1. ¿Qué puede decir del siguiente código? compila? ¿Qué conceptos de OOP logra identificar?
   Desconozco el lenguaje, pero debería compilar, se ve bien tipado y usado. Se identifican clases e instanciación.

2. ¿Cuál es el resultado del siguiente código ? ¿por qué?
   Desconozco el lenguaje, pero creo que debería imprimir:
   5
   5
   10
   Porque methodOne imprime dos veces 5 sin mutar a, por lo que a sigue teniendo largo 10.

3. Cual es la diferencia entre cohesión y acoplamiento
   Cohesión es que una misma parte del código, como un módulo, tenga un sentido o coherencia común en cuanto a funcionalidad, mientras que acoplamineto es que partes distintas del código sean muy dependientes entre sí.

4. ¿Cuál es la diferencia entre estos try/catch ?
   El primero vuelve a lanzar el error, sirve para atender tipos de error particulares con condicionales. El segundo solo atrapa el error.

5. Mencione cual de los principios SOLID conoce y como lo ha aplicado?, si no los reconoce investigue y explique Single responsability (piense que le va explicar a un niño)
   Single responsability quiere decir que cada parte del código debe enfocarse en una sola funcionalidad, objetivo u tarea en concreto. Es como los músicos de una orquesta, cada una se especializa en algo. Si se necesita alguien responsable de muchos instrumentos, es mejor tener un director. El director solo se enfoca en dirigir, en lugar de aprender cada instrumento musical.

6. Mencione y explique 2 patrones de diseño.
   El patrón Singleton se usa cuando se necesita solo una instancia de una clase para restringir las instancias de esta.
   El patrón Decorador sirve para modificar las funcionalidades a un objeto sin modificar el comportamiento de otras instancias de su clase, por ejemplo mediante una función extra o constructor de otra clase que tome el objeto, en vez de modificar los métodos de la clase original.

7. Qué ventajas le ve a las pruebas unitarias.
   Permiten chequear comportamientos esperados de distintas funcionalidades de un programa, detectando errores a tiempo, como por ejemplo romper funcionalidades antiguas al implementar nuevas.