```
package a;
public class A{
        A(){ }
        public void  print(){ System.out.println("A"); }
}

//In file B.java
package b;
import a.*;
 public class B extends A{
  B(){ }
   public void  print(){ System.out.println("B"); }
   public static void main(String[] args){
    new B();
  }
 }
```

What will be printed when you try to compile and run class B?

Cavab : It will not compile.

Sebeb cunki dqiqet etseniz gorersinizki A() constructoru ve default access tipindedi. B sublcassinin obyekti yaranerken bilirikki arxa terefde super() ile ana class default constructoru cagrilir. Ancaq ayri paketlerde default access el catan deyil buna gore de A() constructoru cagira bilmir ve complie time error verir

What will be the result of compiling and running the following code?

```
class Base{     public Object getValue(){ return new Object(); } //1 }
class Base2 extends Base{     public String getValue(){ return "hello"; } //2 }
public class TestClass{
  public static void main(String[] args){
    Base b = new Base2();
     System.out.println(b.getValue()); //3
   }
 }
```

Cavab : It will print hello.

Covariant returns are allowed since Java 1.5, which means that an overriding method can change the return type to a subclass of the return type declared in the overridden method. But remember than covarient returns does not apply to primitives.

```
public class FinallyTest{
    public static void main(String args[]) throws Exception{
        try{
            m1();
            System.out.println("A");
        }
        finally{
            System.out.println("B");
        }
        System.out.println("C");
    }
    public static void m1() throws Exception { throw new Exception(); }

}
```

Cavab :It will print B and throw Exception.

Yeqinki duzunurduz compilte time error verecek catch olmadigi ucun. Ancaq biz diqqet etsek gorerikki throws edilib exception main metodumuzda.Eger etmeseydik unreported exception Exception; must be caught or declared to be thrown bele bir xeta alrdiq.!! Ancaq indi catch olmasa da throws etdiyimiz ucun normal isleyecek. A ekrana verilmeyecek cunki m1-de exceptionumuz throw edilir. Finally ise hemise ise dusur !

What is the result of executing the following fragment of code:

```
boolean b1 = false;
boolean b2 = false;
if (b2 != b1 = !b2){
  System.out.println("true");
} else{
  System.out.println("false");
}
```

Cavab : Compile time error.

Note that boolean operators have more precedence than =. (In fact, = has least precedence of all operators.)
so, in (b2 != b1 = !b2) first b2 != b1 is evaluated which returns a value 'false'. So the expression becomes false = !b2. And this is illegal because false is a value and not a variable!

Had it been something like (b2 = b1 != b2) then it is valid because it will boil down to : b2 = false.
Because all an if() needs is a boolean, now b1 != b2 returns false which is a boolean and as b2 = false is an expression and every expression has a return value (which is actually the Left Hand Side of the expression). Here, it returns false, which is again a boolean.

Note that return value of expression : i = 10 , where i is an int, is 10 (an int).

What will the following code print when compiled and run?

```
class StringWrapper {
  private String theVal;
   public StringWrapper(String str){
this.theVal = str;
}
}

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

StringWrapper sw = new StringWrapper("How are you?");
      StringBuilder sb = new StringBuilder("How are you?");
System.out.println("Hello, "+sw);
```

```
    System.out.println("Hello, "+sb);
  }
}
```

Cavab :
Hello, StringWrapper@<hashcode>
Hello, How are you?

1. When one of the operands of the + operator is a String and other is an object (other than String), toString method is called on the other operand and then both the Strings are concatenated to produce the result of the operation.
2. Object class contains an implementation of toString that returns the name of the class (including the package name) and the hash code of the object in the format <classname>@<hashcode>. For example, System.out.println("Hello, "+new Object()); will print Hello, java.lang.Object@3cd1a2f1, where 3cd1a2f1 is the hash code of the object.
3. StringBuilder class provides its own implementation of toString method, which returns the String value of its contents.

In this question, StringWrapper class does not implement toString method and so Object class's version is used

Consider the following class and interface definitions (in separate files):

```
 public class Sample implements IInt{
  public static void main(String[] args){
    Sample s = new Sample();  //1
     int j = s.thevalue;       //2
     int k = IInt.thevalue;    //3
     int l = thevalue;         //4
  }
 }

public interface IInt{
    int thevalue = 0;
}
```

What will happen when the above code is compiled and run?

Cavab : It will compile and run without any problem.

```
class A {
}

class AA extends A {
}


public class TestClass {
    public static void main(String[] args) throws Exception {
        A a = new A();
        AA aa = new AA();
        a = aa;
        System.out.println("a = "+a.getClass());
        System.out.println("aa = "+aa.getClass());
    }
}
```

What will the following code print when run?
Cavab :
a = class AA
aa = class AA

What will be the result of compiling and running the following code?

```
class Base{
   public short getValue(){ return 1; } //1
}
class Base2 extends Base{
   public byte getValue(){ return 2; } //2
}
public class TestClass{
   public static void main(String[] args){
      Base b = new Base2();
      System.out.println(b.getValue()); //3
   }
}
```

Cavab : Compile time error at //2
Eger eyni adda tanimlanibsa override demekdi ancaq burda return tipler eyni deyil. Ona goire de complie error verecek eger eyni olsa idi cavab 2 olardi. Getvalue adini getvalue1 etseydik ise error da almazdiq !

