1. ¿Qué arroja?

```
public class Main {

public static void main(String[] args) {

                String[] at = {"FINN", "JAKE"};
                for (int x=1; x<4;
                x++){ for (String s :
                at){
                System.out.println(x + " "+
                s); if(x==1){
                break;
                                }

                        }

                }

        }

}
```

Resultado: 1 FINN 2 FINN 2 JAKE 3 FINN 3 JAKE

2. ¿Qué 5 líneas son correctas?

```
class Light{

        protected int lightsaber(int x){return
0;}} class Saber extends Light{

         private int lightsaber (int x){return 0;}
```

//Error. El modificador de acceso en la clase derivada no puede ser más restrictivo que el modificador de acceso en la clase base

```
protected int lightsaber (long x){return 0;}

private int lightsaber (long x){return 0;}
protected long lightsaber (int x){return
0;}
protected long lightsaber (int x, int y){return 0;}

public int lightsaber (int x){return 0;}

protected long lightsaber (long x){return 0;}
```

3. ¿Qué resultado arroja?

```
class Mouse{

        public int numTeeth;
        public int
        numWhiskers; public
        int weight;
```

```
        public Mouse (int weight){

                this(weight.16);

        }

                public Mouse (int weight, int
                        numTeeth){
                        this(weight, numTeeth,
                        6);
        }
                public Mouse (int weight, int numTeeth, int
                        numWhiskers){ this.weight = weight;
                this.numTeeth= numTeeth;
                this.numWhiskers =
                numWhiskers;
        }

        public void print (){

                System.out.println(weight + ""+ numTeeth+ ""+ numWhiskers);

        }
                public static void main (String
                        [] args){ Mouse mouse
                        = new Mouse (15);
                        mouse.print();
        }
    }
```

Resultado: 15 , 16 , 6

4. ¿Cuál es la salida?

```
class Arachnid {

      public String type =
      "a"; public
      Arachnid(){
              System.out.println("arachnid");

              }

      }

      class Spider extends
          Arachnid{ public
          Spider(){
          System.out.println("spider");

          }

 void
 run(){
```

```
 type =
 "s";
      System.out.println(this.type + " " + super.type);

          }

      public static void main(String[]
          args) { new Spider().run();
          }

      }
```

Resultado: arachnid spider s s

5. Resultado

```
class Test {

          public static void main(String[]
          args) { int b = 4;
          b--;
          System.out.println(--
          b);
          System.out.println(b)
          ;
          }

      }
```

Resultado: 2 2

6. Resultado

```
class Sheep {

          public static void main(String[]
          args) { int ov = 999;
          ov--;
          System.out.println(--o
          v);
          System.out.println(ov)
          ;
          }

}
```

Resultado: 997, 997

7. Resultado

```
class Overloading
                    {
```

```
        public static void main(String[]
        args) {
        System.out.println(overload("a"));
        System.out.println(overload("a",
        "b"));
        System.out.println(overload("a", "b", "c"));

    }

        public static String
                overload(String s){
                return "1";
    }
        public static String
                overload(String... s){
                return "2";
    }
        public static String
                overload(Object o){
                return "3";
    }
        public static String
                overload(String s, String
                t){ return "4";

    }

}
```

Resultado: 1, 4, 2

8. Resultado

```
    class Base1 extends
                    Base{
        public void test(){

                System.out.println("Base1");

        }

}

    class Base2 extends
                    Base{
        public void test(){

                System.out.println("Base2");

        }
}

class Test {
```

```
        public static void
            main(String[] args)
            { Base obj = new
            Base1(); ((Base2)
            obj).test();
    }

}
```

Resultado: ClassCastException

9. Resultado

```
public class Fish {

        public static void
            main(String[] args)
            { int numFish = 4;
        String fishType= "Tuna";
        String anotherFish = numFish +1;
        System.out.println(anotherFish + " " +
        fishType);
    System.out.println(numFish + " " + 1);

    }

}
```

