**Preguntas**

1. ¿En qué consiste el principio de responsabilidad única?

El Principio de Responsabilidad Única establece que "una clase debe tener una y solo una única causa por la cual puede ser modificada". Entonces, entendiéndose "responsabilidad" como una razón de cambio, si una clase tiene más de una razón o motivo para modificarse, entonces tiene más de una responsabilidad. Para entenderse mejor este principio de la Programación Orientada a Objetos (POO), se puede tomar como ejemplo una clase Factura, la cual tiene un método que calcula el total de la misma. Si dentro de este método se calculan deducciones y se aplica IVA, para calcular el total facturado, la clase ya puede tener más de un motivo de cambio, ya que el porcentaje del IVA puede cambiar o las deducciones podrían aplicar diferentes dependiendo del cliente. De manera que, bajo estas situaciones u otras, podría modificarse la clase, indicando que ésta tiene más de una responsabilidad.

Continuando el ejemplo anterior, para dejar una sola responsabilidad a la clase Factura, se podría segregar responsabilidades, creando una clase IVA y una clase Deducción con sus respectivos atributos y métodos para calcular su total, de manera que, de presentarse una razón de cambio de las mencionadas anteriormente, se podría ir directamente a la clase creada indicada y realizar un único cambio, y no afectaría a la clase Factura, ya que ésta solo haría el llamado al método de cálculo de cada clase nueva, y ninguna de estas dos situaciones implicaría un cambio en ésta, ya que su única responsabilidad sería la de calcular el total facturado.

El propósito de este principio es, principalmente, simplificar el código, facilitando su mantenimiento y lo que ello conlleva. Por otro lado, la aplicación de este principio permite que haya un menor acoplamiento y una mayor cohesión, ya que se aumenta la granularidad de la solución, permitiendo a su vez, que haya más posibilidades de reutilización de código y la creación de pruebas específicas para cada responsabilidad.

1. ¿Qué características tiene en su opinión el código limpio?

En mi opinión un código puede considerarse limpio mientras se encuentre bien documentado y en su efecto tenga:

* Un excelente uso de los recursos, es decir, no exista código repetido ya sean clases o métodos.
* Que un método no tenga muchas líneas de código ya que entre más líneas de código tenga un método más difícil será encontrar los bugs del sistema, además, si se tienen métodos generales será más fácil integrarlos a otras funcionalidades.
* No tener código, métodos o clases que no se usen sin importar que estas puedan representar funcionalidades que a futuro se vayan a realizar, no es bueno adelantarse al futuro, pues aunque una funcionalidad se vaya a efectuar no sabemos si se realice de la misma forma en como se pensó en un principio y esto solo refleje pérdida de tiempo.
* Los nombres de las clases, métodos y variables deben dar claridad del objetivo de la misma ya que con el tiempo solemos olvidar las cosas y es una pérdida de tiempo estar revisando el esquema del sistema para comprender que hace cada cosa.
* Cada método debe ejecutar una única función no se pueden realizar múltiples procesos en una misma función, esto hace referencia al hecho de hacer un excelente uso de los recursos y evitar código repetido (responsabilidad única).
* Utilizar correctamente los modificadores de acceso.