What will be the output of the following program?

```
 public class TestClass{
  public static void main(String args[ ] ){
    int i = 0 ;
    boolean bool1 = true ;
    boolean bool2 = false;
    boolean bool  = false;
    bool = ( bool2 &  method1(i++) ); //1
     bool = ( bool2 && method1(i++) ); //2
   bool = ( bool1 |  method1(i++) ); //3
    bool = ( bool1 || method1(i++) ); //4
    System.out.println(i);
  }
  public static boolean method1(int i){        return i>0 ? true : false;    } }
```

Cavab :2

Coxunuz niye 4 yox 2 deyecek ama diqeetle baxin bilirikki && veya || -da bir teref artiq umumi neticeni bildirise 2-ci terefa baxilmir
Burada 2 ve 4 setirlerde method1() hec cagrilmayacaq da !Ona giore de i deyeri 2 defe artmis olur.

```
package quiz;

public class Quiz24 {

  private static int increment(int i){
    int num= (++i) + (i++);
    return num;
  }

  public static void main(String[] args) {

    for(int i=0; i < 5; increment(i)){
      System.out.print(i);
    }
  }

}
```

Select correct answer from options below :

- 02
- 24
- 39
- Compilation fails
- Other result

Cavab : Other result (0)

Cunki incrimet i-de inin deyeir artmasi i=0 deyisenine tesir etmir soinsuz dongu yaranir

class MyException extends Throwable{}

```
class MyException1 extends MyException{}
class MyException2 extends MyException{}
class MyException3 extends MyException2{}

public class ExceptionTest{
   void myMethod() throws MyException{
      throw new MyException3();
   }
   public static void main(String[] args){
      ExceptionTest et = new ExceptionTest();
      try{
         et.myMethod();
      }
      catch(MyException me){
         System.out.println("MyException thrown");
      }
      catch(MyException3 me3){
         System.out.println("MyException3 thrown");
      }
      finally{
         System.out.println(" Done");
      }
   }
}
```

What is the result of compiling and running this code?
Cavab : It fails to compile

Cunki biz demisdikki catchde ustde daha subtipler asagi dusdukce boyuk tipler yazilmalidir. cunki yuxaridan asagi yoxlayir. MyException3 Myexcepton tipinden oldugu ucun artiq ilk catchde biz onu tuturuq.
Ona gore de eger biz catchlerin yerini deyissek MyException3 thrown ekrana verilerdi ancaq indiki halda compile olmayacaq cunki MyException3 unreachabledir.

Which of the given options should be inserted at line 1 so that the following code can compile without any errors?

```
package objective1;
// 1 public class StaticImports{
      public StaticImports(){
        out.println(MAX_VALUE);
      }
  }
```

Cavab :

```
import static java.lang.Integer.*;
import static java.lang.System.*;
```

Given:

```
//in file Movable.java
package p1;
public interface Movable {
  int location = 0;
  void move(int by);
  public void moveBack(int by);
}
```

```
//in file Donkey.java
package p2;
import p1.Movable;
public class Donkey implements Movable{
    int location = 200;
    public void move(int by) {
        location = location+by;
    }
    public void moveBack(int by) {
        location = location-by;
    }
}
```

```
//in file TestClass.java
package px;
import p1.Movable;
import p2.Donkey;
public class TestClass {
   public static void main(String[] args) {
      Movable m = new Donkey();
      m.move(10);
      m.moveBack(20);
      System.out.println(m.location);
   }
}
```
Identify the correct statement(s).

Cavab : It will print 0 when TestClass is run.

There is no problem with the code. All variables in an interface are implicitly public, static, and final. All methods in an interface are public. There is no need to define them so explicitly. Therefore, the location variable in Movable is public and static and the move() method is public.

Now, when you call m.move(10) and m.moveBack(20), the instance member location of Donkey is updated to 190 because **the reference m refers to a Donkey at run time and so move and moveBack methods of Donkey are invoked at runtime. However, when you print m.location, it is the Movable's location (which is never updated) that is printed.**

Consider the following classes...

```
class Teacher{
     void teach(String student){
        /* lots of code */
     }
}
class Prof extends Teacher{
      //1
}
```
Which of the following methods can be inserted at line //1 ?

Cavab :
public void teach() throws Exception
protected void teach(String s)
public final void teach(String s)
public abstract void teach(String s)

If super class method is public,while overriding it in child class,it should be public,Since it is the weakest access specifier or least restrictive access specifier
If super class method is protected,while overriding it in child class,it can be public or protected but not anything else
If superclass method is default,then while overriding it in child class,it can be public,protected or no access,but cannot be private.
If superclass method is private,then while overriding it in child class,it can be anything. Since private method cannot be overrided.

Which of the following are benefits of an array over an ArrayList ?
Cavab :
It consumes less memory.
Accessing an element in an array is faster than in ArrayList.

Given:

```
  public static boolean getBool(){
    return true;
  }
  public static String getString(){
    return "true";
  }
  public static void main(String args[]){
     switch( getBool() ){
          case true :
            System.out.println("true");
            break;
        default :
            System.out.println("none");
            break;
```

```
    }
  }
```

What changes can be done so that it will print only true?

Cavab : Call getString instead of getBool in the switch and also change the case label from true to "true".

Ola biler deyersinizki burda onsuzda true hemise ekrana verecek axi.Ancaq bele deyil.Cunki unutmayinki switch boolen qebul etmir. Biz onu stringe cevirsek isleyer

Which of the following are valid classes?

```
public class ImaginaryNumber extends Number {
//implementation for abstract methods of the base class
}
public class ThreeWayBoolean extends Boolean {
//implementation for abstract methods of the base class
}
public class NewSystem extends System {
//implementation for abstract methods of the base class
}
public class ReverseString extends String {
//implementation for abstract methods of the base class
}
```

Cavab :A
Cunki Boolen, Syetem, String finaldir ve onlardan toreme olmur. Number ise final deyil!
1. Remember that wrapper classes for primitives (java.lang.Boolean, java.lang.Integer, java.lang.Long, java.lang.Short etc.) are also final and so they cannot be extended.
2. java.lang.Number, however, is not final. Integer, Long, Double etc. extend Number.
3. java.lang.System is final as well.

class A

```
{

}

class B extends A
{

}

class C extends B
{

}

public class MainClass
{
   static void overloadedMethod(A a)
   {
      System.out.println("ONE");
   }

   static void overloadedMethod(B b)
   {
      System.out.println("TWO");
   }

   static void overloadedMethod(Object obj)
   {
      System.out.println("THREE");
   }

   public static void main(String[] args)
   {
      C c = new C();

      overloadedMethod(c);
   }
}
```

Cavab : TWO

2) In a class, one method has two overloaded forms. One form is defined as static and another form is defined as non-static. Is that method properly overloaded?

Cavab :

Yes. Compiler checks only method signature to verify whether a particular method is properly overloaded or not. It doesn't check static or non-static feature of the method.