Resultado: El código no compila

10. Resultado

```
class MathFun {

        public static void
            main(String[] args)
            { int number1 =
            0b0111;
        int number2 = 0111_000;
        System.out.println("Number1:
        "+number1);
        System.out.println("Number2:
        "+number1);
    }

}
```

Resultado: 7 7

11. Resultado

```
class Calculator {
```

```
        int num =100;

        public void calc(int num){

                this.num =num*10;

        }
public void printNum(){

        System.out.println(num);

        }

        public static void main (String
                [] args){ Calculator obj
                = new Calculator ();
                obj.calc(2);
        obj.printNum();

        }

}
```

Resultado: 20

12. ¿Qué aseveraciones son correctas?

```
import java.lang

class ImportExample {

                public static void main (String []
                        args){ Random r = new
                        Random();
                        System.out.println(r.nex
                        tInt(10));

        }

}
```

* If you omit java.util import statements java compiles gives you an error.
* java.lang and util.random are redundant.
* you don't need to import java.lang.

13. Resultado

```
public class Main {

                public static void
                        main(String[] args)
                        { int var = 10;
                        System.out.println(
                        var++);
```

```
                System.out.println(
                ++var);
        }

}
```

Resultado: 10, 12

14. Resultado

```
class MyTime {

        public static void main
                (String [] args){
                short mn =11;
        short hr;
        short sg =
        0;
                for (hr = mn; hr >
                        6; hr -= 1){
                        sg++;
        }
    System.out.println("sg= " + sg);
    }

}
```

Resultado: sg = 5

15. ¿Cuáles son verdad?

- o An ArrayList is mutable.
- o An Array has a fixed size.
- o An array is mutable.
- o An array allows multiple dimensions.
- o An arrayList is ordered.
- o An array is ordered..

16. Resultado

```
public class MultiverseLoop {

        public static void main
                (String [] args){ int
                negotiate = 9;
        do{
                System.out.println(negotiate);

        }while (--negotiate);

    }

}
```

Errores de compilación, necesita un boolean en while

17. Resultado

```
class App {

		public static void main(String[] args) {
			Stream<Integer> nums =
			Stream.of(1,2,3,4,5); nums.filter(n
			-> n % 2 == 1);
		nums.forEach(p -> System.out.println(p));

	}

}
```

Exception at runtime, se debe encadernar el stream por que se consume

18. Suppose the declared type of x is a class, and the declared type of y is an interface. When is the assignment x = y; legal?

When the type of X is Object

19. When a byte is added to a char, what is the type of the result?

int

20. The standard application programming interface for accessing databases in java?

JDBC

21. Which one of the following statements is true about using packages to organize your code in Java ?

Packages allow you to limit access to classes, methods, or data from classes outside the package.

22. Forma correcta de inicializar un booleano

boolean a = (3>6);

23. Resultado

```
class Y{
```

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

                doSomething();
                }catch (RuntimeException
                exception){
                System.out.println(exception);
                }
        }
                static void doSomething() throws
                        IOException { if
                        (Math.random() > 0.5){
                }
                throw new RuntimeException();

        }

}
```

Resultado: RunTimeException

24. Resultado

```
interface Interviewer {

        abstract int interviewConducted();

        }

        public class Manager implements
                Interviewer{ int
                interviewConducted() {
                return 0;

        }

}
```

Resultado: Won't compile

25. Pregunta

```
class Arthropod {

                public void
                        printName(double
                        Input){
                        System.out.println(
                        "Arth");
        }
}

class Spider extends Arthropod {
```

```
                public void printName(int
                        input) {
                        System.out.println(
                        "Spider");
        }
        public static void
                main(String[] args)
                { Spider spider =
                new Spider();
                spider.printName(4
                );
                spider.printName(9
                .0);
        }
}
```

Resultado: Spider, Arth

26. Pregunta

