Ejemplo de Examen Primer Parcial LTP

EVALUACIÓN: $\text{NOTA} = \frac{\text{Aciertos} - \frac{\text{Fallos}}{3}}{\text{Número de preguntas}} \times 10$

1. ¿Cuál de las siguientes afirmaciones es **FALSA**?

   - Java permite definir tanto clases como métodos genéricos.
   - En Java, una clase genérica puede tener varios parámetros genéricos.
   - Las clases genéricas en Java deben ser necesariamente abstractas.
   - En las clases genéricas de Java es posible restringir las clases a las que se puede instanciar un parámetro genérico.

2. En un lenguaje con tipificación explícita...

   - es necesario declarar el tipo de todas las variables.
   - todas las expresiones tienen un tipo asociado.
   - todos los tipos se determinan en tiempo de ejecución (tipificación estática).
   - todos los tipos se determinan en tiempo de compilación (tipificación dinámica).

3. Indica cuál de las siguientes afirmaciones es **CIERTA** sobre la herencia en Java:

   - Una clase *A* puede extender dos clases diferentes.
   - Una clase hija puede añadir métodos pero no atributos.
   - La clase *Object* está en lo más alto de cualquier jerarquía de clases definida en el programa.
   - La siguiente cabecera define la creación de la clase `Alumno` como clase hija de la clase `Persona`:

     `public class Persona extends Alumno {...}`

4. Indica a qué tipo de polimorfismo corresponde la siguiente definición:

   ... hace referencia a funciones u operadores que pueden aplicarse a argumentos de tipos diferentes y que se comportan de forma diferente dependiendo del tipo de los argumentos a los que se aplica.

   - Herencia.
   - Sobrecarga.
   - Genericidad.
   - Abstracción.

5. Al ejecutar el siguiente fragmento de código

```
1 void inc (int V) {
2    V++;
3 }
4. ..
5 int A = 10;
6 inc(A);
7 print A;
```

¿qué valor se imprimirá en la última instrucción dependiendo del tipo de paso de parámetros?

- 11, tanto si es por valor como por referencia.
- 10, tanto si es por valor como por referencia.
- 10 si es por valor, y 11 si es por referencia.
- 11 si es por valor y 10 si es por referencia.

6. Dado el siguiente fragmento de código:

```
String n; %% variable global
void foo() {
   String n;
   n="foo";
}
void proc() {
   foo();
   n="proc";
}
begin %% cuerpo principal del programa
  n = "main"
  proc();
  print(n);
end
```

¿qué imprime la última instrucción dependiendo del tipo de alcance de las variables?

- Alcance estático: `proc`; alcance dinámico: `proc`.
- Alcance estático: `proc`; alcance dinámico: `foo`.
- Alcance estático: `main`; alcance dinámico: `foo`.
- Alcance estático: `main`; alcance dinámico: `proc`.

7. La reflexión permite (indica la **FALSA**):

   - Obtener y mostrar, durante la ejecución del programa, el nombre de todas las instancias de la clase que se han creado en ejecución.
   - Leer de teclado un string y usarlo para crear un objeto con ese nombre.
   - Definir métodos genéricos cuya ejecución depende de la instanciación de los parámetros.
   - Leer de teclado un string y usarlo para invocar a un método con ese nombre.

8. Indica cuál de las siguientes afirmaciones sobre los paradigmas de programación es **FALSA**:

   - El paradigma imperativo no se ve afectado por efectos laterales debido a la transparencia referencial.
   - El paradigma declarativo incluye al funcional y al lógico.
   - Un lenguaje de programación puede pertenecer a varios paradigmas.
   - La programación conducida por eventos es un paradigma basado en interacción.

9. Indica cuál de las siguientes afirmaciones sobre los lenguajes de programación es **CIERTA**:

   - Los lenguajes de script no pueden ser orientados a objetos.
   - Todos los lenguajes declarativos son perezosos.
   - Ningún lenguaje es concurrente y declarativo simultáneamente.
   - En programación imperativa y programación funcional una llamada a ejecución no contiene variables.

10. La función principal del dispatcher en la programación por eventos consiste en:

- determinar el tipo de evento y seleccionar el manejador apropiado que lo trata.
- seleccionar el evento que se trata por el manejador que está en ejecución en un momento dado.
- ejecutar, cada vez que recibe un evento, todos los manejadores hasta que uno de ellos trate el evento.
- gestionar la concurrencia entre dos procesos que tratan el mismo evento.

11. En la compilación de un programa, la optimización del código intermedio se realiza durante la fase de:

- análisis sintáctico.
- síntesis.
- enlace (*linking*).
- análisis semántico.

12. La semántica dinámica de un programa:

- Se obtiene a partir del texto del programa durante la fase de análisis semántico.
- Se refiere a los posibles cambios en el texto del programa al evolucionar en el tiempo para adaptarse a nuevas especificaciones del usuario final.
- Es un concepto inexistente; en lenguajes de programación solo hablamos de semántica *estática*.
- Permite razonar acerca de la ejecución del mismo. Según la descripción semántica utilizada es posible desarrollar un intérprete, verificar propiedades del programa formuladas mediante asertos, etc.