3) In the below class, is 'method' overloaded or duplicated?
?

```
public class MainClass
{
    void method(int ... a)
    {
        System.out.println(1);
    }

    void method(int[] a)
    {
        System.out.println(2);
    }
}
```

View Answer

Answer :

Duplicated. Because, var args (int … a) are nothing but the arrays. So here, (int … a) and (int[] a) are the same.

5) In the below Class X, is 'method' properly overloaded?
?

```
1
2   class X
3
4   {
5
6       int method(int i, int d)
7
8       {
9
10          return i+d;
11
12      }
13
14
15      static int method(int i, double d)
16
17      {
18
19          return (int)(i+d);
20
21      }
22

        double method(double i, int d)

        {

            return i+d;

        }


        static double method(double i, double d)

        {

            return i+d;

        }

    }
```

View Answer

Answer :

Yes.

```
1
2   class X
3
4   {
5
6       void method(int a)
7
8       {
9
10          System.out.println("ONE");
11
12      }
13
14
15      void method(double d)
16
17      {
18
19          System.out.println("TWO");
20
21      }
22  }
23
24
25  class Y extends X
26
27  {
28
29      @Override
        void method(double d)
        {
            System.out.println("THREE");
        }
    }
```

```
public class MainClass
{
    public static void main(String[] args)
    {
        new Y().method(100);
    }
}
```

View Answer

Answer :

ONE

What will be the outcome of the below program?

?

```
1
2   public class MainClass
3
4   {
5
6       double overloadedMethod(double d)
7
8       {
9
10          return d *= d;
11
12      }
13
14
15      int overloadedMethod(int i)
16
17      {
18
19          return overloadedMethod(i *= i);
20
21      }
```

```
22
23      float overloadedMethod(float f)
24
        {
            return overloadedMethod(f *= f);
        }


        public static void main(String[] args)
        {
            MainClass main = new MainClass();


            System.out.println(main.overloadedMethod(100));
        }
    }
```

View Answer

Answer :

It will throw java.lang.StackOverflowError at run time. Because, overloadedMethod(int) keeps calling itself.

In a class, One method has 4 overloaded forms. All have different access modifiers (private, default, protected and public). Is that method properly overloaded?

View Answer

Answer :

Yes. That method is properly overloaded.

What will be the output of this program?

?

```
1
2   class SuperClass
3
4   {
5       void superClassMethod(Number n)
6
7       {
8
9           System.out.println("From Super Class");
10
11      }
12  }
13
14
15
16  class SubClass extends SuperClass
17
18  {
19
20      void superClassMethod(Double d)
21
22      {
23
24          System.out.println("From Sub Class");
25
        }

    }


    public class MainClass

    {

        public static void main(String[] args)

        {

            SubClass sub = new SubClass();


            sub.superClassMethod(123321);

        }
```

```
        }
```

View Answer

Answer :

From Super Class

```
class A
{
    public A(int i)
    {
        System.out.println(1);
    }

    public A()
    {
        this(10);

        System.out.println(2);
    }

    void A()
    {
        A(10);

        System.out.println(3);
```

```
    }

    void A(int i)
    {
        System.out.println(4);
    }
}


public class MainClass
{
    public static void main(String[] args)
    {
        new A().A();
    }
}
```

View Answer

Answer :

1

2

4

3

What will be the output of the following program?

?

```
1
2   class A
3
4   {
5       static void methodOne()
6
7       {
8
9           System.out.println("AAA");
10
11      }
12
13  }
14
15
16  class B extends A
17
18  {
19
20      static void methodOne()
21
22      {
23
24          System.out.println("BBB");
25
        }

    }


    public class MainClass

    {

        public static void main(String[] args)

        {

            A a = new B();


            a.methodOne();

        }
```

```
}
```

Cavab : Cunki staticdir metod. Overrifde edilmir cunki static metodlar instance-ye aid deyil!

Can you identify the error in below code snippet?

?

```
1  class A
2  {
3      void myMethod()
4      {
5          System.out.println("Super Class");
6      }
7  }
8
9  class B extends A
10 {
11     @Override
12     void myMethod() throws SQLException
13     {
           System.out.println("Sub Class");
       }
```

```
14      }
15
16
```

View Answer

Answer :

SQLException is not compatible with throws clause of super class method. If super class method doesn't have throws clause, then it can be overridden with only unchecked type of exceptions. SQLException is not an unchecked type of exception.

Can we remove throws clause of a method while overriding it?
View Answer

Answer :

Yes, we can remove throws clause of a method while overriding it.

How many objects have been created by the time the main method reaches its end in the following code?

```
public class Noobs {
   public Noobs(){
      try{
         throw new MyException();
      }catch(Exception e){
      }
   }
   public static void main(String[] args) {
      Noobs a = new Noobs();
      Noobs b = new Noobs();
      Noobs c = a;
   }
}
class MyException extends Exception{

}
```

Cavab :4

Cunki Noobs her yarananda thriw new MyException setri ile exception clasi da yaradilir.
c=a -da ise menimsetme var yeni elave obyekt yaranmir !
Ona gore de 2*2=4

Consider the following code:

```
class A{
   A() {  print();   }
   void print() { System.out.println("A"); }
}
class B extends A{
   int i =   4;
   public static void main(String[] args){
      A a = new B();
      a.print();
   }
   void print() { System.out.println(i); }
}
```

What will be the output when class B is run ?

Cavab : 0,4

Bu cox cetin sualdir. Burada bir bir gedek demeli ilk olaraq B yaradilarken super() cagirilir bilirikki. A-nin defult constructorunda ise print cagrilir bu override oldugu ucun B -ninkini cagirir ancaq hele bu zaman B class bloku run olmadigi ucun i deyeri 0-dir . Ekrana 0 verir daha sonra normal olaraq oz printini cagriir ve ekrana 4verir

Consider the following class...

```
public class ParamTest {

  public static void printSum(double a, double b){
     System.out.println("In double "+(a+b));
  }
  public static void printSum(float a, float b){
     System.out.println("In float "+(a+b));
```

```
  }

  public static void main(String[] args) {
     printSum(1.0, 2.0);
  }
}
```

What will be printed?

