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

Los elementos que caracterizan un sistema software orientado a objetos son los objetos y las clases.

2.) Diferencia entre clase y objeto.

Los objetos son entidades dentro de un programa que consta de un estado y de un comportamiento.

Las clases son abstracciones que representan un conjunto de objetos con un comportamiento e interfaz común.

Una clase no es más que una plantilla para la creación de objetos.

3.) Relación entre encapsulamiento y ocultación de información.

La encapsulación es un mecanismo que consiste en organizar datos y métodos de una estructura, conciliando el modo en que el objeto se implementa. La encapsulación garantiza la integridad de los datos que contiene un objeto.

La ocultación de la información establece que el usuario de una clase no necesita saber cómo están estructurados los datos dentro de ese objeto, es decir, un usuario no necesita conocer la implementación. Al evitar que el usuario modifique los atributos directamente y obligándole a utilizar funciones definidas (interfaces) se garantiza la integridad de los datos del objeto.

Con ambos métodos garantizamos la integridad de los datos del objeto de la clase que estamos creando.

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

Se refiere a que los atributos privados de un objeto no pueden ser modificados ni obtenidos a no ser que se haga a través del paso de un mensaje. Esto supone proporcionar una interfaz estable que proteja el resto del programa de la implementación que es susceptible a cambios. Ejemplo: encapsulación.

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

Ayuda ahorrar código y tiempo, ya que la clase padre ha sido implementada y verificada con anterioridad, solo hay que referenciar desde la clase derivada a la clase base.

Los objetos pueden ser construidos a partir de otros similares (Se necesita que exista una clase base).

La clase derivada hereda el comportamiento y los atributos de la clase base.

Toda clase puede servir como clase base para crear otras.

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

Corrección: Es la capacidad de los productos software de realizar con exactitud sus tares tal y como se definen en las especificaciones.

Robustez: Capacidad de los sistemas software de reaccionar apropiadamente ante condiciones excepcionales.

Facilidad de uso: Facilidad con la cual personas con diferentes formaciones y aptitudes pueden aprender a usar los productos software y aplicarlos a la resolución de problemas, facilidad de instalación, operación y supervisión.

Eficiencia: Capacidad de un sistema software para exigir la menos cantidad de recursos hardware.

Extensibilidad: Facilidad de adaptar productos software a los cambios de especificación

Reutilización: Capacidad de los elementos software de servir para la construcción de muchas aplicaciones diferentes.

Compatibilidad: Facilidad de combinar unos elementos de software con otros

Portabilidad: Facilidad de transferir los productos software a diferentes entornos hardware y software.

Funcionalidad: Conjunto de posibilidades que proporciona un sistema.

Oportunidad: Capacidad de un sistema de software de ser lanzado cuando los usuarios lo desean o antes.

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

Un programa eficiente es aquel que consume los mínimos recursos posibles del ordenador.

Un programa correcto es aquel que cumple el objetivo para el cual fue creado.

Por lo tanto, si disponemos de un programa para una tarea aleatoria y conseguimos que sea eficiente, será también correcto si sigue el algoritmo de resolución de dicha tarea.

8.) Diferencia entre compatibilidad y portabilidad.

La compatibilidad es la propiedad de un programa que hace que un sistema logre comprender a ese programa correctamente.

La portabilidad es aquella propiedad de los programas que les permite ejecutarse sobre plataformas diferentes. En muchas ocasiones, para referirse al término portabilidad se dice que un programa es “multiplataforma”.

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

- Funcionalidad: Capacidad de un programa para proporcionar funciones que cubran las necesidades requeridas cuando es utilizado bajo las condiciones especificadas.

- Fiabilidad: Capacidad de un programa para mantener el nivel especificado de rendimiento cuando es usado bajo las condiciones especificadas.

- Usabilidad: Capacidad de un programa para ser entendido, aprendido, usado y atractivo para el usuario, cuando es usado bajo las condiciones especificadas.

- Eficiencia: Capacidad de un programa para proporcionar un rendimiento adecuado, relatico a la cantidad de recursos utilizados, bajo unas condiciones establecidas.

- Mantenibilidad: Capacidad de un programa para ser modificado. Las modificaciones pueden incluir correcciones, mejoras o adaptaciones del programa a cambios en el entorno, en los requerimientos o las especificaciones funcionales.

- Portabilidad: Capacidad de un programa para ser transferido de un sistema a otro.

- Efectividad: Capacidad para permitir al usuario alcanzar objetivos especificados con precisión e integridad en un contexto especificado de uso.

- Productividad: capacidad para permitir al usuario utilizar cantidades apropiadas de recursos con relación a la efectividad alcanzada en un contexto especificado de uso.

- Seguridad: Capacidad para alcanzar niveles aceptables de riesgo de daño para las personas, software, equipos o entorno en un contexto especificado de uso.

- Satisfacción: Capacidad para satisfacer a los usuarios en un contexto especificado de uso.

- Corrección: Es el grado en el que un programa satisface sus especificaciones y consigue los objetivos encomendados por el cliente.

- Integridad: Es el grado con que puede controlarse el acceso al programa o a los datos a personal no autorizado.

- Reusabilidad: Es el grado en que partes de un programa pueden utilizarse en otros programas.