Preguntas Java 3ra Vuelta

1.
```
public class Test5 {
     public static void main(String args[]) {
          Side primerIntento = new Head();
          Tail segundoIntento = new Tail();
          Coin.overload(primerIntento);
          Coin.overload((Object)segundoIntento);
          Coin.overload(segundoIntento);
          Coin.overload((Side)primerIntento); } }
interface Side { String getSide(); }
class Head implements Side {
     public String getSide() { return "Head"; } }
class Tail implements Side {
     public String getSide() { return "Tail"; }  }
class Coin {
     public static void overload(Head side) {
          System.out.println(side.getSide()); }
     public static void overload(Tail side) {
          System.out.println(side.getSide()); }
     public static void overload(Side side) {
          System.out.println("Side"); }
     public static void overload(Object obj) {
          System.out.println("Object"); } }
```

Side Object Tail Side

2. ¿Qué 3 líneas producen el siguiente resultado?

Shape: cpnstructor
Shape: foo
Square: foo

```
public class SuperTest {
   public static void main(String[] args) {
       //statement1
```

```
      //statement2
      //statement3 } }
class Shape {
   public Shape() {
       System.out.println("Shape: constructor"); }
     public void foo() {
       System.out.println("Shape: foo"); } }
class Square extends Shape {
   public Square() { super(); }
   public Square(String label) {
       System.out.println("Square: constructor"); }
   public void foo() { super.foo(); }
   public void foo(String label) {
       System.out.println("Square: foo"); } }
```

D: Square square = new Square(); square.foo(); square.foo("bar");

Para que se imprima la primera línea, tiene que pasar por el constructor vacío de Shape, esto se puede hacer creando un objeto Shape o Square. Para que imprima la segunda línea debe pasar por la función foo de Shape, la función Square llama a la función foo de Shape, por lo que sigue sin importar el tipo de objeto que se creó. Para imprimir la tercer línea tiene que pasar por la función foo con un String de Square, por lo que el objeto creado tiene que ser Square, aunque la variable sea Shape o Square

3. ¿Cuál de las siguientes opciones son instanciaciones e inicializaciones válidas de un arreglo multidimensional?

   A: `int[][] array2D = {{0, 1, 2, 4,}{5, 6}}; //Esta no es válida porque le falta una coma para separar los arreglos internos`

   B:
   ```
   int[][] array2D = new int[][2];
   array2D[0][0] = 1;
   array2D[0][1] = 2;
   ```

```
   array2D[1][0] = 3;
   array2D[1][1] = 4; //Esta no es válida porque es necesario que el primer nivel del arreglo este definido
```

C: `int[] array3D = new int[2][2][2]; //Esta no es correcta porque de un lado declara un arreglo unidimensional y del otro declara un arreglo tridimensional`

D: `int[][][] array3D = {{{0, 1}, {2, 3}, {4, 5}}};`

E: `int[] array = {0, 1};`

F:
```
array3D[0][0] = array;
array3D[0][1] = array;
array3D[1][0] = array;
array3D[1][1] = array; //Si array3D ya fue declarado con 3 dimensiones y array ya fue declarado con 1 dimensión, es una inicialización válida
```

4.
```
public 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(); } }
```

20

calc asigna la multiplicación 2 * 10 a la variable num del objeto

5.
```
class Feline {
     public String type = "f";
     public Feline() { System.out.println("feline"); } }
public class Cougar extends Feline {
     public Cougar() { System.out.println("cougar"); }
```

```
void go() {
     type = "c";
     System.out.println(this.type + super.type); }
public static void main(String[] args) {
     new Cougar().go(); } }
```

feline cougar c c

Ya que Cougar hereda de Feline, primero pasa por el constructor de Feline y luego por el de Cougar al crear un objeto Cougar. Ya que Cougar no define su propia variable, type se modifica globalmente al entrar en la función go

6.
```
interface Rideable { String getGait(); }
public class Camel implements Rideable {
     int weight = 2;
     String getGait() { return " mph, lope"; }
     void go(int speed) {
          ++speed;
          weight++;
          int walkrate = speed * weight;
          System.out.println(walkrate + getGait()); }
     public static void main(String[] args) {
          new Camel().go(8); } }
```

Error de compilación

Cuando una clase escribe el método de una interfaz, debe ser public ya que no se puede reducir la visibilidad y los métodos de las interfaces son implícitamente public

7.
```
class ClassA {}
class ClassB extends ClassA {}
class ClassC extends ClassA {}
```

```
ClassA p0 = new ClassA();
ClassB p1 = new ClassB();
ClassC p2 = new ClassC();
ClassA p3 = new ClassB();
ClassA p4 = new ClassC();
```

A: p0 = p1

B: p1 = p2 //Esta no se puede porque ClassB y ClassC son hermanas

C: p2 = p4

D: p2 = (ClassC)p1 //ClassCastException

E: p1 = (ClassB)p3

F: p2 = (ClassC)p4

8.
```
class X {}
class Y{ Y() {} }
class Z{ Z(int i) {} }
```

Solo X tiene un constructor default

Aunque el constructor de Y es igual que el default, ese fue declarado por el desarrollador

9.
```
Map<String, List<? extends CharSequence>> stateCitiesMap =
new HashMap<String, List<? extends CharSequence>>();
```

¿Cuál de las siguientes opciones logra la misma declaración usando inferencia de tipos?