Cavab : In double 3.0

The call to printSum(1.0, 2.0) will be bound to printSum(double, double) because 1.0 and 2.0 are double, which are exact match to double, double.

Note that if you call printSum(1, 2) , printSum(float, float) would have been invoked instead of printSum(double, double) because a float is closer than a double to an int.

We advise you to run this program and try out various combinations. The exam has questions on this pattern.

Following is a supposedly robust method to parse an input for a float :

```
public float parseFloat(String s){
   float f = 0.0f;
   try{
      f = Float.valueOf(s).floatValue();
      return f ;
   }
   catch(NumberFormatException nfe){
      System.out.println("Invalid input " + s);
      f = Float.NaN ;
      return f;
   }
   finally { System.out.println("finally");  }
   return f ;
}
```

Which of the following statements about the above method is/are true?

Cavab : The code will not compile.

Note that the return statement after finally block is unreachable. Otherwise, if this line were not there, choices 1, 2, 3 are valid

What will the following program print?

```
class Test{
   public static void main(String[] args){
      int i = 4;
      int ia[][][] = new int[i][i = 3][i];
      System.out.println( ia.length + ", " + ia[0].length+", "+ ia[0][0].length);
   }
}
```

Cavab : 4 3 3
In an array creation expression, there may be one or more dimension expressions, each within brackets. Each dimension expression is fully evaluated before any part of any dimension expression to its right. The first dimension is calculated as 4 before the second dimension expression sets 'i' to 3.
Note that if evaluation of a dimension expression completes abruptly, no part of any dimension expression to its right will appear to have been evaluated.

What will the following program print when run?

```
public class Operators{

    public static int operators(){
        int x1 = -4;
        int x2 = x1--;
        int x3 = ++x2;
        if(x2 > x3){
            --x3;
        }else{
            x1++;
        }
        return x1 + x2 + x3;
    }
    public static void main(String[] args) {
        System.out.println(operators());
    }
}
```

Cavab : -10
Addim addim gedek
Demeli x2=x1-- ; burada evvelce x1 deyeri azalir olur artiq -5 ve x2 ise olur -4**.Cunki burada diqqet edin eger x2=--x1 olsa idi x2 de -5 olardi** . Daha sonra x2++ ile x2 yeni deyeri -3 olur ve burada ++x2 oldugu ucun x3 deyeri de -3 olur . Sert duzgun deyil elseye kecir ve x1 deyeri artir olur -4 . -4 -3 -3 cemi ise edir -10 !

What will the following code snippet print?

```
List s1 = new ArrayList( );
try{
   while(true){
      s1.add("sdfa");
   }
}catch(RuntimeException e){
   e.printStackTrace();
}
System.out.println(s1.size());
```

Cavab : It will throw an error at runtime that will not be caught by the catch block.
It will throw a java.lang.OutOfMemoryError. Note that this is not a subclass of RuntimeException or even Exception. It is a subclass of java.lang.Error.

Given:

```
import java.util.*;
public class TestClass {
   public static void main(String[] args) throws Exception {
      ArrayList<Double> al = new ArrayList<>();

      //INSERT CODE HERE
    }
}
```

What can be inserted in the above code so that it can compile without any error?

al.add(111);
You cannot box an int into a Double object.

System.out.println(al.indexOf(1.0));
indexOf's accepts Object as a parameter. Although 1.0 is not an object, it will be boxed into a Double object.

System.out.println(al.contains("string"));

Double d = al.get(al.length);
ArrayList does not have a field named length. It does have a method named size() though. So you can do:

Double d = al.get(al.size()); It will compile but will throw IndexOutOfBoundsException at run time in this case because al.size() will return 0 and al.get(0) will try to get the first element in the list.

Note that al is declared as ArrayList<Double>, therefore the add method is typed to accept only a Double.

What will be the output of the following program (excluding the quotes)?

```
public class SubstringTest{
   public static void main(String args[]){
      String String = "string isa string";
      System.out.println(String.substring(3, 6));
   }
}
```

Cavab : ing
Remember, indexing always starts from 0.
"hamburger".substring(4, 8) returns "urge"
"smiles".substring(1, 5) returns "mile"
Parameters:
beginIndex - the beginning index, inclusive.
endIndex - the ending index, exclusive.
Returns:
the specified substring.
Throws:
IndexOutOfBoundsException - if the beginIndex is negative, or endIndex is larger than the length of this String object, or beginIndex is larger than endIndex.

What will the following program print?

```
public class TestClass{
  public static void main(String[] args){
     Object obj1 = new Object();
     Object obj2 = obj1;
     if( obj1.equals(obj2) ) System.out.println("true");
     else  System.out.println("false");
  }
}
```

Cavab : true
Equals metodu deyende eslinde 2 sual sorusulmalidi obyektlerinmi muqayisesidi yoxsa stringinmi ? obyektlerin muqayisesinde eyni referansi gisterib gostemeediyini qaytarrir stringde ise deyerlerini muqayise edir

What can be added to the following Person class so that it is properly encapsulated and the code prints 29?

```
class Person{
   //Insert code here
}
public class Employee extends Person{
       public static void main(String[] args) {
          Employee e = new Employee();
          e.setAge(29);
          System.out.println(e.getAge());
       }
}
```

Cavab : A,D

```
private int age;
   public int getAge() {
      return age;
   }
   public void setAge(int age) {
      this.age = age;
   }
```

```
protected int age;
   public int getAge() {
      return age;
   }
   public void setAge(int age) {
      this.age = age;
   }
```

protected is not a valid way to encapsulate a field because any class in a package can access the field.

```
int age;
   public int getAge() {
      return age;
   }
```

```
    public void setAge(int age) {
        this.age = age;
    }
```

No access modifer to age means it has default access i.e. all the members of the package can access it. This breaks encapsulation.

```
private int age;
    private int getAge() {
        return age;
    }
    private void setAge(int age) {
        this.age = age;
    }
```

If you make getAge and setAge private, you cannot call them from Employee class.

```
private int age;
    public int getAge() {
        return age;
    }
    protected void setAge(int age) {
        this.age = age;
    }
```

What is the result of compiling and running the following code ?

```
public class TestClass{
  static int si = 10;
  public static void main (String args[]){
    new TestClass();
  }
  public TestClass(){
    System.out.println(this);
  }
  public String toString(){
    return "TestClass.si = "+this.si;
  }
}
```

Cavab : It will print TestClass.si = 10

It will print TestClass@nnnnnnnn, where nnnnnnnn is the hash code of the TestClass object referred to by 'this'.
It would have been correct if toString() were not overridden. This is the behavior of the toString() provided by Object class.