```
public class Main {

        public enum Days{Mon,Tue,
        Wed} public static void
        main(String[] args) {
                for (Days d:Days.values()) {
                        Days[] d2 =
                        Days.values();
                        System.out.println(d2[2
                        ]);
                }

        }

}
```

Resultado:

Wed

Wed

Wed

27. Pregunta

```
public class Main{

        public enum Days {MON, TUE,
        WED}; public static void
        main(String[] args) {
                boolean x= true, z = true;
                int y = 20;
```

```
                x = (y!=10)^(z=false);
                System.out.println(x + " " + y + "
                "+ z);
        }

}
```

Resultado: true 20 false

28. Pregunta

```
class InitializacionOrder {

        static {add(2);}

        static void add(int num){

                System.out.println(num+"");

        }

        InitializacionOrder(){add(5)
        ;} static {add(4);}
        {add(6);}

        static {new InitializacionOrder();}

        {add(8);}

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

}
```

Resultado: 2 4 6 8 5

29. Pregunta

```
public class Main {

                public static void
                        main(String[] args) {
                        String message1 =
                        "Wham bam";
                String message2 = new String("Wham
                bam"); if (message1!=message2){
                        System.out.println("They dont match");
                        }else {
                System.out.println("They
                match");
                }
        }

}
```

Resultado: They don't match

30. Pregunta

```
class Mouse{

        public String
        name; public
        void run(){
                System.out.println("1
                "); try{
                System.out.println("2
                "); name.toString();

                System.out.println("3");
                }catch(NullPointerEx
                        ception      e){
                        System.out.pr
                        intln("4");
                        throw e;
        }

        System.out.println("5");

        }
        public static void
                main(String[] args)
                { Mouse jerry =
                new Mouse();
                jerry.run();
                System.out.println(
                "6");
        }
}
```

Resultado: 1 2 4 NullPointerException

31. Pregunta

```
public class Main {

        public static void main(String[] args) {

                try (Connection con = DriverManager.getConnection(url,
                uname, pwd)){
                Statement stmt =con.createStatement();
                System.out.print(stmt.exeuteUpdate("INSERT
                INTO User VALUES (500, 'Ramesh')"));
                }
        }

}
```

Resultado: arroja 1

32. Pregunta

```
class MarvelClass{

        public static void main
            (String [] args){
            MarvelClass ab1,
            ab2, ab3; ab1
            =new
            MarvelClass();
        ab2 = new
        MarvelMovieA(); ab3 =
        new MarvelMovieB();
        System.out.println ("the profits are " + ab1.getHash()+
        "," + ab2.getHash()+","+ab3.getHash());
        }

    public int getHash(){

        return 676000;

    }

}


    class MarvelMovieA extends
        MarvelClass{ public int
        getHash (){
        return 18330000;

    }

}

    class MarvelMovieB extends
        MarvelClass { public int
        getHash(){
        return 27980000;
    }

}
```

Resultado: the profits are 676000, 18330000, 27980000

33. Pregunta

```
class Song{

    public static void main (String [] args){

        String[] arr = {"DUHAST","FEEL","YELLOW","FIX YOU"};
```

```
                for (int i =0; i <= arr.length;
                i++){
                System.out.println(arr[i]);
                }
        }
}
```

Resultado: 4 An arrayindexoutofbondsexception

34. Pregunta

```
class Menu {

        public static void main(String[] args) {
                String[] breakfast = {"beans", "egg", "ham",
                "juice"}; for (String rs : breakfast) {
                        int dish = 2;
                                while (dish < breakfast.length)
                                        { System.out.println(rs
                                        + "," + dish); dish++;
                        }

                }

        }

}
```

Resultado: beans,2, beans,3, egg,2, egg,3, ham,2, ham,3, juice,2, juice,3

35. Which of the following statement are true:

* string builder es generalmente más rápido qué string buffer.
* string buffer is threadsafe; stringbuilder is not.

36. Pregunta

```
class CustomKeys{

        Integer key;
        CustomKeys(Integer
        k){
                key = k;

