Unidad IV

Programación orientada a objetos y modelado.

4.1 Características del modelo orientado a objetos.

El diseño y modelado Orientado a Objetos es una manera fundamental que provee

un medio uniforme para moldear un sistema desde la captura de requerimientos en la etapa inicial del análisis hasta la implementación, atravesando todo el ciclo de desarrollo del sistema.

En la actualidad, se define el software orientado a objetos como una “técnica” para la construcción de un software marcado por ciertas características, que lo llevan a cumplir los siguientes objetivos:

Robustez: Siendo esta la capacidad del software para reaccionar ante condiciones excepcionales. Claro que esta parte tiene muchas cuestiones, por

ejemplo, ¿reacción apropiada? Ampliaremos este aspecto en las próximas líneas.

Extensibilidad: La capacidad de adaptar el software a los cambios en las especificaciones. Se suelen definir dos principios para mejorar la simplicidad de diseño y la descentralización, indicando que los mismos deben ser para mejorar la adaptación sobre la estructura compleja sobre la que seguramente se asocian. Se habla de descentralización si el software se construye bajo el concepto de módulos o Servicios, por lo que un cambio afectará a un conjunto de módulos y no provocará una reacción en cadena.

Reutilización: Requerimiento base de OOP la cual es la capacidad que tienen los distintos elementos software para coonstruir aplicaciones diferentes.

Eficiencia: Se define como la capacidad de un sistema software para exigir el mínimo de recursos (hardware, software) en la utilización de sus tareas. Se aplica como sinónimo de rendimiento.

Portabilidad: Capacidad de utilizar el software en diferentes entornos

hardware y software.

4.2 Elementos primordiales en el modelo de objetos.

La programación Orientada a Objetos trata de cumplir las necesidades de los usuarios finales, estás tareas se realizan mediante la modelización del mundo real, el sopote fundamental es el modelo objeto.

Los elementos más importantes de este modelo son:

Abstracción
La abstracción, un principio por el cual se aísla toda aquella información que no resulta relevante a un determinado nivel de conocimiento.

Encapsulamiento
En programación modular, y más específicamente en programación orientada a objetos, se denomina encapsulamiento al ocultamiento del estado, es decir, de los datos miembro, de un objeto de manera que sólo se puede cambiar mediante las operaciones definidas para ese objeto.

Cada objeto está aislado del exterior, es un módulo natural, y la aplicación entera se reduce a un agregado o rompecabezas de objetos. El aislamiento protege a los datos asociados a un objeto contra su modificación por quien no tenga derecho a acceder a ellos, eliminando efectos secundarios e interacciones.
De esta forma el usuario de la clase puede obviar la implementación de los métodos y propiedades para concentrarse sólo en cómo usarlos. Por otro lado se evita que el usuario pueda cambiar su estado de maneras imprevistas e incontroladas.

Modularidad
La modularidad es la capacidad que tiene un sistema de ser estudiado, visto o entendido como la unión de varias partes que interactúan entre sí y que trabajan para alcanzar un objetivo común, realizando cada una de ellas una tarea necesaria para la consecución de dicho objetivo. Cada una de esas partes en que se encuentre dividido el sistema recibe el nombre de módulo. Idealmente un módulo debe poder cumplir las condiciones de caja negra, es decir, ser independiente del resto de los módulos y comunicarse con ellos (con todos o sólo con una parte) a través de unas entradas y salidas bien definidas.

Jerarquia y Herencia
La Jerarquía es una propiedad que permite la ordenación de las abstracciones. Las dos jerarquías más importantes de un sistema complejo son: estructura de clases (jerarquía “es-un” (is-a): generalización/especialización) y estructura de objetos (jerarquía “parte-de” (part-of): agregación).
Las jerarquías de generalización/especialización se conocen como herencia. Básicamente, la herencia define una relación entre clases, en donde una clase comparte la estructura o comportamiento definido en una o más clases (herencia

simple y herencia múltiple, respectivamente).
La agregación es el concepto que permite el agrupamiento físico de estructuras relacionadas lógicamente. Así, un camión se compone de ruedas, motor, sistema de transmisión y chasis; en consecuencia, camión es una agregación, y ruedas, motor, transmisión y chasis son agregados de camión.
stefanycuevas

4.3 Representación gráfica del diseño.

La disciplina del diseño industrial requiere obligatoriamente la expresión gráfica como medio de representación de ideas. En el caso de las propuesta necesitaremos el dibujo conceptual y para el diseño definitivo del producto requeriremos del dibujo descriptivo.

Rapid Sketching

El dibujo conceptual es la manera de manifestar el pensamiento, sensibilizar las implicaciones de diseño, cristalizar las ideas, plasmar el aspecto global del producto. Es ideal incorporar descripciones breves de la configuración, materiales y dimensiones del producto con el uso de flechas y palabras. Siendo bocetos no precisa una escala exacta pero si debe ser fiel a la proporción para define correctamente la tridimensionalidad del objeto. Su grado de calidad visual se basa en un boceto conceptual el cual debe tener un perfil definido, con líneas de contorno y secciones resaltadas, llamados a detalles e incorporación de datos específicos.

Veneno

La representación pictórica de gran calidad visual fue empleada después de la depresión económica de los años 30 con el objetivo de vender la idea. Para mediados del siglo XX en Estados Unidos se trabajaba con la obsolescencia estética de los productos para la continua renovación de la oferta en el mercado. El dibujo técnico era inadecuado para cautivar miradas, su representación a pesar de ser muy descriptiva no era sugerente, no funcionaba para fines publicitarios. Desde entonces el styling, sobretodo en el ámbito del diseño automotriz es ampliamente usado.

4.4 Relación entre la programación orientado a objetos y la estructurada

Una vez estudiados los conceptos fundamentales dentro del paradigma de objetos, se abordan cuestiones importantes relacionadas con las clases de objetos, como por ejemplo la jerarquía de clases, y los conceptos de herencia y extensión de las clases. Se usa Java como lenguaje que permite poner en práctica los conocimientos aprendidos.

1. Relaciones entre clases

En la forma pura de tecnología de objetos, sólo existen dos relaciones: cliente y herencia. Éstas corresponden a dos clases distintas de dependencias posibles entre dos tipos de objetos A y B:

• B es un cliente de A si todo objeto de tipo B puede contener información sobre uno o más objetos de tipo A.

• B hereda de A si B denota una versión especializada de A.

La relación cliente es lo suficientemente amplia como para abarcar muchas formas de dependencia, como por ejemplo lo que se conoce como agregación (la presencia en cada objeto de tipo B de un subobjeto de tipo A).

La relación de herencia abarca la especialización en sus muchas y diferentes formas. En el resto del tema se hablará de la relación de herencia entre clases.

2. El concepto de herencia

Inventar clases nuevas y únicas es posible, pero los sistemas interesantes rara vez surgen de la nada. Casi siempre, el nuevo software se construye sobre desarrollos previos; la mejor manera de crearlo es por imitacin, refinamiento y combinación. Los métodos de diseño tradicionales han ignorado durante mucho tiempo este aspecto del desarrollo de sistemas, que en la tecnología de objetos es una cuestión esencial.

La herencia es un mecanismo que facilitará la creación de clases nuevas a partir de otras ya existentes. Constituye uno de los conceptos centrales de los métodos orientados a objetos y tiene profundas consecuencias en el proceso de desarrollo del software