empleado

Pregunta 1

Finalizado

Puntúa como 1,00

Marcar pregunta

Has escrito un bloque de código para imprimir diferentes mensajes en base al resultado de una tirada de un dado de seis caras. Cuando la tirada sale entre 3 y 6 (incluidos), quieres q muestre un mensaje especial. ¿Cómo habría que completar el código para implementar dicha lógica?

```
when(dado){
    1 -> println("Ha salido un 1")
    2 -> println("Ha salido un 2")
    _____ -> println("Premio!")
    else -> println("Utiliza un dado de 6 caras, por favor")
}
```

- a. 3:6
- b. 3 to 7
- c. range(3,6)
- d. 3,4,5,6

Pregunta 2

Finalizado

Puntúa como 1,00

Marcar pregunta

Considerando el siguiente código, selecciona la afirmación correcta:

```
open class Medico
data class Psiquiatra(val nombre: String): Medico()

fun main(){
    val cuadroMedico: MutableList<Medico> = mutableListOf(Psiquiatra("Ana"), Psiquiatra("Pedro"), Psiquiatra("Espe"))
    val medicos: List<Medico> = cuadroMedico

    cuadroMedico.add(Psiquiatra("Iker"))
    println(medicos.count())
}
```

- a. Se imprime *4* por pantalla
- b. El código no compila porque no se puede añadir un elemento a una lista no mutable
- c. El código no compila porque se intenta añadir instancias de *Psiquiatra* en una lista de tipo *Médico*
- d. Se imprime *3* por pantalla

Pregunta 3
Finalizado
Puntúa como 1,00
Marcar pregunta

Señala el valor que el siguiente extracto de código muestra por pantalla:

```
val conjunto = setOf("rojo", "amarillo", "azul", "rojo")
println(conjunto.count())
```

- a. 5
- b. 4
- c. 1
- d. 3

Pregunta 4
Finalizado
Puntúa como 1,00

Considera la clase *Personaje* con dos atributos: *nombre* y *nivel*.

```
fun main() {
    val personaje = Personaje("Frodo", 2)
```

Pregunta 4

Finalizado

Puntúa como 1,00

Marcar pregunta

Considera la clase *Personaje* con dos atributos: *nombre* y *nivel*.

```
fun main() {
    val personaje = Personaje("Frodo", 2)
    val (nombre, nivel) = personaje
}
```

Usar la variable *personaje* así:

a. Solo es posible si Personaje es un *singleton/object*

b. Es posible si la clase *Personaje* implementa *component1()* y *component2()*

c. Solo es posible si *Personaje* es una *data class*

d. Es posible para cualquier tipo de clase con únicamente dos atributos

Pregunta 5

¿Cuál es la diferencia entre la declaración de variables con var y val?

c. Solo es posible si *Personaje* es una *data class*

d. Es posible para cualquier tipo de clase con únicamente dos atributos

Pregunta 5

Finalizado

Puntúa como 1,00

Marcar pregunta

¿Cuál es la diferencia entre la declaración de variables con var y val?

a. var se utiliza para variables mutables, mientras que val se utiliza para variables inmutables

b. var requiere de inicialización al declararla y val no

c. var se utiliza para variables inmutables, mientras que val se utiliza para variables mutables

d. var y val son intercambiables; no hay diferencia

Pregunta 6

Finalizado

Puntúa como 1,00

Marcar

¿Por qué no compila este código?

```
val sumador = 2
infix fun Int.add(otro: Int=1) = this + sumador
```

Pregunta 6
Finalizado
Puntúa como 1,00
Marcar pregunta

¿Por qué no compila este código?

```
val sumador = 2
infix fun Int.add(otro: Int=1) = this + sumador

fun main(){
    println(10 add 3)
}
```

- a. La función infix debe de ser declarada como pública
- b. Desde una función infix no se puede acceder a una variable de fuera de la clase (*sumador*)
- c. La funciones que extienden una clase utilizan *it* y no *this* para referirse al objeto sobre el que se aplican
- d. Las funciones infix no pueden tener parámetros con valores por defecto

Pregunta 7

Finalizado

Puntua como 1,00

Desmarcar

Observando el siguiente código se puede afirmar:

```
class M : A {
    override fun funcionA() { /**/ }
    override fun funcionB() { /**/ }
}

class P : A {
    override fun funcionA() { /**/ }
}
```

a. A es una interfaz con dos métodos *funcionA* y *funcionB*, sabiendo que *funcionA* no es abstracto

b. A es una interfaz con dos métodos *funcionA* y *funcionB*, donde *funcionB* no es abstracto

c. A es una interfaz con un método abstracto *funcionA* y un método no abstracto *funcionB*

d. A es una interfaz con dos métodos abstractos *funcionA* y *funcionB*

Pregunta 8

Finalizado

Considera la siguiente función infix e indica la afirmación correcta

```
enum class Tipo {
```

Pregunta 8

Finalizado

Puntúa como 1,00

Marcar pregunta

Considera la siguiente función infix e indica la afirmación correcta

```
enum class Tipo {
    AGUA, FUEGO, HIERBA
}

data class Pokemon(val tipo: Tipo)

infix fun Pokemon.venceA(otro: Pokemon): Boolean {
    return if(((this.tipo.ordinal+1)%3) == otro.tipo.ordinal){
        true
    } else {
        false
    }
}

fun main(){
    val squirtle = Pokemon(Tipo.AGUA)
    val charmander = Pokemon(Tipo.FUEGO)
    val bulbasaur = Pokemon(Tipo.HIERBA)
}
```

a. Colocar la definición de la clase *Pokemon* y la declaración de la función *venceA* en archivos distintos da lugar a un error de compilación

b. *charmander.venceA(squirtle)* produce un error de compilación porque es una función infix

c. La variable implícita *this* de la función *venceA* hace referencia a una instancia de la clase *Tipo*

d. *squirtle venceA bulbasaur* se evalúa a false

d. *squirtle venceA bulbasaur* se evalúa a false

Pregunta 9
Finalizado
Puntúa como 1,00
Marcar pregunta

Indica cuál de las siguientes líneas de código muestra la longitud de una nullable string y, en caso de que esta sea null, muestra un 0.

a. println(b?.length ?: 0)
b. println(b?.length ?? 0)
c. println(b==null? 0 : b.length)
d. println(b!!.length ?: 0)

Pregunta 10

Finalizado

Puntúa como 1.00

Marcar pregunta

¿Qué hace el siguiente código?

```
foo()()
```

- a. Llama a la función devuelta por la llamada a *foo*
- b. Crea un array bidimensional
- c. Instancia un objeto de tipo *foo()*
- d. No compila

Finalizar revisión