        }

        public boolean equals(Object o){

                return ((CustomKeys)o).key==this.key;

        }

}
```

Resultado: compilation fail

37.The catch clause is of the type:

Throwable.

Exception but NOT including RuntimeException.

CheckedException.

RunTimeException.

38. An enhanced for loop

also called for each, offers simple syntax to iterate through a collection but it can´t be used to delete elements of a collection

39. Which of the following methods may appear in class Y, which extends x ?

public void doSomething(int a, int b){…}

40. Pregunta

```
public class Main {

                public static void
                        main(String[] args)
                        { String s1= "Java";
                String s2 = "java";

                if (s1.equalsIgnoreCase(s2)){


                        System.out.println ("Equal");

                        } else {

                        System.out.println ("Not equal");

                }

        }

}
```

Resultado: Equal; respuesta: s1.equalsIgnoreCase(s2)

41. Pregunta

```
class App {
```

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

                String[] fruits = {"banana", "apple", "pears", "grapes"};

                // Ordenar el arreglo de frutas utilizando
        compareTo Arrays.sort(fruits, (a, b) ->
        a.compareTo(b));
        // Imprimir el arreglo de frutas
        ordenado for (String s : fruits) {
                System.out.println(""+s);
                }

        }

}
```

Resultado: apple, banana, grapes, pears

42. Pregunta

```
public class Main {

                public static void main(String[] args)
                        { int[]countsofMoose = new
                        int [3];
                        System.out.println(countsof
                        Moose[-1]);
        }

}
```

Resultado: this code will trow an ArrayIndexOutOfBoundsExpression

43. Pregunta

```
class Salmon{

        int count;

        public void Salmon (){

                count =4;

        }

                public static void
                        main(String[] args)
                        { Salmon s = new
                        Salmon();
                        System.out.println(
                        s.count);
        }

}
```

Resultado: 0

44. Pregunta

```
class Circuit {

                public static void
                        main(String[] args)
                        { runlap();
                int
                c1=c2;
                int c2 =
                v;
        }

        static void runlap(){

                System.out.println(v);

        }

        static int v;

}
```

Resultado: Hay que corregir linea 6; c1 se le asigna c2 pero c2 aún no se declara

45. Pregunta

```
class Foo {

                public static void
                        main(String[] args)
                        { int a=10;
                long
                b=20;
                short
                c=30;
                System.out.println(++a + b++ *c);
        }
}
```

Resultado: 611

46. Pregunta

```
public class Shop{

                public static void
                        main(String[] args)
                        { new
                        Shop().go("welcom
                        e",1);
```

```
                new Shop().go("welcome", "to", 2);
                }
                public void go (String... y, int
                        x){
                        System.out.print(y[y.len
                        gth-1]+"");
        }

}
```

Resultado: Compilation fails

47. Pregunta

```
class Plant {

        Plant() {

        }


System.out.println("plant");

}

class Tree extends Plant {

        Tree(String type) {

                System.out.println(type);
        }

}

        class Forest
                extends
                Tree {
                Forest() {
                super("leaves");
                new
                Tree("leaves");
        }
        public static void main(String[]
        args) { new Forest();
        }

}
```

Resultado: plant, leaves, plant, leaves

48. Pregunta

```
class Test {

        public static void
                main(String[] args)
                { String s1 =
                "hello";
        String s2 = new String ("hello");
        s2=s2.intern(); // el intern() asigna el mismo hash conforme a la
        cadena System.out.println(s1==s2);
    }

}
```

Resultado: true

49. ¿Cuál de las siguientes construcciones es un ciclo infinito while?:

- while(true);.
- while(1==1){}.

