**Conceptos y Paradigmas de Lenguajes de Programación - Seg Parcial 1ra Fecha. T1.- 13/06/2025**

Realice el parcial con lapicera, de otra forma se desaprobará el/los ejercicio/s.

Se considera presentismo cuando se realiza completamente un ejercicio.

| Legajo: 230010 | | | | | | Corrigió: Laura |
|---|---|---|---|---|---|---|
| Apellido y Nombres: Guaymas Matas Julián | | | | | | |
| 1 | a B | b B | | | | |
| 2 | a — | b B | | | | |
| 3 | a B | b B- | | | | |
| 4 | a B | b B | c B | d B | e B | |
| | | | | | Resultado Final | Aprobado |

**Ejercicio 1** (30p)

Realice la pila de ejecución para el siguiente código:

a) por cadena estática b) por cadena dinámica **Nota:** La forma de evaluación de este lenguaje es de izquierda a derecha.

```
Program Main;
Var h, m, n, o:integer;
a, b: array[1..3] of integer;

Procedure Uno(val-res m: integer; nombre n: integer);
 var h:integer;
 begin
    h=5;
    m:= f + 2;
    n:= n + 3;
    o:=1;
    h:= n + 10;
end;

Function F: integer;
 Var n:integer;
 Begin
   n:=1+h;
   if(m < 3 and h > 0) then
   begin
          b(1):= b(1) + 1;
          m:= m + 1;
          n:=2
end;

 if (n< 3) then
 begin
    a(n):=a(n)+b(1)-2;
    a(n)=a(n) * 2;
    n:=1;
end
else
begin
      n:=3;
end
return b(n);
end //de la función F

begin //del Main
      m:= 1;  n:= 1; h=1;
      for o:=1 to 3 do begin
              a(o):= o+1;
              b(o):= o * 2;
      end;
      Uno(a(o), b(o));
      for o:=1 to 3 do  write (a(o), b(o));
end.
```

**Ejercicio 2**

a) (10pts) Clasifique las siguientes estructuras de datos de acuerdo a lo visto en la práctica.Justifique en cada caso:

i) Python

```
class Moto:
    def __init__(self):
        self._patente = None
        self._anio = None
        self._modelo = None
        self._kms = []
    @property
    def anio(self):
        return self._anio
 @anio.setter
 def anio(self, valor):
   self._anio = valor

 def get_kms(self, km):
   return self._kms[km]
```

ii) Pascal

```
type
  PVertice = ^Vertice;
  PAdyacente = ^Adyacente;

  // Lista de adyacencia (punteros a otros vértices)
  Adyacente = record
    destino: PVertice;
    siguiente: PAdyacente;
  end;

  // Nodo del grafo
  Vertice = record
    id: Integer;
    adyacentes: PAdyacente;
    siguiente: PVertice;
  end;
```

Ejercicio 2 b) (10 pts) Responda si las siguientes afirmaciones son V o F. Justifique en cada caso

*i)* No existen lenguajes compilados que sean débilmente tipados, básicamente por poder chequear estáticamente el tipo

ii) En java implementar una clase, asegura de por sí que ya se posee encapsulamiento y ocultamiento.

**Ejercicio 3**

a) (15 pts) Dado el siguiente código en Java, establezca cuáles de las opciones indicadas más abajo son válidas como camino de ejecución. Justifique su selección con una breve descripción del flujo de ejecución, caso contrario no se considerará válida la respuesta)

```
1  public class ParcialExcepciones {
2      public static void main(String[] args) {
4              try{
5                      for (int i = 0; i < 5; i++) {
6                              if(i==1){
7                                      System.out.println(Integer.toString(i));
8                                      relanzador(i);
9                              }
10                             else{
11
12                                     if(i==2 || i==3){
13                                             switch(i){
14                                                     case 2:
15                                                             System.out.println(Integer.toString(i));
16                                                             relanzador(i);
17                                                             break;
18                                                     case 3:
19                                                             System.out.println(Integer.toString(i));
20                                                             relanzador(i);
21                                                             break;
22                                             }
23                                     }
24                                     else{
25                                             System.out.println(Integer.toString(i));
26                                             relanzador(i);
27                                     }
28                             }
29                     }
30             }catch(ThirdException | FourthException e){
31                     System.out.println(e.getMessage());
32             }
34     }
35     static void relanzador(int i) throws ThirdException, FourthException {
36         try {
37                 try {
38                         switch(i){
39                           case 1:
40                                   throw new FirstException(Integer.toString(i));
41                                   break;
42                            case 2::
43                                   throw new SecondException(Integer.toString(i));
44                                   break;
45                           case 3:
46                                   throw new ThirdException(Integer.toString(i));
47                                   break;
48                           default:
49                                   throw new FourthException(Integer.toString(i));
50                                   break;
51                         }
52                 }catch (SecondException e) {
53                         ThirdException e1=new ThirdException(Integer.toString(i));
54                         throw e1;
55                 }
56         }catch (FirstException e) {
57                 FourthException e1=new FourthException(Integer.toString(i));
58                 throw e1;
59                 }
60         }
```

