## Programación

## EXAMEN RECUPERACIÓN – TEMAS 1-7

1. (1,25p) Expresa con un diagrama de flujo el algoritmo para dar solución al siguiente problema: programa que recibe dos números y dice cuál es el mayor, o si son iguales.

2. (1p) Tienes un *ArrayList<Integer>* con los siguientes valores: [2, 4, 6, 8, 10].
   a) Utiliza un *Iterator* para eliminar los números menores a 8 mientras recorres la lista.
   b) ¿Por qué no se puede usar un *for-each* para esto sin lanzar excepción?

3. (1p) Tenemos las clases *Instrumento* y *Guitarra*. Ambas implementan el método *tocar():*

```
public class Instrumento {
    public void tocar() {
        System.out.println("El instrumento está sonando.");
    }
}

public class Guitarra extends Instrumento {
    public void tocar() {
        System.out.println("La guitarra está tocando un temazo.");
    }
}
```

   a) Si escribimos lo siguiente:

```
Instrumento i = new Guitarra();
i.tocar();
```

   ¿qué método se ejecuta y por qué?

   b) ¿Qué pasaría si *tocar()* no estuviera en *Guitarra* pero sí en *Instrumento*?

4. (1,25p) Dada esta clase *Persona*:

```
public class Persona {
    public int edad;
}
```

una clase externa modifica la edad así: ***p.edad = -5;***

¿Qué problema representa esto? ¿Cómo aplicarías encapsulación para evitarlo y validar la edad >= 0 correctamente?

5. (1p) Explica qué mecanismo tiene *Java* para dar solución a la herencia múltiple de clases, la cual no está permitida.

6. (1,25p) Considera el siguiente código:

```
int resultado = 18;

for (int i = 1; i <= 3; i++) {
    for (int j = 3; j >= 1; j--) {
        resultado -= i;
        System.out.println(i + " - " + j + " - res: " + resultado);
    }
}

System.out.println("resultado final: " + resultado);
```

¿Qué valores van tomando ***i***, ***j***, ***resultado*** y las diferentes salidas por pantalla durante la ejecución del programa? Realiza una traza.

7. (1,25p) Tenemos el siguiente método recursivo:

```
public static void main(String[] args) {
    recursiva(4);
}

public static void recursiva(int n) {
    if (n == 0){
        return;
    }
    System.out.print("<");
    recursiva(n - 1);
    System.out.print(">");
}
```

a) Haz el seguimiento haciendo uso de una pila de llamadas y dibuja lo que devuelve impreso por pantalla.
b) ¿Qué pasaría si ***n*** fuera negativo?

8. (1p) Supón que estás programando un sistema bancario y necesitas lanzar una excepción cuando un usuario intenta retirar más dinero del que tiene disponible.
Crea una excepción personalizada llamada *FondosInsuficientesException*, que extienda de *Exception* ò *RuntimeException* y justifica por qué. Invócala desde el siguiente método de ejemplo:

```
public void retirar(double cantidad, double saldo) {
    if (cantidad > saldo) {
        // lanzar excepción
    } else {
        saldo -= cantidad;
    }
}
```

9. (1p) ¿Qué imprime este código?

```
int[][] matriz = {{5, 3, 2},{1, 4, 6},{0, 7, 8}};

int res = 0, j=2;
for (int i = 0; i < matriz.length; i++) {
    res += matriz[i][j];
}
System.out.println(res);
```

¿Cómo modificarías el bucle si quisieras sumar siempre la última fila?