Following options show the complete code listings of a file. Which of these will compile?

```
//In file A.java
import java.io.*;
package x;
public class A{
}
```

The package statement, if exists, must be the first statement in a java code file. If you move it up before the import, this code will compile

```
//In file B.java
import java.io.*;
class A{
  public static void main() throws IOException{ }
}
```

There is nothing wrong with this code.
1. You can have a non-public class in a file with a different name.
2. You can have a main method that doesn't take String[] as an argument. It will not make the class executable from the command line though.

```
//In file A.java
public class A{
   int a;
   public void m1(){
     private int b = 0;
     a = b;
   }
}
```

Access modifiers (public/private/protected) are valid only inside the scope of a class, not of a method.

```
//In file A.java
public class A{
  public static void main(String[] args){
    System.out.println(new A().main);
  }
  int main;
```

```
}
```
There is nothing wrong with this code. You can have a method and a field with the same name in a class.

What will be result of attempting to compile this class?

```
import java.util.*;
package test;
public class TestClass{
   public OtherClass oc = new OtherClass();
}
class OtherClass{
   int value;
}
```

Cavab :
The class will fail to compile .
The package declaration can never occur after an import statement.

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

      boolean hasParams = (args == null ? false : true);
      if(hasParams){
         System.out.println("has params");
      }{
         System.out.println("no params");
      }
   }
}
```

Cavab :
has params
no params

Casdirici sualdir ama asandir.Cunki burada if-den sonraki scoplar elseye aid deyil!! Ve run olacaq sertden asili olmayaraq !

```
public class TestClass{
     static TestClass ref;
     String[] arguments;
     public static void main(String args[]){
         ref = new TestClass();
         ref.func(args);
    }
    public void func(String[] args){
        ref.arguments = args;
    }
}
```

Cavab :
The program will compile and run successfully.
Hetta arguments static de olsa idi ruin olacaqdi cunki ref.arguments deyerek biz cagirirqsa static olmayan metodda problem deyil!

Consider the following classes :

```
class A{
    public static void sM1() {  System.out.println("In base static"); }
}

class B extends A{

Line 1 --> // public static void sM1() {  System.out.println("In sub static"); }

Line 2 --> // public  void sM1() {  System.out.println("In sub non-static"); }

}
```

Which of the following statements are true?

Cavab :
class B will not compile if line 2 is uncommented.
static method cannot be overridden by a non-static method and vice versa

class B will not compile if line 1 and 2 are both uncommented.

What will the following code print when compiled and run?

```
class Base{
  void methodA(){
    System.out.println("base - MethodA");
  }
}

class Sub extends Base{
  public void methodA(){
    System.out.println("sub - MethodA");
  }
  public void methodB(){
    System.out.println("sub - MethodB");
  }
  public static void main(String args[]){
    Base b=new Sub(); //1
    b.methodA(); //2
    b.methodB(); //3
  }
}
```

Cavab :

Compile time error at //3

The point to understand here is, b is declared to be a reference of class Base and methodB() is not defined in Base. So the compiler cannot accept the statement b.methodB() because it only verifies the validity of a call by looking at the declared class of the reference.

For example, the compiler is able to verify that b.methodA() is a valid call because class Base has method methodA. But it does not "bind" the call. Call binding is done at runtime by the jvm and the jvm looks for the actual class of object referenced by the variable before invoking the method.

What will the following code print when compiled and run?

```
public class Discounter {
   static double percent; //1
```

```
    int offset = 10, base= 50; //2
    public static double calc(double value) {
        int coupon, offset, base; //3
        if(percent <10){ //4
            coupon = 15;
            offset = 20;
            base = 10;
        }
        return coupon*offset*base*value/100; //5
    }
    public static void main(String[] args) {
        System.out.println(calc(100));
    }
}
```

Cavab :
compilation error at //5

Remember that static and instance variables are automatically assigned a value even if you don't initialize them explicitly but local variables must be initialized explicitly before they are used.

Now, observe that the calc method declares local variables coupon, offset, and base but does not assign them a value. Even though at run time, we know that since percent is 0 and is thus < 10, a value will be assigned to these variables, the compiler doesn't know this because the compiler doesn't take values of "variables" into consideration while determining the flow of control. It only considers the values of compile time constants. Therefore, as far as the compiler is concerned, coupon, offset, and base may remain uninitialized before they are used.

Having uninitialized variables itself is not a problem. So there is no compilation error at //3. However, using them before they are initialized is a problem and therefore the compiler flags an error at //5.

Had percent been defined as final (static final double percent = 0;), the code would work and print 3000.0.

public class TestClass{
What will the following code print?

```
  int i = 1;
```

```
int j = i++;
if( (i==++j) | (i++ == j) ){
  i+=j;
}
System.out.println(i);
```

Cavab :5

If-den evvel i deyeri 2 , j deyeri 1 olur. Sert daxilinde de ilk ifade dogru olmadigi ucun her iki sert calisacaq . ilk sertde j deyeri , 2-ci sertde i deyeri 1 artir yeni i=3, j=2 olur Sert daxilinde ise i+=j yeni i=i+j 2+3 5-e beraber olur!

Given the following line of code:
List students = new ArrayList();
Identify the correct statement:

Cavab : The reference type is List and the object type is ArrayList.

Consider the following lines of code:

```
System.out.println(null + true); //1
System.out.println(true + null); //2
System.out.println(null + null); //3
```

Which of the following statements are correct?

Cavab : None of the 3 lines will compile.

Given the class

```
// Filename: Test.java
public class Test{
  public static void main(String args[]){
    for(int i = 0; i< args.length; i++){
      System.out.print("  "+args[i]);
```

```
    }
  }
}
```
Now consider the following 3 options for running the program:
a: java Test
b: java Test param1
c: java Test param1 param2
Which of the following statements are true?

Cavab :

It will print param1 param2 on option c.

It will not print anything on option a.

It will not throw NullPointerException because args[] is never null. If no argument is given (as in option a) then the length of args is 0.

Considering the following program, which of the options are true?