13. Determina cuál es la configuración que debe aparecer en lugar del (*) en la siguiente transición:
`<if X<4 then X:=X-1 else X:=X+1,{X→3}> -> (*)`
empleando la semántica operacional de paso pequeño (*small-step*):

- `<skip,{X→2}>`
- `<skip,{X→4}>`
- `<X:=X-1,{X→3}>`
- `<X:=X+1,{X→3}>`

14. Escribe la premisa de la siguiente regla de transición:

$$\frac{?}{(y := x{+}{+},\ s) \Downarrow s'}$$

para definir la semántica operacional de paso grande (*big-step*) de una instrucción de *"asignación con post-incremento"* $(y := x{+}{+})$ que incrementa en uno el valor de la variable x **después** de realizar la asignación $y := x$:

- $(x + 1,\ s) \Rightarrow s'$
- $(y := x,\ s) \Downarrow s' \wedge (x := x + 1,\ s) \Rightarrow s'$
- $(x := y + 1,\ s) \Downarrow s'$
- $(y := x;\ x := x + 1,\ s) \Downarrow s'$

15. Dada la siguiente ejecución con la semántica operacional de paso grande (big-step):

$$(\text{while } X > 0 \text{ do } X := X - 1,\ \{X \mapsto 2\}) \Downarrow (*)$$

¿cuál es valor de $(*)$?

- $\{X \mapsto 0\}$.
- $\{X \mapsto 1\}$.
- $\langle \text{skip}, \{X \mapsto 0\}\rangle$.
- $\langle \text{skip}, \{X \mapsto 1\}\rangle$.

16. Dado el siguiente fragmento de programa con sus pre- y post-condiciones:

```
{P}
X := X-1;
Y := X;
{Q} = { Y > 0 }
```

¿Cuál de las siguientes expresiones para P nos permite verificar que el código es correcto usando la semántica axiomática?

- `X = 0`.
- `X = 1`.
- `X > 0`.
- `X = 2`.

17. En una terna de Hoare $\{P\}\ S\ \{Q\}$:

- P y Q son programas y S es un estado de la máquina.
- P se denomina *precondición* y Q se denomina postcondición y siempre se cumple $P \Rightarrow Q$.
- La corrección de la terna se garantiza si $P \Rightarrow pmd(S, Q)$.
- La corrección de la terna se garantiza si $pmd(S, P) \Rightarrow Q$.

18. ¿Cuál es la configuración que falta (en la posición *) para completar la siguiente evaluación usando la semántica operacional de paso pequeño (small-step)?

```
<if X>Y then Y:=Y+X else Y:=0, {X->42, Y->0}>
  <X>Y, {X->42, Y->0}> => true
    <X, {X->42, Y->0}> => 42
    <Y, {X->42, Y->0}> => 0
-> <Y:=Y+X, {X->42, Y->0}>
    <Y+X, {X->42, Y->0}> => 42
      <Y, {X->42, Y->0}> => 0
      <X, {X->42, Y->0}> => 42
-> (*)
```

- `<skip, {X->42, Y->0}>`
- `<Y:=0, {X->42, Y->0}>`
- `<skip, {X->42, Y->42}>`
- `<if X>Y then Y:=Y+X else Y:=0, {X->42, Y->0}>`

19. Dados los siguientes programas P_1 y P_2.

P_1:

```
x:=0;
if x>0 then y:=10
       else y:=5
```

P_2:

```
x:=0;
x:=x+5;
y:=x
```

podemos decir que:

- P_1 y P_2 son equivalentes, independientemente de la semántica que se siga.
- P_1 y P_2 son equivalentes con respecto a la semántica big-step.
- P_1 y P_2 son equivalentes con respecto a la semántica small-step.
- P_1 y P_2 no son equivalentes, independientemente de que consideremos la semántica big-step o small-step.

20. ¿Cuál de las siguientes afirmaciones sobre los procesos de traducción e interpretación en la implementación de los lenguajes de programación es [FALSA]?

- Los intérpretes son más adecuados para la fase de desarrollo, mientras que los compiladores son más adecuados para la fase de explotación.
- Los compiladores son más adecuados para la fase de desarrollo, mientras que los intérpretes son más adecuados para la fase de explotación.
- En la práctica no se suele usar la traducción pura ni la interpretación pura, sino una implementación mixta.
- Java es un lenguaje con implementación mixta.

Hoja de Respuestas

Primer Parcial (LTP) - TEORÍA - 12 Noviembre de 2072

TIPO A

Marcar la casilla correspondiente a vuestra respuesta para cada una de las preguntas anteriores

Alumno:

1. [A] [B] [C] [D]
2. [A] [B] [C] [D]
3. [A] [B] [C] [D]
4. [A] [B] [C] [D]
5. [A] [B] [C] [D]
6. [A] [B] [C] [D]
7. [A] [B] [C] [D]
8. [A] [B] [C] [D]
9. [A] [B] [C] [D]
10. [A] [B] [C] [D]
11. [A] [B] [C] [D]
12. [A] [B] [C] [D]
13. [A] [B] [C] [D]
14. [A] [B] [C] [D]
15. [A] [B] [C] [D]
16. [A] [B] [C] [D]
17. [A] [B] [C] [D]
18. [A] [B] [C] [D]
19. [A] [B] [C] [D]
20. [A] [B] [C] [D]

Hoja de Respuestas (Copia para el Alumno)
Primer Parcial (LTP) TEORÍA- 12 Noviembre de 2072

TIPO A

Sólo es posible sacar esta hoja del examen

Alumno:

1. [A] [B] [C] [D]
2. [A] [B] [C] [D]
3. [A] [B] [C] [D]
4. [A] [B] [C] [D]
5. [A] [B] [C] [D]
6. [A] [B] [C] [D]
7. [A] [B] [C] [D]
8. [A] [B] [C] [D]
9. [A] [B] [C] [D]
10. [A] [B] [C] [D]
11. [A] [B] [C] [D]
12. [A] [B] [C] [D]
13. [A] [B] [C] [D]
14. [A] [B] [C] [D]
15. [A] [B] [C] [D]
16. [A] [B] [C] [D]
17. [A] [B] [C] [D]
18. [A] [B] [C] [D]
19. [A] [B] [C] [D]
20. [A] [B] [C] [D]