Preguntas Java Cuestionario 3

1. Boolean j = (1<5)

   Boolean guarda valores true o false

2. El patrón de diseño DTO se usa para intercambiar datos entre procesos

3. 11

   Ya que jj se inicializa en 0 y solo se incrementa dentro del for, además de que el for decrementa la variable, se necesita que kk = 5 + 6

4. Hash is: 111111, 44444444, 9999999999

   Se toma la función del objeto, no de la variable

5. 0 1 2

   En el while imprime 0 1 2 y hace que ii = 3. Cuando sale también sale del do while

6. aa, 0 aa, 1 aa, 2 bb, 0 bb, 1 bb, 2 cc, 0 cc, 1 cc, 2

7. 5: abstract class Dog{, 9: public class Poodle extends Dog{

   Un método abstracto no puede estar en una clase que no sea abstracta. Las clases heredan de otras clases, no las implementan

8. StringBuilder sb = new StringBuilder(128);

   La primera opción no se puede porque intenta asignar un String a un StringBuilder. Los métodos setCapacity(int) y getInstance(int) no existen

9. 20

   La función calc asigna a la variable de obj num la multiplicación 2*10

10. A: Reemplazar la línea 8 con handy.dandy.KeyStroke stroke = new KeyStroke(); //No funciona porque se queda el error de lado derecho

    B: Reemplazar la línea 8 con handy.*.KeyStroke stroke = new KeyStroke(); //No funciona porque el asterisco no puede ir en medio

    C: Reemplazar la línea 8 con handy.dandy.KeyStroke stroke = new handy.dandy.KeyStroke();

    D: Importar handy.* antes de la línea 1 //No funciona porque los imports van después del package

    E: Importar handy.dandy.* después de la línea 1

    F: Importar handy.dandy.KeyStroke después de la línea 1

    G: Importar handy.dandy.KeyStroke.typeExclamation() después de la línea 1 //No funciona porque tiene que importarse toda la clase

11. Exception Finally

    Se lanza una excepción al querer dividir entre 0

12. Override

    Permite que diferentes métodos con la misma firma tengan diferentes implementaciones

13. 1 2 3 4 5

    10, 1 -> true. 9, 2 -> true. 8, 3 -> true. 7, 4 -> true. 6, 5 -> true. 5, 6 -> false. Entonces, entra 5 veces en el bucle.

14. 1 2 3 4 5 7 9

    Set no acepta valores duplicados

15. 8 7

    El 0 antes de los números significa que están en el sistema octal

16. El programa compilará, se llama hiding o shadowing

17. A: static es independiente del modificador de acceso que use una clase

    B: Las clases static pueden contener métodos no static

18. A: Un constructor es llamado cuando se crea un objeto

19. Los tipos primitivos son char, int, boolean, float, long, double, short, byte

20. La clase Integer hereda de Number

21. Singleton

Solo se crea una instancia

22. A: String temp[] = new String{“j”, ”a”, ”z”}; //No esta declarando un nuevo String, sino un array

B: String temp[] = {“j” ”b” ”c”}; //Necesita comas para separar los elementos

C: String temp = {“a”, ”b”, ”c”}; //Esta declarando un String no un array

D: String temp[] = {“a”, ”b”, ”c”};

23. 9

La función no devuelve ningún valor

24. A: Los archivos java pueden contener o no package, pero solo uno

B: Si un archivo java tiene package e import, el import va después del package

C: Un archivo java no siempre tiene clases definidas, puede ser un enum, interfaz u otra cosa

D: Si un archivo java tiene package, debe ser la primera línea (sin contar los comentarios)

25. Error de compilación

Para implementar la interfaz Comparator se necesita implementar su método compare correctamente. Para que el

Override sea correcto, se debe respetar sus parámetros, que son dos Object, no Strings

26. Para borrar todos los pares de llave-valor de un HashMap, se usa el método clear()

    remove() elimina un solo registro. clearAll() y empty() no existen

27. Factory

    Porque devuelve un calendario según nuestra fecha y zona horaria

28. Inside BaseC::method

    El objeto que se crea es ImplC, aunque esta clase no tiene el método method, sí hereda de una clase que sí lo tiene

29. A: (b instanceof B)

    B: (b instanceof B) && (!(b instanceof A)) //true && (!true) = true && false = false – Como B hereda de A, b también se considera una instancia de A

    C: (b instanceof B) && (!(b instanceof C)) //true && (!false) = true && true = true – b no se puede considerer instancia de C porque C es una clase hija de B

30. Null

    La función Constructor no es el constructor porque devuelve void