```
public class FinallyTest{
  public static void main(String args[]){
    try{
       if (args.length == 0) return;
       else throw new Exception("Some Exception");
    }
    catch(Exception e){
       System.out.println("Exception in Main");
    }
    finally{
       System.out.println("The end");
    }
  }
}
```

Cavab :
If run with no arguments, the program will only print 'The end'.
If run with one argument, the program will print 'Exception in Main' and 'The end'.

```
 class IncrementDemo{

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

```
    int a=20;
    a= a++ + ++a;

    System.out.println(a);

}
}
```

Cavab :42

a= 20 + 22
A++ da deyer artir ancaq heleki menimsetmede kohne deyer verilir
++a -da ise kohne deyer artmisdi ve yeni deyer artirilir hem de menimsedile bilir

What will the following code print when run?

```
public class TestClass {
    public static void main(String[] args) throws Exception {

        boolean flag  = true;
        switch (flag){
            case true : System.out.println("true");
                default: System.out.println("false");
        }

    }
}
```

Cavab :It will not compile.
A boolean cannot be used for a switch statement. It needs an integral type, an enum, or a String.
Here are the rules for a switch statement:
1. Only String, byte, char, short, int, (and their wrapper classes Byte, Character, Short, and Integer), and enums can be used as types of a switch variable. (String is allowed only since Java 7).
2. The case constants must be assignable to the switch variable. For example, if your switch variable is of class String, your case labels must use Strings as well.
3. The switch variable must be big enough to hold all the case constants. For example, if the switch variable is of type char, then none of the case constants can be greater than 65535 because a char's range is from 0 to 65535.
4. All case labels should be COMPILE TIME CONSTANTS.

5. No two of the case constant expressions associated with a switch statement may have the same value.
6. At most one default label may be associated with the same switch statement.

What will the following code print ?

```
class Test{
   public static void main(String[] args){
      int k = 1;
      int[] a = { 1 };
      k += (k = 4) * (k + 2);
      a[0] += (a[0] = 4) * (a[0] + 2);
      System.out.println( k + " , " + a[0]);
   }
}
```

Cavab : 25 , 25

The value 1 of k is saved by the compound assignment operator += before its right-hand operand (k = 4) * (k + 2) is evaluated. Evaluation of this right-hand operand then assigns 4 to k, calculates the value 6 for k + 2, and then multiplies 4 by 6 to get 24. This is added to the saved value 1 to get 25, which is then stored into k by the += operator. An identical analysis applies to the case that uses a[0].

```
k += (k = 4) * (k + 2);
a[0] += (a[0] = 4) * (a[0] + 2);
k = k + (k = 4) * (k + 2);
a[0] = a[0] + (a[0] = 4) * (a[0] + 2);
```

What will the following code print?

```
public class BreakTest{
  public static void main(String[] args){
    int i = 0, j = 5;
    lab1 : for( ; ; i++){
      for( ; ; --j)  if( i >j ) break lab1;
    }
    System.out.println(" i = "+i+", j = "+j);
  }
}
```

Cavab :
i = 0, j = -1
The values of i and j in the inner most for loop change as follows:
i = 0 j = 5
i = 0 j = 4
i = 0 j = 3
i = 0 j = 2
i = 0 j = 1
i = 0 j = 0
i = 0 j = -1
Therefore, the final println prints i = 0, j = -1

Given:

```
class Triangle{
    public int base;
    public int height;
    private final double ANGLE;

    public  void setAngle(double a){  ANGLE = a;  }

    public static void main(String[] args) {
        Triangle t = new Triangle();
        t.setAngle(90);
    }
}
```

Cavab : The code will not compile.
java:6: error: cannot assign a value to final variable ANGLE

What will the following class print when run?

```
public class Sample{
   public static void main(String[] args)  {
     String s1 = new String("java");
     StringBuilder s2 = new StringBuilder("java");
     replaceString(s1);
     replaceStringBuilder(s2);
     System.out.println(s1 + s2);
  }
```

```
  static void replaceString(String s) {
     s = s.replace('j', 'l');
  }
  static void replaceStringBuilder(StringBuilder s) {
     s.append("c");
  }
}
```

Cavab : javajavac

String is immutable while StringBuilder is not. So no matter what you do in replaceString() method, the original String that was passed to it will not change. On the other hand, StringBuilder methods, such as replace or append, change the StringBuilder itself. So, 'c' is appended to java in replaceStringBuilder() method.

What can be inserted in the code below so that it will print UP UP UP?

```
public class Speak {
    public static void main(String[] args) {
        Speak s = new GoodSpeak();

        INSERT CODE HERE

    }
}
class GoodSpeak extends Speak implements Tone{
    public void up(){
        System.out.println("UP UP UP");
    }
}
interface Tone{
    void up();
}
```

Cavab :

```
((Tone)s).up();
((GoodSpeak)s).up();
```

Consider the following class :

```
class Test{
  public static void main(String[] args){
    for (int i = 0; i < 10; i++) System.out.print(i + " ");  //1
    for (int i = 10; i > 0; i--) System.out.print(i + " ");  //2
    int i = 20;                  //3
    System.out.print(i + " ");   //4
  }
}
```

Which of the following statements are true?

