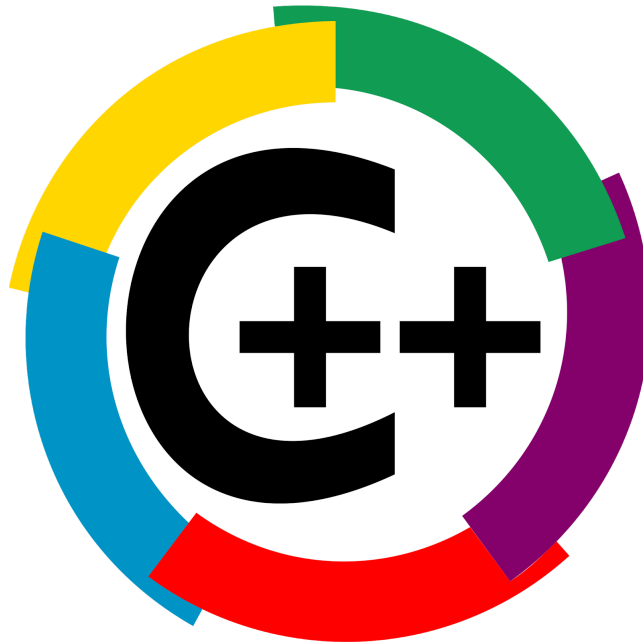


Seminario 1

Cuestiones de la tarea 1.2 - Estudio del enfoque orientado a objetos



Componentes

Guillermo Girón García
Guillermo López García
Jesús Sánchez Borrego

Universidad de Cádiz
6 de Septiembre 2018

1. ¿Cuáles son los elementos/componentes que caracterizan un sistema software desarrollado mediante una metodología orientada a objetos?:

Principalmente son tres:

- Objetos: los cuales contienen una serie de datos, a habitualmente conocidos como atributos.
- Procedimientos: es el código que permite interactuar con esos datos, y que a menudo son conocidos como métodos.
- Clases: es donde se definen los formatos de los atributos y procedimientos para un tipo de objeto. Pueden también contener datos y procedimientos mismos.

2. Diferencia entre clase y objeto:

Una clase es un compendio de datos, y métodos a partir de las cuales se pueden crear los objetos, que no son, ni más, ni menos, que una instancia de una clase. Es decir, contienen todos los atributos y métodos pertenecientes a una clase, pero con una serie de valores en un momento determinado.

3. Relación entre encapsulamiento y ocultación de información:

La encapsulación nos permite que si una clase no deja hacer llamadas de acceso al código interno de los datos de una clase, y permite acceder mediante métodos solamente, ocultar la información del exterior y aumentar las abstracción.

4. Comenta la siguiente afirmación: La ocultación de información limita la comunicación entre los objetos:

En absoluto. La ocultación de la información, se podría decir que restringe la comunicación de un objeto con el exterior, o como en el caso que nos ocupa, otro objeto. Haciendo inaccesibles, campos o métodos del mismo que podría interferir en la comunicación y dar lugar a posibles errores si se manipulan desde un objeto inadecuado o simplemente de una manera incorrecta.

5. Ventajas que proporciona la herencia a las tareas de programación:



Las ventajas que aporta el uso de herencia son muchísimas, aunque a continuación destacaremos las más importantes y esenciales:

- El uso de la herencia en POO, en el caso de querer especializar una clase ya existente, tiene una enorme ventaja y es que no es necesario nuevamente el diseño de toda la clase de nuevo completa, con la herencia se usaría la clase ya creada sin necesidad de la pérdida de tiempo para tener que volver a crear el diseño de una clase nueva con sus métodos y atributos que ya hemos hecho. Usaríamos los mismos atributos y métodos que se repiten y el diseño de la nueva clase heredada tan sólo tendría los atributos nuevos que esa nueva clase tendría y atributos de esa clase, el resto, los hereda de la clase base.
- También una gran ventaja esencial es definir atributos y métodos nuevos para una subclase, que luego se sumarían a los atributos y métodos heredados, esto es muy ventajoso ya que de esta forma, se consigue crear una estructura jerárquica de clases cada vez más especializada.

