

Paradigmas de la Programación – Tercer Parcial

13 de Junio de 2024

100 1. [10 pt.] Indique los números de línea en los que observa las siguientes propiedades de los lenguajes de scripting, o ninguno si la propiedad no se encuentra en el siguiente fragmento de código:

- expresiones regulares _____
- abstracciones lingüísticas propias del dominio 1,5,6, 11,13,15, 10,12 {VIENEN DE 1,5}
- abstracciones lingüísticas para la comunicación con el sistema operativo 10,12
- no declaración de tipos de variables _____
- otra POCA VERBOSIDAD

```
1 entity player;  
2 float playerSpeed = 100.0;  
3  
4 void main() {  
5     player = spawn();  
6     setmodel(player, "player.mdl");  
7  
8     while (1) {  
9         // Process keyboard input  
10        if (key_down(KLEFTARROW)) {  
11            player.velocity.y = -playerSpeed;  
12        } else if (key_down(KRIGHTARROW)) {  
13            player.velocity.y = playerSpeed;  
14        } else {  
15            player.velocity.y = 0;  
16        }  
17    }  
18 }
```

70 100 2. [10 pt.] Según el siguiente texto:

A synchronized block in Java is synchronized on some object. All synchronized blocks synchronize on the same object and can only have one thread executed inside them at a time. All other threads attempting to enter the synchronized block are blocked until the thread inside the synchronized block exits the block.

Seleccione las respuestas verdaderas:

- ☒ un bloque sincronizado preserva la atomicidad del bloque a nivel semántico.
- en un bloque sincronizado se pueden dar condiciones de carrera.
- ☒ la semántica de un bloque sincronizado se puede escribir mediante locks.
- ☒ la semántica de un bloque sincronizado se puede escribir mediante un buffer productor-consumidor.

IMITAN A LOS LOCKS

100 3. [10 pt.] Dada la siguiente base de conocimiento:

```

1 tiene_pelo(gato).
2 tiene_pelo(perro).
3 vuela(loro).
4 nada(pinguino).
5 tiene_plumas(pinguino).
6
7 tiene_plumas(X) :- not(tiene_pelo(X)).
8 mamifero(X) :- tiene_pelo(X).
9 ave(X) :- vuela(X).
10 ave(X) :- not(pez(X)).
11 ave(X) :- not(mamifero(X)).
12 pez(X) :- nada(X), not(tiene_pelo(X)).

```

Seleccione cuáles de los siguientes subobjetivos se tienen que evaluar (determinar si son ciertos o falsos) para que el motor de inferencia de Prolog determine que `ave(loro)` es cierto.

- ☒ `ave(loro).`
- ☒ `nada(loro).`
 - `tiene_plumas(loro).`
- ☒ `mamifero(loro).`

100 4. [5 pt.] En el siguiente fragmento de código, de qué tipo de vulnerabilidad se está protegiendo este programa con la estrategia de programación defensiva que implementa?

Respuesta: SE ESTÁ PROTEGIENDO DE UNA VULNERABILIDAD EN EL TIPO DE LAS VARIABLES.

[5 pt.] Tache las líneas que considere necesario para convertirlo en un ejemplo de programación ofensiva en lugar de programación defensiva.

```

1 def divide_numbers(dividend, divisor):
2     if divisor == 0:
3         raise ValueError("Cannot divide by zero.")
4     result = dividend / divisor
5     return result
6
7 def get_user_input():
8     try:
9         dividend = int(input("Enter the dividend: "))
10        divisor = int(input("Enter the divisor: "))
11        result = divide_numbers(dividend, divisor)
12    except ValueError as e:
13        print("Invalid input:", e)
14
15 get_user_input()

```

50 5. [10 pt.] El siguiente texto:

Immutable variables (val in Scala) are inherently thread-safe because their values cannot be changed after initialization. This makes them a preferred choice in concurrent programming.

- se refiere a seguridad de tipos, del tipo que se puede solucionar mediante canonicalización.
- ☒ se refiere a seguridad con respecto a la sección crítica.
- se refiere a seguridad en memoria.
- ☒ describe una propiedad de las componentes declarativas que ofrece mayor seguridad en programación concurrente.