Cavab :

As such, the class will compile and print "20 " (without quotes) at the end of its output.
It will not compile if line 3 is removed.
It will not compile if line 3 is removed and placed before line 1.
It will not compile if line 4 is removed and placed before line 3.
Only Option 2, 3, and 4 are correct. (sehv)

Diqqet edinki for-larda istfde olunan i ancaq oz scopunda tainir kenarda taninmir.Ona gore tekrar istfde olar i adi ile.
The scope of a local variable declared in 'for' statement is the rest of the 'for' statement, including its own initializer. So, when line 3 is placed before line 1, there is a redeclaration of i in the first for() which is not legal.
As such, the scope of i's declared in for() is just within the 'for' blocks. So placing line 4 before line 3 will not work since 'i' is not in scope there.

Which of the following lines of code that, when inserted at line 1, will make the overriding method in SubClass invoke the overridden method in BaseClass on the current object with the same parameter.

```
class BaseClass{
  public void print(String s) {  System.out.println("BaseClass :"+s); }
}
class SubClass extends BaseClass{
  public void print(String s){
    System.out.println("SubClass :"+s);
    // Line 1
  }
  public static void main(String args[]){
    SubClass sc = new SubClass();
    sc.print("location");
  }
}
```

Cavab : super.print(s);

This is the right syntax to call the base class's overridden method. However, note that there is no way to call a method if it has been overriden more than once. For example, if you make BaseClass extend from another base class SubBase, and if SubBase also has the same method, then there is no way to invoke SubBase's print method from SubClass's print method. You cannot have something like super.super.print(s);

What will the following program print when run?

```
// Filename: TestClass.java
public class TestClass{
   public static void main(String args[] ){ A b = new B("good bye");  }
}
class A{
   A() { this("hello", " world");  }
   A(String s) { System.out.println(s); }
   A(String s1, String s2){ this(s1 + s2); }
}
class B extends A{
   B(){ super("good bye"); };
   B(String s){ super(s, " world"); }
   B(String s1, String s2){ this(s1 + s2 + " ! "); }
}
```

Cavab : It will print "good bye world".

What will the following code print when compiled and run?

```
public class DaysTest{

    static String[] days = {"monday", "tuesday", "wednesday", "thursday", "friday",
"saturday", "sunday" };

    public static void main(String[] args) {

        int index = 0;
        for(String day : days){
```

```
            if(index == 3){
               break;
            }else {
                continue;
            }
            index++;
            if(days[index].length()>3){
               days[index] = day.substring(0,3);
            }
        }
        System.out.println(days[index]);
    }
}
```

Cavab : It will not compile.
error: unreachable statement
index++

Given the following declaration:

```
int[][] twoD = { { 1, 2, 3} , { 4, 5, 6, 7}, { 8, 9, 10 } };
```

What will the following lines of code print?

```
System.out.println(twoD[1].length);
System.out.println(twoD[2].getClass().isArray());
System.out.println(twoD[1][2]);
```

Cavab :
4
true
6

```
public class DaysTest{

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

```
        int i1 = 1, i2 = 2, i3 = 3;
        int i4 = i1 + (i2=i3 );
        System.out.println(i4);
    }
}
```

Cavab :4

What will the following program print?

```
class Test{
    public static void main(String args[]){
        int c = 0;
        boolean flag = true;
        for(int i = 0; i < 3; i++){
            while(flag){
                c++;
                if(i>c || c>5) flag = false;
            }
        }
        System.out.println(c);
    }
}
```

Cavab :6

While ise dusur 6 olduqda flag false olur. Normalda 3 defe for use dusur ancaq while 1 defe cunki ikincid efe artiq flag false oldugu ucun ise dusmur ce c deyeri 6 qalir!

Consider the following class definition:

```
public class TestClass{
   public static void main(String[] args){  new TestClass().sayHello(); }   //1
   public static void sayHello(){ System.out.println("Static Hello World"); }  //2
   public void sayHello() { System.out.println("Hello World "); }  //3
}
```

What will be the result of compiling and running the class?

Cavab :Compilation error at line 3.

You cannot have two methods with the same signature (name and parameter types) in one class.
Also, even if you put one sayHello() method in other class which is a subclass of this class, it won't compile because you cannot override/hide a static method with a non static method and vice versa.

Consider the following class...

```
class Test{
  public static void main(String[ ] args){
    int[] a = { 1, 2, 3, 4 };
    int[] b = { 2, 3, 1, 0 };
    System.out.println( a [ (a = b)[3] ] );
  }
}
```

What will it print when compiled and run ?

Cavab : 1

In an array access, the expression to the left of the brackets appears to be fully evaluated before any part of the expression within the brackets is evaluated.
In the expression a[(a=b)[3]], the expression a is fully evaluated before the expression (a=b)[3]; this means that the original value of a is fetched and remembered while the expression (a=b)[3] is evaluated. This array referenced by the original value of a is then subscripted by a value that is element 3 of another array (possibly the same array) that was referenced by b and is now also referenced by a. So, it is actually a[0] = 1.
Note that if evaluation of the expression to the left of the brackets completes abruptly, no part of the expression within the brackets will appear to have been evaluated.

Assume the following declarations:

```
class A{ }
class B extends A{ }
class C extends B{ }
```

```
class X{
   B getB(){ return new B(); }
}

class Y extends X{
  //method declaration here
}
```

Which of the following methods can be inserted in class Y?

Cavab :
public C getB(){ return new B(); }
Its return type is specified as C, but it is actually returning an object of type B. Since B is NOT a C, this will not compile.

protected B getB(){ return new C(); }
Since C is-a B, this is valid. Also, an overriding method can be made less restrictive. protected is less restrictive than 'default' access.

C getB(){ return new C(); }
Covariant returns are allowed in Java 1.5, which means that an overriding method can change the return type of the overridden method to a subclass of the original return type. Here, C is a subclass of B. So this is valid.

A getB(){ return new A(); }
An overriding method cannot return a superclass object of the return type defined in the overridden method. A subclass is allowed in Java 1.5.

What will be printed when the following code is compiled and run?

```
public class LoadTest{

    public static void main(String[] args) throws Exception {
        LoadTest t = new LoadTest();
        int i = t.getLoad();
        double d = t.getLoad();
        System.out.println( i + d );
    }

    public int getLoad() {
       return 1;
    }
```

```
    public double getLoad(){
       return 3.0;
    }

}
```

Cavab : The code will not compile.

You cannot have more than one method in a class with the same signature. Method signature includes method name and the argument list but does not include return type.
Therefore, the two getLoad() methods have the same signature and will not compile.

This shows that method overloading cannot be done on the basis of the return types.
Adlari ferqli olsaydi cavab 4.0 olardi .

What will be the output of the following program?