6. Clasifica por niveles de importancia, según tu criterio, los factores externos de calidad del software. Justifica la respuesta:

Al tratarse de factores externos, nos centraremos en su importancia de cara a lo que pide el cliente/usuario final.

A continuación detallamos la importancia de mayor a menor:

- Corrección: este punto lo considero de los más valiosos ya que es la capacidad de que el producto software tenga adecuación funcional, es decir, que hace lo que dice que hace. No nos valdría de nada un programa que esté muy optimizado, que sea muy robusto... si al fin y al cabo, el programa no va a hacer lo que dice que hace.
- Eficiencia, facilidad de uso y solidez: estos mismos los pondría por igual, y detallaremos el porqué. Cuando un cliente compra un producto software, el cliente quiere en cierta forma que donde se vaya a comprar el producto software genere confianza. Con estas tres características, el cliente obtendrá un software seguro, sin necesidad de estar pendiente si la licencia es o no legal, si el software necesitará actualizaciones por futuros errores de seguridad que tenga y que sea intuitivo de manera que el cliente aparte de la documentación no necesite en demasía tanta genialidad para manejarlo. La solidez, a su vez, es muy importante ya que es el complemento ideal al punto anterior nombrado y la corrección, gracias a las excepciones tendremos un correcto flujo de ejecución del código. Respecto a la facilidad de uso de nada valdría que el programa sea eficiente, económico... si al fin y al cabo el cliente no sabrá dar uso al software. Por último, la eficiencia, en un producto software el cliente “da por sabido” que el software

que va a usar es un software que se usará en una máquina en la cual aprovechará al máximo todos sus recursos.

- Extensibilidad y compatibilidad: si el cliente quiere constantes versiones diferentes del software, se puede adaptar en base a lo que pida sin problema alguno y evitando incompatibilidades con otro software que esté usando. A su vez, la extensibilidad es otro punto a tener bastante en cuenta ya que si el cliente final decide a medida que va usando el software ciertas ampliaciones del programa o bien porque se haya dado cuenta que necesita más funcionalidades o incluso porque hasta que no está terminado un software no se mira más allá de lo que va a hacer y es en la fase de pruebas por parte del cliente cuando utiliza la aplicación cuando se da cuenta de las carencias que nunca había pensado y decide ahora tenerlas.
- Portabilidad y reutilización: un software portable a la hora del desarrollo cuanto más portable sea más rango de usuarios finales tendrá. La reutilización es también importante ya que si un software se quiere adaptar a nuevas modificaciones para el desarrollador será más fácil y el cliente esperará bastante menos a la hora de la entrega, ya que en gran medida el aprovecharse bastante código se ahorra tiempo, dinero...

7. ¿En qué medida afecta la eficiencia a la corrección de un programa?:


En ninguna, ya que la corrección de un programa se basa en si hace lo que tiene que hacer, es decir, si cumple el objetivo para el que ha sido diseñado (su especificación, no en cuanto tiempo lo hace. De eso se encarga propiamente la eficiencia.

8. Diferencia entre compatibilidad y portabilidad:

La compatibilidad se basa en la facilidad de compaginar o mezclar varios elementos software con otros. Sin embargo, la portabilidad se basa en la capacidad de transferir esos componentes software a distintos sistemas hardware.

9. Factores de calidad que contribuyen a la fiabilidad del software:

Los factores de calidad que facilitan que un software sea fiable son:

- Eficiencia
 - Portabilidad
 - Facilidad de uso
- 

-
- Integridad
 - Extensibilidad
 - Robustez
 - Verificabilidad
 - Compatibilidad
 - Corrección
 - Reutilización

