Preguntas Java Cuestionario 1

1. C: java.lang.NumberFormatException

2.

3. B: Mock – indica a JUint que la propiedad que usa esta anotación es una simulación, se inicializa como tal y es susceptible a ser inyectada por @InjectMocks

4. 3true1

   Primero imprime la longitud del primer arreglo interno. Después pregunta si el segundo arreglo interno es un arreglo, entonces imprime true. Al final imprime el segundo elemento del primer arreglo interno

5. A: Una interfaz puede heredar a otra interfaz

   B: Una clase abstracta puede heredar a una clase concreta

   C: Una clase abstracta puede heredar a otra clase abstracta

   D: Una clase abstracta puede implementar una interfaz

   E: Una interfaz no puede heredar a una clase (abstracta), debe ser implementada

6. A: `private int blipvert(long x) {return 0:}` `//Se puede porque es un overloading`

   B: `protected int blipvert(long x) {return 0:}` `//Se puede porque es un overloading`

C: protected long blipvert(int x, int y) {return 0:} //Se puede porque es un overloading

D: public int blipvert(int x) {return 0:} //Se puede porque implementa la función correctamente

E: private int blipvert(int x) {return 0:} //No se puede porque reduce la visibilidad

F: protected long blipvert(int x) {return 0:} //No se puede porque no implementa la función correctamente, debería devolver un int

G: protected long blipvert(long x) {return 0:} //Se puede porque es un overloading

7. B: Cambiar la línea 13 a public Sub(){super(5);}

   D: Cambiar la línea 13 a public Sub(){this(5);}

8. A: Compila sin errores

9. atom granite atom granite

   Al crear un objeto Mountain pasa primero por el costructor vacío de Atom (imprime “atom ”), después pasa por el constructor vacío de Rock y después pasa por el constructor vacío de Mountain. En el constructor vacío de Mountain pasa por el constructor con un String de Rock (imprime “granite”) y crea un objeto Rockcon un String. Primero pasa por el constructor vacío de Atom (imprime “atom ”) y después pasa por el constructor con un String de Rock (imprime “granite”)

10. three

    La función principal es publica, estática y devuelve void, puede o no ser final

11. feline cougar c c

    Cuando se crea un objeto Cougar primero pasa por el constructor vacío de Feline (imprime “feline ”) y después pasa por el constructor vacío de Cougar (imprime “cougar ”). Al entrar a la función go declara la variable type como “c” y luego la imprime dos veces ya que aunque la llama desde diferentes clases, comparten la variable, es la misma

12. Error de compilación

    Si la variable y el objeto asignado a esa variable no son de la misma clase, la variable de referencia debe ser de la super clase y el objeto, de la sub clase

13. InputMismatchException

    Scanner devuelve una cadena. La función solo lee enteros, por lo que lanza la excepción al llegar al carácter después del primer 1 en la cadena

14. 1 thrown to main

    El hilo que se crea no hace nada. La combinación de la función wait(5000) y el synchronized hace que se lance la InterruptedException. No se desborda el programa pero no llega a imprimir 2

15. A: 420 es la salida

16. B: El Singleton tiene una sola instancia de una clase y puede ser usada por otras clases

17. 111.234 222.5678

El throws ParseException Evita que marque un error dde compilación al intentar convertir una cadena en número. El setMaximumFractionDigits establece la cantidad de números después del punto pero solo funciona cuando va de números a String, no al revés. Es decir, no altera las cantidades cuando convierte de String a número

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

Para imprimir la primera línea tiene que pasar por el constructor vacío de Shape, por lo que se puede crear un objeto Shape o un objeto Square. Para imprimir la segunda línea tiene que pasar por la función foo de Shape, la función vacía foo de Square llama a la función foo de Shape por lo que un objeto Shape o un objeto Square pueden funcionar igualmente. Para imprimir la tercera línea, el objeto debe ser Square y llamar a la función foo con un String

19. A: class Test implements SampleCloseable{
public void close() throws java.io.IOException{}}

B: class Test implements SampleCloseable{
public void close() throws Exception{}} //No se puede porque Exception es padre de IOException

C: class Test implements SampleCloseable{
public void close() throws FileNotFoundException{}}
//Se puede porque FileNotFounException es hija de IOException

