TareasTema1.pdf

Anónimo

Programación Orientada a Objetos

2º Grado en Ingeniería Informática

Escuela Superior de Ingeniería
Universidad de Cádiz

Programación Orientada a Objetos

Tema 1. Evolución histórica y principios fundamentales

Cuestiones de la tarea 1.1: Estudio del paradigma de programación orientada a objetos y su evolución histórica (6h)

1. Diferencia principal entre el paradigma de la programación orientada a objetos y el paradigma de la programación estructurada.

La diferencia fundamental estriba en que la POO trata de organizar el sistema en torno a los objetos que intervienen en él, en vez de hacerlo alrededor de los procesos y los datos, que es como se lleva a cabo en las metodologías estructuradas.

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

Objetos, Clases de objetos, Datos y Operaciones de las clases y Relaciones entre ellas.

3. Diferencia entre clase y objeto.

Un objeto es una entidad presente en el sistema que se está desarrollando que posee unos datos (estado) y un conjunto de operaciones que trabajan sobre ellos (comportamiento).

En cambio, una clase es una descripción general que permite representar un conjunto de objetos similares. Por definición, todos los objetos que existen dentro de una clase comparten los mismos atributos y métodos.

Mientras que un objeto es una entidad que posee un conjunto de datos y de operaciones, una clase es una descripción general que permite representar un conjunto de objetos similares.

4. Relación entre encapsulamiento y ocultación de información.

El encapsulamiento consiste en abstraer los datos y operaciones necesarias para describir un cierto tipo de entidades del mundo real, se implementa mediante las clases de objetos.

Mientras que la ocultación de datos impone límites sobre lo que el usuario programador podrá manipular y/o ver, se implementa mediante las partes pública y privada de las clases.

La relación se encuentra en que los datos que se ocultan se encapsulan para poder hacer este proceso más fácilmente.

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

Falso, la ocultación de información sólo afecta a la visibilidad de los componentes, la comunicación entre los mismos sigue siendo igual.

6. Ventajas que proporciona la herencia a las tareas de programación.

Reutilización de código y su consecuente ahorro.
Generalización y especialización.

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

Corrección
Robustez
Extensibilidad
Reutilización
Compatibilidad
Eficiencia
Portabilidad
Facilidad de uso
Funcionalidad
Oportunidad
Integridad
Economía
Facilidad de mantenimiento

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

En la medida en que esa eficiencia es necesaria para la corrección del programa, es decir, si un programa necesita una respuesta en un tiempo determinado, la eficiencia del subprograma será necesaria para su corrección.

9. Diferencia entre compatibilidad y portabilidad.

La portabilidad es la capacidad que ha de tener el programa para funcionar en distintos entornos operativos.

La compatibilidad, en cambio, determina el grado de portabilidad que tiene dicho programa.

Por lo tanto, a más compatibilidad tenga un programa con los distintos entornos operativos, más portable será.

10. Factores de calidad que contribuyen a la fiabilidad del software.

La calidad de un software es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario.

La fiabilidad del software se define en términos estadísticos como la probabilidad de operación libre de fallos de un programa de computadora es un entorno determinado y durante un tiempo específico.

Factores:

Adecuación funcional: Representa la capacidad del producto software para proporcionar funciones que satisfacen las necesidades declaradas e implícitas.

Eficiencia de desempeño: Esta característica representa el desempeño relativo a la cantidad de recursos necesarios para utilizar el software.

Usabilidad: Capacidad del producto software para ser entendido, aprendido, usado y resultar atractivo para el usuario.

Mantenibilidad: Esta característica representa la capacidad del producto software para ser modificado efectiva y eficientemente.

Compatibilidad: Capacidad de dos o más sistemas o componentes para intercambiar información y/o llevar a cabo sus funciones requeridas cuando comparten el mismo entorno hardware o software

Portabilidad: Capacidad del producto o componente de ser transferido de forma efectiva y eficiente de un entorno hardware, o software, a otro.

Fiabilidad: Capacidad de un sistema o componente para desempeñar las funciones especificadas, cuando se usa bajo unas condiciones y periodo de tiempo determinados

Seguridad: Capacidad de protección de la información y los datos de manera que personas o sistemas no autorizados no puedan leerlos o modificarlos.