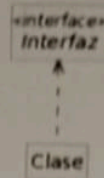


Nombre: Padrón: / 10

75.07 - ALGORITMOS Y PROGRAMACIÓN III - PRIMER PARCIAL - Jueves 20/10/2022

1) Dado el siguiente diagrama de clases UML:

... / 2



- indique qué relación representa el mismo;
- explique cómo dicha relación podría aparecer en un programa.

2) Dado el siguiente programa:

... / 3

```
package ejercicio2;

public class Suma {

    public void sumar1() {
        Main.pantalla.print("Ingrese n: ");
        int numero = Main.teclado.nextInt();
        int resultado = numero + 1;
        Main.pantalla.print("El sucesor de " + numero + " es: ");
        Main.pantalla.println(resultado);
    }
}
```

```
package ejercicio2;

public class Main {

    public static final java.util.Scanner teclado = new java.util.Scanner(System.in);
    public static final java.io.PrintStream pantalla = new java.io.PrintStream(System.out);

    public static void main(String[] args) {
        new Suma().sumar1();
        new Producto().multiplicarPor2();
    }
}
```

```
package ejercicio2;

public class Producto {

    public void multiplicarPor2() {
        Main.pantalla.print("Ingrese n: ");
        int numero = Main.teclado.nextInt();
        int resultado = numero * 2;
        Main.pantalla.print("El doble de " + numero + " es: ");
        Main.pantalla.println(resultado);
    }
}
```

- describa su funcionamiento (en palabras o mediante un diagrama de secuencia) y dé un ejemplo;
- indique si viola algún principio de diseño y, en caso afirmativo, indique cuál(es) y explique por qué lo hace;
- justifique si la clase es candidata a ser refactorizada mediante algún patrón de diseño y, en caso afirmativo, indique mediante cuál(es) y explique cómo lo haría.

3) Dada la siguiente clase:

... / 3

```
package ejercicio3;

public class Sumador {
    private static Sumador sumador = null;
    private Sumador() {
    }
    public double sumar(double a, double b) {
        return a + b;
    }
    public static Sumador getInstance() {
        if (sumador == null) {
            sumador = new Sumador();
        }
        return sumador;
    }
}
```

Salida:

9.8

- explique cómo se la utilizaría desde otra clase para obtener la salida mostrada;
- indique si viola algún principio de diseño y, en caso afirmativo, indique cuál(es) y explique por qué lo hace;
- justifique si la clase se basa en algún patrón de diseño y, en caso afirmativo, indique en cuál(es).

4) Dado el siguiente método:

... / 2

```
public byte nthFibonacci(int n) {
    if (n < 0 || n > 11)
        throw new ArithmeticException("Valor fuera del rango permitido.");
    else if (n < 2)
        return (byte) n;
    else {
        byte f, a = 0, b = 1;
        do {
            f = (byte) (a + b);
            a = b; b = f;
            n--;
        } while (n > 1);
        return f;
    }
}
```

① a=0 b=1 ③ a=5 b=8
② a=1 b=1 ④ a=8 b=13
⑤ a=2 b=3 ⑥ a=13 b=21
⑦ a=3 b=5 ⑧ a=21 b=34
⑨ a=34 b=55

que retorna un valor del tipo **byte**, es decir un entero del intervalo [-128 .. 127], el cual representa el n -ésimo término de la sucesión de Fibonacci, definida como $F_0 = 0$; $F_1 = 1$ y por último, para $n \geq 1$, $F_n = F_{n-1} + F_{n-2}$.

- escriba la menor cantidad de pruebas unitarias que deberían considerarse para cubrir todas las clases de equivalencia* posibles;
- indique qué debería cambiarse en el método y en sus pruebas si, en lugar de **byte**, se usara **short**, es decir, un entero del intervalo [-32768 .. 32767].

(*) Clase de equivalencia: conjunto de datos de entrada donde el comportamiento del software es el mismo para todos los elementos del conjunto (por lo tanto, en una clase de equivalencia, cualquier elemento de la clase es representativo del resto del conjunto).