D: class Test extends SampleCloseable{
public void close() throws java.io.IOException{}}
//Las clases no pueden heredar de interfaces

E: class Test implements SampleCloseable{
Public void close(){}}

20. 4

Guarda 4 porque se declaró el método equals pero no el método hashCode

21. int x = 2, y = 3, z = 4;

La única variable que se imprime es k. k es un entero. El entero más grande que se imprime es 4. k se incrementa antes de imprimirse. k se imprime siempre que sea menor a z. Por lo tanto, z = 4. Solo hay una opción con z = 4

22. C: ((Truth)speakIT).tellItLikeItIS();

D: tellIT.tellItLikeItIS();

F: ((Truth)tellIT).tellItLikeItIS();

Para que se pueda implementar una función, la variable y el objeto la tienen que conocer

23. [7, 5, 1]

remove elimina el primer elemento que encuentra que coincida

24. 2 2

La primera vez que imprime, b = 3, pero con el pre incremento, imprime 2. La segunda vez solo vuelve a imprimir b = 2

25. throws indica el tipo de excepción que la función no va a manejar. throw arroja una excepción

26. sc = asc;

Del lado izquierdo van las super clases y del lado derecho van las subclases, cuando la variable de referencia y el objeto no son de la misma clase

27. 4 ArrayIndexOutOfBoundsException

No existe el elemento 4 en el segundo arreglo de array

28. A: p0 = p1:

E: p1 = (ClassB)p3;

F: p2 = (ClassC)p4;

Aunque la variable de referencia sea una superclase y el objeto sea una sub clase, el objeto es de la super clase, por lo que se tiene que realizar un casteo implícito si una variable de la sub clase quiere apuntar a ese objeto. La segunda opción no se puede porque son clases hermanas. La cuarta opción lanza un ClassCastException

29. C: Class2 has-a v2

D: Class3 has-a v1

F: Class2 has-a Class1

Si una clase tiene una variable u objeto de otra clase, entonces has-a esa clase y sus atributos

30. 7 5 3 2

Al hacer que el segundo parámetro llame a compareTo para compararlo con el primer parámetro, hace que se ordene descendentemente

31. A: Before if clause After if clause

    B: Before Exception

    En la segunda posibilidad, el programa se desborda porque se lanza la excepción pero no se maneja

32. Found Red Found Blue Found White

    Cuando encuentra el elemento que coincide, ejecuta los siguientes bloques hasta encontrar un break o terminar

33. 12

    La función no devuelve la variable

34. Más de una línea tienen que ser removidas para que se pueda hacer la compilación

    La línea 10 intenta castear un objeto a una sub clase. La línea 12 intenta castear un objeto a una clase hermana

35. the fox lazy ...

36. Error de compilación

    En la línea 18 hay “ en lugar de =

37. D: class Base3{ abstract int var1 = 89; }

    Las variables no pueden ser abstractas

38. Result: 235 Result: 215

Ya que hay un String, toma el símbolo + como concatenación

39. True y MyStuff no cumple con el contrato de Object.quals

El método equals devuelve true a menos que el name del objeto con el que se compara sea null. No cumple con el contrato porque Object.equals pregunta si se apunta al mismo objeto

40.

41. Error de compilación

Cuando una clase implementa el método de una interfaz, debe ser public para no reducir su visibilidad

42. hello

El constructor cambia el valor de str

43. 3 5

A partir del índice 1 toma dos valores, que cambia por lo que hay en el índice 2 y 3 = [1, 3, 4, 4, 5]

44. 0

Se multiplica por 0

45. Error de compilación

while espera un boolean, no un entero

46. Cambiar la función one de private a protected

La función two no puede acceder a one

47. B: int a, b, c;

D: int d, e, f;

int va con minúsculas. Solo se escribe una vez int si es en la misma línea o si es antes del ;

48. A: public void foo(){}

B: private void foo(){} //No se puede reducir la visibilidad

C: protected void foo(){} //Protected tiene más visibilidad que default

D: int foo(){} //Es un overriding

49.