50. Pregunta

```
public class Main {

        public static void
                main(String[] args)
                { int a= 10;
        int b =37;

        int z= 0;

        int w=
        0; if
        (a==b){
                z=3;

                }else if(a>b){

        z=6;

        }

    w=10*z;
    System.out.println(
    z);
    }

}
```

Resultado: 0

51. Pregunta

```
public class Main{

        public static void
            main(String[] args)
            { course c = new
            course();
            c.name="java";
            System.out.println(
            c.name);
        }

    }

class course {

    String
    name;
    course(){
        course c = new
        course();
        c.name="Oracle";
    }

}
```

Resultado: Exception StackOverflowError

52. Pregunta

```
public class Main{

        public static void
            main(String[] args) {
            String a;
            System.out.println(a.t
            oString());
    }

}
```

Resultado: builder fails

53. Pregunta

```
    public class
        Main{
        public static void
            main(String[] args) {
            System.out.println(2+
            3+5);
            System.out.println("+"
            +2+3+5);
    }
```

```
}
```

Resultado: 10 + 235

54. Pregunta

```
public class Main {

		public static void
			main(String[] args)
			{ int a = 2;
		int b =
		2; if
		(a==b)
		System.out.println("Here1");
	if (a!=b)
	System.out.println("here2"
	); if (a>=b)
	System.out.println("Here3
	");

	}

}
```

Resultado: Here1 , Here 3

55. Pregunta

```
public class Main extends count {

		public static void
			main(String[] args) {
			int a = 7;
			System.out.println(co
			unt(a,6));
	}
}

class count {

	int count(int x, int y){return x+y;}

}
```

Resultado: builder fails

56. Pregunta

```
class trips{

	void main(){
```

```
                System.out.println("Mountain");

        }
                static void main (String
                        args){
                        System.out.println(
                        "BEACH");
        }
                public static void main (String
                        [] args){
                        System.out.println("ma
                        gic town");
        }

        void mina(Object[] args){

                System.out.println("city");

        }

}
```

Resultado: magic town

57. Pregunta

```
public class Main{

                public static void
                        main(String[] args)
                        { int a=0;
                        System.out.println(
                        a++ +2);
                        System.out.println(
                        a);
        }
}
```

Resultado: 2, 1

58. Pregunta

```
public class Main{

                public      static      void
                        main(String[]  args)
                        {  List<E>  p  =new
                        ArrayList<>();
                        p.add(2);
                p.add(1);

                p.add(7);

                p.add(4);
```

```
        }
}
```

Resultado: builder fails

59. Pregunta

```
public class Car{
        private void accelerate(){
                System.out.println("car acelerating");
        }
        private void break(){
                System.out.println("car breaking");
        }
                public void control
                        (boolean faster){
                        if(faster==true)
                        accelerate();
                else
                break(
                );
                }
                public static void main
                        (String [] args){ Car
                        car = new Car();
                        car.control(false);
        }
}
```

Resultado: break es una palabra reservada

60. Pregunta

```
class App {
        App() {
        }

 System.out.println("1");

        App(Integer num) {
```

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

        }

        App(Object num) {

            System.out.println("4");

        }
            App(int num1, int num2,
                int num3) {
                System.out.println
                ("5");
        }
            public static void
                main(String[] args)
                { new App(100);
            new App(100L);

        }

}
```

Resultado: 3, 4

61. Pregunta

```
class App {

            public static void
                main(String[] args)
                { int i=42;
            String s = (i<40)?"life":(i>50)?"universe":"
            everything"; System.out.println(s);
        }
}
```

Resultado: everything

62. Pregunta

```
class App {

        App(){


        }

System.out.println("1");


        App(int num){

            System.out.println("2");
```

```
        }

        App(Integer num){

                System.out.println("3");

        }

        App(Object num){

                System.out.println("4");

        }
                public static void main(String[]
                        args) { String[]sa =
                        {"333.6789","234.111"};
                NumberFormat inf=
                NumberFormat.getInstance();
                inf.setMaximumFractionDigits(2);
                for(String s:sa){

                        System.out.println(inf.parse(s));

                }

        }

}
```

Resultado: java: unreported exception java.text. ParseException; must be caught or declared to be thrown

63. Pregunta