i) Se imprime en pantalla "0", luego "0" y termina ✓
ii) Se imprime en pantalla "0", "0", "1", "1", "2", "2", "3", "3", "4", "4" y luego termina
iii) Se imprime en pantalla "0", "0", "1", "1", "2", "2", "3", "3" y luego termina
iv) Ninguna de las anteriores

b) (15 pts) Indique si iniciando el for en 1 en vez de 0, el resultado es el mismo. Indique qué se imprime en ese caso

**Ejercicio 4**

(20pts). Marcar si son verdaderas o falsas las siguientes afirmaciones. Acompañar la respuesta con una justificación, caso contrario, NO se tomarán cómo válidas

a. La unión y la unión discriminada son igualmente seguras V F ✓
b. Java tiene un equivalente a la sentencia YIELD de python V F ✓
c. El switch en Java no requiere la cláusula default de manera obligatoria V ✓ F
d. En PL/1 si se genera una excepción, se ejecuta el manejador correspondiente y luego el control se pasa inmediatamente a la rutina padre de aquella donde se generó la excepción V F ✓
e. La sentencia finally de python en el manejo de excepciones se ejecuta siempre y cuando no se haya levantado una nueva excepción dentro de un except V F ✓

13 06 25

GUAYMAS MATIAS Legajo: 23061/0 Hoja 1/2

1) a)

*1 ***Reg Activ Main
PR
LE
LD
h = 1 ✓
m = ~~1~~ - 2 ✓
n = 1 ✓
0 = 1..~~3~~ - ~~1~~ - 1..3 ✓
a(1) = 2 ✓
a(2) = ~~3~~ - ~~4~~ - 8 ✓
a(3) = ~~4~~ - 5 ✓
b(1) = ~~3~~ - 3 ✓
b(2) = 4 ✓
b(3) = ~~6~~ - 9 ✓
Procedure uno
Function F
VR ✓

*2 ***Reg Activ Uno
PR
LE(*1)
LD(*1)
m = ~~4~~ - 5 ✓
n ↑ b(0) ✓
h = ~~5~~ - 13 ✓
VR 3 ✓

*3 ***Reg Activ F
PR
LE(*2)
LD(*2)
n = ~~2~~ - ~~2~~ - 1 ✓
VR

Imprime a(2)

a(1) 2 3 b(1)
8 4 b(2)
5 9 b(3)
a(3)

B

b) *1 ***Reg Activ Main

PR

LE

LD

n = 1 ✓

m = 1 ✓

n = 1 ✓

o = 1..3 - X - 1..3 ✓

a(1) = 2 ✓

a(2) = 3 ✓

a(3) = X - 8 ✓

b(1) = 2 ✓

b(2) = 4 ✓

b(3) = b - 9 ✓

Procedure Uno

Function F

PR ✓

*2 ***Reg Activ Uno

PR

LE (*1)

LD (*1)

m = X - 8 ✓

n ↑ b(0) ✓

n = 5 - 12 ✓

VR 6 ✓

*3 ***Reg Activ F

PR

LE (*2)

LD (*2)

n = b - 3 ✓

VR

B

Imprime 2 ✓ 2 ✓

a(1) b(1)

3 ✓ 4 ✓

a(2) b(2)

