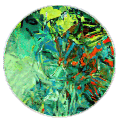

C.F.G.S. Desarrollo de Aplicaciones Web/Multiplataforma

MÓDULO: Programación

IES CAMAS Curso 2025/26

Nombre y apellidos: _____ Fecha: __/__/__

(2,5 puntos) Ejercicio 1:Crea un programa en Java llamado **PromedioNotas.java** que pida al usuario tres notas (de tipo `double`). Es obligatorio usar funciones para:

- El **promedio** de las tres.
- Si el alumno está **aprobado** (nota media ≥ 5) o **suspenso** (nota media < 5).

```
public class PromedioNotas {
   // Función que calcula el promedio de tres notas
   public static double calcularPromedio(double n1, double n2, double n3)
{
        return (n1 + n2 + n3) / 3.0;
   }
   // Función que determina si el alumno está aprobado
   public static boolean estaAprobado(double promedio) {
        return promedio >= 5.0;
   }
   public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Introduce la primera nota: ");
        double n1 = sc.nextDouble();
        System.out.print("Introduce la segunda nota: ");
        double n2 = sc.nextDouble();
        System.out.print("Introduce la tercera nota: ");
        double n3 = sc.nextDouble();
        double promedio = calcularPromedio(n1, n2, n3);
        System.out.printf("El promedio es: %.2f%n", promedio);
        if (estaAprobado(promedio)) {
            System.out.println("El alumno está aprobado.");
        } else {
            System.out.println("El alumno está suspenso.");
        }
        sc.close();
   }
}
```

(2,5 puntos) **Escribe una función a la que se pase como parámetros de entrada una cantidad de días, horas y minutos. La función calculará y devolverá el número de segundos que existen en los datos de entrada.**

```
/**
    * Convierte una cantidad de tiempo expresada en días, horas y minutos
    * al número total de segundos equivalentes.
    * @param dias    número de días que se desean convertir.
    * @param horas   número de horas adicionales que se desean convertir.
    * @param minutos número de minutos adicionales que se desean convertir.
    * @return el número total de segundos correspondientes al tiempo
indicado.
    */
   public static int convertirASegundos(int dias, int horas, int minutos)
{
        int segundos = dias * 24 * 3600 + horas * 3600 + minutos * 60;
        return segundos;
   }
```

(5 puntos) Ejercicio 3: Selecciona la respuesta correcta

4.1. Los parámetros en la llamada a una función en Java pueden ser opcionales si: a) Todos los parámetros son del mismo tipo. b) Todos los parámetros son de distinto tipo. **c) Nunca pueden ser opcionales.** d) Siempre que el tipo devuelto no sea void.	**4.2. Una variable local (declarada dentro de una función) puede usarse:** a) En cualquier lugar del código. b) Solo dentro de main(). **c) Solo en la función donde se ha declarado.** d) Ninguna de las opciones anteriores es correcta.
4.3. El tipo devuelto de todas las funciones definidas en nuestro programa tiene siempre: a) int. b) double. c) void. **d) Ninguna de las opciones anteriores es correcta**	**4.4. ¿Qué instrucción permite a una función devolver un valor?** a) value. **b) return.** c) static. d) function.
4.5. La forma de distinguir entre dos o más funciones sobrecargadas es: a) Mediante su nombre. b) Mediante el tipo devuelto. c) Mediante el nombre de sus parámetros. **d) Mediante su lista de parámetros: número o tipos.**	**4.6. ¿Cuál es la definición de una función recursiva?** **a) Es aquella que se invoca desde dentro de su propio bloque de instrucciones.** b) Es aquella cuyo nombre permite la sobrecarga y además realiza alguna comparación mediante if. c) Es aquella cuyo bloque de instrucciones utiliza alguna sentencia if (lo que llamamos caso base). d) Es aquella que genera un bucle infinito.
4.7. El paso de parámetros a una función en Java es siempre: **a) Un paso de parámetros por copia.** b) Un paso de parámetros por desplazamiento. c) Un paso de parámetros recursivo. d) Un paso de parámetros funcional.	**4.8. En el caso de que una función devuelva un valor, ¿cuál es la recomendación con respecto a la instrucción return?** a) Utilizar tantos como hagan falta. b) Emplear tantos como hagan falta, pero siempre que se encuentren en bloques de instrucciones distintas. c**) Usar solo uno.** d) Utilizar solo uno, que será siempre la primera instrucción de la función.
4.9. ¿Cuáles de las siguientes operaciones se pueden implementar fácilmente mediante funciones recursivas? a) $a^n = a \times a^{n-1}$. b) esPar(n) = esImpar(n − 1) y esImpar(n) = esPar(n − 1). c) suma(a, b) = suma(a + 1, b − 1). **d) Todas las respuestas anteriores son correctas.**	**4.10. En los identificadores de las funciones, al igual que en los de las variables, se recomienda utilizar la siguiente nomenclatura:** a) suma_notas_alumnos(). b) sumanotasalumnos(). c) SumaNotasAlumnos(). **d) sumaNotasAlumnos().**

4.1.

✅ **c) Nunca pueden ser opcionales.**
➡️ En Java todos los parámetros deben pasarse siempre. No existen parámetros opcionales como en otros lenguajes.

4.2.

✅ **c) Solo en la función donde se ha declarado.**
➡️ Una variable local solo existe dentro del método donde se declara. Fuera de él no es accesible.

4.3.

✅ **d) Ninguna de las opciones anteriores es correcta.**
➡️ Una función puede devolver cualquier tipo de dato o nada (`void`). No hay un tipo fijo.

4.4.

✅ **b) return.**
➡️ La instrucción `return` permite devolver un valor desde una función y termina su ejecución.

4.5.

✅ **d) Mediante su lista de parámetros: número o tipos.**
➡️ La sobrecarga de métodos se distingue por el número o tipo de los parámetros, no por el nombre ni el tipo devuelto.

4.6.

✅ **a) Es aquella que se invoca desde dentro de su propio bloque de instrucciones.**
➡️ Una función es recursiva cuando se llama a sí misma dentro de su cuerpo.

4.7.

✅ **a) Un paso de parámetros por copia.**
➡️ En Java los parámetros se pasan siempre por valor (se copia la referencia en caso de objetos).

4.8.

✅ **c) Usar solo uno.**
➡️ Se recomienda usar un único `return` para hacer el código más claro y legible.

4.9.

✅ **d) Todas las respuestas anteriores son correctas.**
➡️ Todas las expresiones dadas pueden resolverse mediante recursividad.

```
public static int potencia(int a, int n) {
    // Caso base: cualquier número elevado a 0 es 1
    if (n == 0)
        return 1;
    // Caso recursivo
    return a * potencia(a, n - 1);
}
```

```
public static int suma(int a, int b) {
    // Caso base: si b es 0, el resultado es a
    if (b == 0)
        return a;
    // Caso recursivo: incrementar a y reducir b
    return suma(a + 1, b - 1);
}
```

```
public static boolean esPar(int n) {
    if (n == 0)
        return true;          // 0 es par
    else
        return esImpar(n - 1);
}

public static boolean esImpar(int n) {
    if (n == 0)
        return false;         // 0 no es impar
    else
        return esPar(n - 1);
}
```

Entre ambas funciones, se produce una **recursión mutua**.

4.10.

✅ **d) sumaNotasAlumnos().**
➡️ La convención en Java es *camelCase*: la primera palabra en minúscula y las siguientes con mayúscula inicial.