Programación

EXAMEN TEMA 4 – PROGRAMACIÓN MODULAR

(15/01/2025)

1. (2,5p) Teniendo en cuenta el siguiente menú de opciones a ejecutar,

```
***** BIENVENIDO A LA CALCULADORA RÁPIDA *****
   [C] - Iniciar
   [X] - Salir
```

```
CALCULADORA INICIADA...
Elige la operación a realizar:
   [+] Sumar
   [-] Restar
   [x] Multiplicar
   [/] Dividir
   [X] - Volver a pantalla principal
```

   a) (1p) Indica cuántos métodos crearías dentro de un programa *Java* para darle funcionalidad y explica por qué.

   b) (1,5p) Escribe la cabecera que tendrían cada uno de los métodos (incluidos los parámetros) y lo que devolverían (en caso de que devuelvan algo).

2. (1p) Identifica las variables globales y locales sobre los métodos de la siguiente clase:

```
public class Ejemplo {
     static int x;
     static int cuadrado(int z){
         return z*z;
     }
     public static void main (String args[]){
         int p = 5;
         z = 3;
         x = cuadrado(p);
         System.out.print(x);
     }
}
```

   - ¿Qué pasará en el programa?

3. (1p) Enumera las etiquetas típicas de *Javadoc* e indica con tus palabras la utilidad de documentar una aplicación usando esta herramienta.

4. (1p) ¿En qué consiste la sobrecarga de métodos en *Java*? Pon ejemplos.

5. (2p) Dada la siguiente función recursiva,

```java
public static int funcion_recursiva(int n) {
    if (n <= 0) {
        return 0;
    } else {
        return -n + funcion_recursiva(--n);
    }
}
```

a) (0,5p) Marca sobre el código dado el caso base (salida no recursiva) y el caso general (salida recursiva).

b) (1,5p) ¿Qué mostraría el siguiente programa que llama a la función recursiva anterior?

```java
public static void main(String[] args) {
        System.out.println("El resultado es: " + funcion_recursiva(4));
}
```

6. (1p) Crea el código para el método ***comprobarDivisor()*** teniendo en cuenta que se van a lanzar las siguientes pruebas contra él para verificar su funcionamiento en *JUnit*:

```java
@Test
void comprobarDivisor() {

    assertAll(() -> assertTrue(Dividir.comprobarDivisor(2)),
              () -> assertFalse(Dividir.comprobarDivisor(0)),
              () -> assertTrue(Dividir.comprobarDivisor(-7)));

}
```

7. (1,5p) Intenta replicar el comportamiento de la pila de llamadas del siguiente fragmento de código y contesta a las preguntas:

a) ¿Qué mostrará el método principal ***main***?

b) ¿Qué devolverá el método ***proceso2()*** en cada una de sus llamadas?

```java
public class Examen {

    static int proceso1(int n1) {
        int p1 = 0;
        for (int i = 0; i < n1; i += 2) {
            p1 += proceso2(i);
        }
        return p1;
    }

    static int proceso2(int n2) {
        int p2 = 0;
        for (int i = 0; i < n2; i++) {
            p2 += n2;
        }
        return p2;
    }

    public static void main(String args[]) {
        System.out.println(proceso1(5));
    }

}
```