A:
```
Map<String, List<? extends CharSequence>>
stateCitiesMap = new HashMap<String, List<>>();
```

B:
```
Map<String, List<? extends CharSequence>>
stateCitiesMap = new HashMap();
```

C:
```
Map<String, List<? extends CharSequence>>
stateCitiesMap = new HashMap<>();
```

D: Map<String, List<? extends CharSequence>>
stateCitiesMap = new HashMap<<>>();

10.
```
int i = 1;
int j = i++;
if((i == ++j) | (i++ == j))
     i += j;
System.out.println(i);
```

5

j = 1. i = 2. ((2 == 2) | (3 == 2)) = (true | false) = true.
i = 3. j = 2

11. ¿Qué implementación de max() retornará correctamente el número más grande?

A:
```
int max (int x, int y){
          return( if(x>y){x;} else {y;}); }
```
B:
```
int max (int x, int y){
          return( if(x>y){return x;} else {return y;}); }
```
C:
```
int max (int x, int y){
          switch(x<y){
               case true: return y;
               default: return x; }; }
```
D:
```
int max (int x, int y){
          if(x>y) return x; return y; }
```

12. ¿Qué imprimirá cuando se ejecute sin argumentos?

```
public class TestClass {
     public static int m1(int i) { return ++i; }
     public static void main(String[] args) {
          int k = m1(args.length);
          k += 3 + ++k;
          System.out.println(k); } }
```

6

Si se ejecuta sin argumentos, args.length = 0, por lo tanto k = 1. Cuando se encuentra el operador += se resuelve primero el lado derecho y luego se suma el lado izquierdo:

k += 3 + 2; -> k += 5; -> k = 1 + 5; -> k = 6

13.

```
public class PromotionTest {
     public static void main(String args[]) {
          int i = 5;
          float f = 5.5f;
          double d = 3.8;
          char c = 'a';
          if (i == f) c++;
          if ((int)(f + d) == ((int)f + (int)d))
               c += 2;
          System.out.println(c); } }
```

a

El primer if da false porque 5 no es lo mismo que 5.5. El segundo if da false porque ((int)(9.3) == (5 + 3)) -> (9 == 8) -> false. El valor de c nunca se modifica

14.

```
public class ForSwitch {
public static void main(String args[]) {
     char i;
     LOOP: for (i=0; i<5; i++) {
          switch(i++) {
               case '0': System.out.println("A");
               case 1:
                    System.out.println("B");
                    break LOOP;
               case 2:
                    System.out.println("C"); break;
```

```
            case 3:
                 System.out.println("D"); break;
            case 4: System.out.println("E");
            case 'E':
                 System.out.println("F"); } } } }
```

C E F

15. ¿Qué es verdadero si se ejecuta el programa en la línea de comandos: java Test closed?

```
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]); } }
```

A: ArrayIndexOutOfBoundsException

B: Termina sin excepciones y no imprime nada

C: Imprime “Go away”

D: Imprime “Go away” y ArrayIndexOutOfBoundsException

E: Ninguna de las anteriores

16. ¿Cuántos objetos se crean en la función main?

```
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 { }
```

2

c apunta a un objeto ya creado

17.
```
public class TestClass {
     static char ch;
     static float f;
     static boolean bool;
     public static void main(String[] args) {
          System.out.println(f);
          System.out.println(“ ”);
          System.out.println(ch);
          System.out.println(“ ”);
          System.out.println(bool); } }
```

0.0 false

Son los valores por default

18. ¿Qué escribir en Line 1 para que el programa imprima x?

```
public class AccesTest {
          String a = “x”;
          static char b = ‘x’;
          String c = “x”;
          Class Inner{
               String a = “y”;
               String get(){
                    String c = “temp”;
                    //Line 1
                    return c; } }
          AccesTest() {
               System.out.println(new Inner().get()); }
          public static void main(String[] args) {
```

```
                    new AccesTest(); } }
```

A: c = c; //temp
B: c = this.a //y
C: c = “”+AccesTest.b;
D: c = AccesTest.this.a;
E: c = “”+b;

19.

```
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); } }
```

0

No llega a entrar al for

20. ¿Cuáles 3 líneas son correctas?

```
public class Speak{
     public static void main(String[] args){
          Speak speakIT = new Tell();
          Tell tellIT = new Tell();
          speakIT.tellItLikeItIs();
          (Truth)speakIT.tellItLikeItIs();
          ((Truth)speakIT).tellItLikeItIs();
          tellIT.tellItLikeItIs();
          (Truth)tellIT.tellItLikeItIs();
```

```
          ((Truth)tellIT).tellItLikeItIs(); } }
class Tell extends Speak implements Truth{
     @Override
     public void tellItLikeItIs(){
          System.out.println(“Right on!”); } }

Interface Truth{
     Public void tellItLikeItIs(); }
```

A: ((Truth)speakIT).tellItLikeItIs();

B: tellIT.tellItLikeItIs();

C: ((Truth)tellIT).tellItLikeItIs();