8 ✓ 9 ✓

a(3) b(3)

2) i) Falso. Que un lenguaje sea compilado no quiere decir que aplique al sistema de tipos, en este caso débilmente tipado. Un contraejemplo es C, que es compilado y débilmente tipado, por ejemplo puedo sumar o restar un valor a una variable de tipo puntero. ✓

ii) Falso. En Java cuando implementamos una clase podemos definir el estado interno de la clase como público y los conceptos de ocultamiento y encapsulamiento se estarían "violando" (se dice que rompe con el concepto de encapsulamiento de una clase). Podemos definir las variables de instancia de la clase como public, y en caso de no declararse ningún formato de privacidad, los mismos también son públicos. ✓

3) a) La opción correcta es la opción i). El flujo de ejecución es este.

En la línea 4 tenemos un try que adentro del bloque del mismo hace un FOR desde 0 (empieza inicializado en cero) hasta el índice menor estricto que 5. Es de importancia esta declaración ya que si se lanza alguna excepción dentro del flujo del FOR, este dejará de ejecutarse y no avanzará a las iteraciones restantes (y finalizará la ejecución del bloque try-catch por completo).

como "i" vale cero, las expresiones de las líneas 6 y 11 resultan falsas y se ejecuta la lógica asociada al else de la línea 23. En este se imprime cero y se invoca al método relanzador con el parámetro "i" que vale cero.

El método relanzador es capaz de propagar dinámicamente las excepciones thirdException y fourthException gracias a la sentencia: throws ThirdException, fourthException. El método tiene 2 try-catch y cuando evalúa los posibles caminos del switch (línea 37) todos resultan falsos porque "i" vale cero, ejecutando la sentencia 48 y 49 porque está declarado un default. (se ejecuta cuando los caminos anteriores declarados resultaron falsos) y lanza una excepción de tipo FourthException. Dentro de ese try-catch no hay un manejador declarado del mismo tipo de la excepción lanzada, por lo que se propaga al try-catch externo, que tampoco tiene el manejador de la excepción necesario, propagándose dinámicamente al cuasin. Por último existe un manejador de tipo FourthException (línea 29) en el método main que controla la anomalía e imprime cero, finalizando la ejecución del programa. ✓

A) Con mensaje "0" y finaliza la ejecución de la unidad.

b) El resultado no será el mismo ya que lanzará la excepción FirstException en la línea 39 porque "i" es igual a 1, como no tiene un manejador asociado a dicho tipo se propaga estáticamente, en el catch externo la captura y lanza una excepción FourthException que se propaga dinámicamente al cuasin, finalizando la ejecución de la unidad. Se maneja la excepción en la línea 29 y finaliza la ejecución del main.

Imprime: "1", "1" y termina.

4) a) Falso. La unión y la unión discriminada no son igualmente seguras, ya que la unión discriminada ofrece la opción de saber si de la unión los discriminantes realmente tienen un valor asignado para no acceder a datos inválidos, aunque es opcional la unión no cuenta con dicha verificación. ✓

b) Falso. Java no tiene un equivalente a la sentencia YIELD de Python, se suele comparar el return de Java con la sentencia YIELD pero "return" devuelve un único valor mientras que "yield" una lista de valores (no es posible implementarlo en Java), otras diferencias son que return finaliza la ejecución de la función y ocupa más memoria que si usáramos "yield", que es más eficiente en cuanto a memoria ocupada y puede frenar la ejecución de la función y continuar a partir de dicho punto. ✓

→ pero se podía llegar a simular con excepciones.

c) Verdadero. Es opcional lo sentencia default en un case y un ejemplo de implementación es el código mostrado de Java del ejercicio 3 en las líneas 12 a 21. ✓

d) Falso. En PL/1 si se genera una excepción, se ejecuta el manejador correspondiente y luego se continúa con la ejecución a partir de la siguiente línea donde se generó la excepción, debido a que PL/1 sigue el modelo de terminación por reasunción. ✓

e) Falso. La sentencia finally de Python en el manejo de excepciones se ejecuta siempre haya ocurrido o no una excepción. Para lo mencionado en el inciso usamos la sentencia else, y no finally.