50. Result: 35 Result: 8

Al haber un String, lo toma como concatenación, a menos que este entre paréntesis

51. StringIndexOutOfBoundsException

La cadena solo tiene hasta el índice 10

52. Cambiar price=4; por price.price=4;

price=4 intenta asignar un entero a un objeto Simple. La solución sería asignar 4 al atributo drice del objeto Simple que se creó (price)

53. A: Set no puede contener elementos duplicados

B: Set no tiene orden

54. sc: AnotherSampleClass asc: AnotherSampleClass

Se declara que la variable sc apunta a un objeto asc

55. boolean j = (1<5);

Los booleanos solo aceptan valores true o false

56. C: class Fred1{ public static void main(String[] args) { System.out.println(args[1]); } }

D: class Fred1{ public static void main(String[] args) { System.out.println(args); } }

La segunda opción se descarta porque los parámetros de main no está correctamente declarados. La primera opción se descarta porque llama al argumento en el índice 2, pero solo se mandaron dos argumentos, entonces lanzaría un ArrayIndexOutOfBoundsException. La tercera opción imprimiría “walls.”; y la segunda, la referencia de memoria donde se guarda args

57. 1

El break hace que se rompa el ciclo while pero el ciclo for hace que vuelva a entrar en él con valores diferentes ya que ii solo se declara como 0 afuera del for

58. A: array2D[0][0] = 1; array2D[0][1] = 2; array2D[1][0] = 3; array2D[1][1] = 4; //Inicialización

B: array3D[0][0] = array; array3D[0][1] = array; array3D[1][0] = array; array3D[0][1] = array; //Suponiendo que array es un arreglo unidimensional previamente declarado, no se puede porque array3D es un arreglo bidimensional, no tridimensional

C: int[][] array2D = {0, 1}; //Declara un arreglo bidimensional y lo inicializa como unidimensional

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

E: int [][] array2D = {{0, 1, 2, 4}{5, 6}}; int[][] array2D = new int[2][2]; //Falta la coma que separa los arreglos

59. La API estándar para acceder a bases de datos en java es JDBC

60. if(h1.equals(h2))

equals compara el contenido. == compara si es el mismo objeto. same no existe

61. NullPointerException

Trata de hacer una operación aritmética con un Integer que nunca fue inicializado

62. Cuando una variable se declara como protected puede ser accedida desde cualquier clase en el mismo paquete

63. Error de compilación

No se puede declarar una variable que usa una clase padre en una clase hijo

64. Error de compilación

Para que doStuff pueda imprimir myScope.z tendría que ser pasado como parámetro

65. Error de compilación

Cuando se usan los varargs, deben ser el último parámetro

66. Error de compilación

assert debe devolver un booleano no un String

67.

68. Bob’s name: Jian

Ya que jian apunta al mismo objeto que bob, cambió el nombre de Bob por Jian

69. Solo X

Aunque Y tiene un constructor como el que se hace por default, este fue declarado por el desarrollador

70. Palabras que pueden ir antes de class: public, static

71. El patrón de diseño Builder se usa para crear objetos complejos de forma más sencilla

72. 1 2

    La primera vez imprime a antes de hacer el incremento. Hace el incremento y luego vuelve a imprimir a

73. 11

    Ya que jj empieza en 0 y solo se incrementa en el for, debe entrar 5 veces. La condición es que ii sea mayor que 6 y se va decrementando de uno en uno. 5 + 6 = 11

74. Un bloque de código static significa que se ejecuta incluso antes que la función main

75. 1 2 3 4

    Este es el camino que sigue: Constructor con un entero (5) de Y. Constructor vacío de Y. Constructor con un entero (6) de X. Constructor vacío de X

76. A: Cambiar el catch a IOException `//No es necesario manejar la RuntimeException para que compile, pero si IOException`

    B: Poner throws IOException en el doSomething y en el main `//No maneja la excepción pero si la arroja`

77. Error de compilación

    Si no se sabe cuántos argumentos se pasarán se pueden usar los varargs

78. 6 5 6 4

Lo primero que imprime es la variable que se declara en la función main. Lo segundo es la variable que se declara en doStuff; doStuff2 cambia el valor de la variable del objeto. Lo tercero es otra vez la variable que se declara en la función main. Lo último es la variable del objeto