```
class TestClass{
   public static void main(String[] args) throws Exception{
      try{
         amethod();
         System.out.println("try ");
      }
      catch(Exception e){
         System.out.print("catch ");
      }
      finally   {
         System.out.print("finally ");
      }
      System.out.print("out ");
   }
   public static void amethod(){ }
}
```

Cavab :try finally out

Catch ekrana verilmir cunki main metod throw elese de metodumuz throw elemir hecne!

What is the output of the following program?
1: public class Cat {

```
2: public String name;
3: public void parseName() {
4: System.out.print("1");
5: try {
6: System.out.print("2");
7: int x = Integer.parseInt(name);
8: System.out.print("3");
```
Review Questions 329
```
9: } catch (NullPointerException e) {
10: System.out.print("4");
11: }
12: System.out.print("5");
13: }
14: public static void main(String[] args) {
15: Cat leo = new Cat();
16: leo.name = "Leo";
17: leo.parseName();
18: System.out.print("6");
19: }
20: }
```

Cavab : 1,2

Ola biler siz dunurduzki cavab 1,2,4 olacaq ancaq bele deyil!!!. Cunki diqqet edin 7-ci setirde bas veren exception NullPointerException deyil!! Catch icerisine hec girilmeyecek !

What is the output of the following snippet, assuming a and b are both 0?

```
3: try {
4: return a / b;
5: } catch (RuntimeException e) {
6: return -1;
7: } catch (ArithmeticException e) {
8: return 0;
9: } finally {
10: System.out.print("done");
11: }
```

A. -1
B. 0
C. done-1
D. done0
E. The code does not compile.
F. An uncaught exception is thrown.

Cavab : E

Cunki demisdik spesifik tipler daha ustde olmalidir !
Burada Arithmatic exception runtimeexceptiondan ustde yazilmali idi !

Which of the following can be inserted on line 8 to make this code compile? (Choose all that apply)
7: public void ohNo() throws IOException {
8: // INSERT CODE HERE
9: }
A. System.out.println("it's ok");
B. throw new Exception();
C. throw new IllegalArgumentException();
D. throw new java.io.IOException();
E. throw new RuntimeException();

Cavab : B-den basqa hamisi

B ola bilmez cunki Exception IOExceptiondan daha boyukdur.
A yaza bilerik,
C ve E RuntimeExxceptiondur.Ve onlar istenilen metodda throw edile bilerler !
D metodda throw edilerek declare olundugu ucun icaze var ;)

Which of the following are unchecked exceptions? (Choose all that apply)
A. ArrayIndexOutOfBoundsException
B. IllegalArgumentException
C. IOException
D. NumberFormatException
E. Any exception that extends RuntimeException
F. Any exception that extends Exception

Cavab : A ,B , D , E

Which of the following are true? (Choose all that apply)
A. Checked exceptions are allowed to be handled or declared.
B. Checked exceptions are required to be handled or declared.
C. Errors are allowed to be handled or declared.
D. Errors are required to be handled or declared.
E. Runtime exceptions are allowed to be handled or declared.

F. Runtime exceptions are required to be handled or declared.

Cavab : A , B , C , E

Consider the following class...

```
class MyString extends String{
  MyString(){ super(); }
}
```

The above code will not compile.

Cavab : true

This will not compile because String is a final class and final classes cannot be extended.
There are questions on this aspect in the exam and so you should remember that StringBuffer and StringBuilder are also final. All Primitive wrappers are also final (i.e. Boolean, Integer, Byte etc).
java.lang.System is also final.

```
class Triangle{
   public int base;
   public int height;
   public double area = 0;

   public Triangle(int pBase, int pHeight){
      this.base = pBase; this.height = pHeight;
      updateArea();
   }
   public void updateArea(){
      double a = base*height/2;
      area = a;
   }
   public void setBase(int b){ base  = b; updateArea(); }
   public void setHeight(int h){ height  = h; updateArea(); }
}
```

Which variables are not accessible from anywhere within given class code except from the scope in which they are declared?

CAVAB : b, h, a

b and h are method parameters and are only accessible in the method setBase and setHeight respectively.
a is a local variable and is accessible only in updateArea method.

base, height, and area are instance level and can be accessed from anywhere within the class where "this" is accessible.

Given that OurClass is a MyClass and OurClass has a YourClass object. Which of the following options are correct?

CAVAB :
OurClass contains a reference to MyClass
OurClass contains a reference to YourClass

Consider the following variable declaration within the definition of an interface: int i = 10; Which of the following declarations defined in a non-abstract class, is equivalent to the above?

CAVAB : public static final int i = 10;
Fields in an interface are implicitly public, static and final. Although you can put these words in the interface definition but it is not a good practice to do so.

Which of these statements concerning interfaces are true?

An interface may extend an interface.
Unlike a class, an interface can extend from multiple interfaces.

An interface may extend a class and may implement an interface.
An interface cannot implement another interface. It can extend another interface but not a class.

A class can implement an interface and extend a class.

A class can extend an interface and can implement a class.

An interface can only be implemented and cannot be extended.
It can be extended by another interface.

Which of the following statements are true?

Private methods cannot be overridden in subclasses.
Only methods that are inherited can be overridden and private methods are not inherited.

A subclass can override any method in a non-final superclass.
Only the methods that are not declared to be final can be overridden. Further, private methods are not inherited so they cannot be overridden either.

An overriding method can declare that it throws a wider spectrum of checked exceptions than the method it is overriding

The parameter list of an overriding method must be a subset of the parameter list of the method that it is overriding.
An overriding method (the method that is trying to override the base class's method) must have the same parameters.

The overriding method may opt not to declare any throws clause even if the original method has a throws clause.
No exception (i.e. an empty set of exceptions) is a valid subset of the set of exceptions thrown by the original method so an overriding method can choose to not have any throws clause.