```
class Y{


        public static void main(String[]
                args) { String s1 =
                "OCAJP";
        String s2 = "OCAJP" + "";
        System.out.println(s1 == s2);

        }

}
```

Resultado: true

64. Pregunta

```
class Y{
```

```
        public static void main(String[]
                args) { int score = 60;
               switch (score) {
                        default:
                 System.out.println("Not a valid score");
                 case score < 70:
                 System.out.println("Failed");
                 break;

                  break;

          }

}


  case score >= 70: System.out.println("Passed");
```

Resultado: Error de compilacion - java: reached end of file while parsing

65. Pregunta

```
class Y{


        public static void main(String[]
                args) { int a = 100;
                System.out.println(-a++);

          }
}
```

Resultado: -100

66. Pregunta

```
class Y{


        public static void main(String[]
                args) { byte var = 100;
        switch(var) {

                case 100:

                System.out.println("var is 100");
                break;
                case 200:
                System.out.println("var is 200");
                break;
                default:
                System.out.println("In default");
```

```
                }

        }

}
```

Resultado: Error de compilacion - java: incompatible types: posible lossy conversion from int to byte

67. Pregunta

```
class Y{

        public static void main(String[]
                args) { A obj1 = new A();
        B obj2 = (B)obj1;
        obj2.print();


}

class A {


}


public void print(){

        System.out.println("A");


        }

}

class B extends A {

        public void print(){

                System.out.println("B");

        }

}
```

Resultado: ClassCastException

68. Pregunta

```
class Y{

        public static void main(String[]
                args) { String fruit =
                "mango"; switch (fruit) {
                default:
```

```
                System.out.println("ANY FRUIT WILL DO");
                case "Apple": System.out.println("APPLE");
                case "Mango":
                System.out.println("MANGO"); case
                "Banana":
                System.out.println("BANANA");
                break;

                 }

        }

}
```

Resultado: ANY FRUIT WILL DO, APPLE, MANGO, BANANA

69. Pregunta

```
abstract class Animal {

        private String
        name;
        Animal(String
        name) {
                this.name = name;
        }

        public String getName() {

                return name;

        }

}

class Dog extends Animal {

        private String
        breed;
        Dog(String breed)
        {

                this.breed = breed;

        }

                Dog(String name,
                        String breed) {
                        super(name);
                this.breed = breed;

        }

        public String getBreed() {
```

```
                return breed;

        }

}

class Test {

                public static void
                        main(String[] args) {
                        Dog dog1 = new
                        Dog("Beagle");
                Dog dog2 = new Dog("Bubbly", "Poodle");
                System.out.println(dog1.getName() + ":" +
                dog1.getBreed() + ":" + dog2.getName() + ":" +
                dog2.getBreed());
        }

}
```

Resultado: compilation fails

70. Pregunta

```
public class Main {

                public static void main(String[] args) throws
                        ParseException { String[]sa =
                        {"333.6789","234.111"};
                NumberFormat nf =
                NumberFormat.getInstance();
                nf.setMaximumFractionDigits(2);
                for (String s: sa ) {
                        System.out.println(nf.parse(s));
                }

        }

}
```

Resultado: 333.6789, 234.111

71. Pregunta

```
public class Main
                 {
                public static void main(String[] args) throws
                        ParseException { Queue<String>
                        products = new ArrayDeque<String>();
                        products.add("p1");
                products.add("p2");
                products.add("p3");
                System.out.println(products.peek
                ());
```

```
                System.out.println(products.poll()
                ); System.out.println("");
                        products.forEach(s
                        -> {
                        System.out.println(s)
                        ;
                });

        }

}
```

Resultado: p1, p1, , p2, p3.

72. Pregunta

```
public class Main
                 {
                public static void main(String[] args) throws
                        ParseException {
                        System.out.println(2+3+5);
                        System.out.println("+"+2+3*5);
        }
}
```

Resultado: 10 + 215