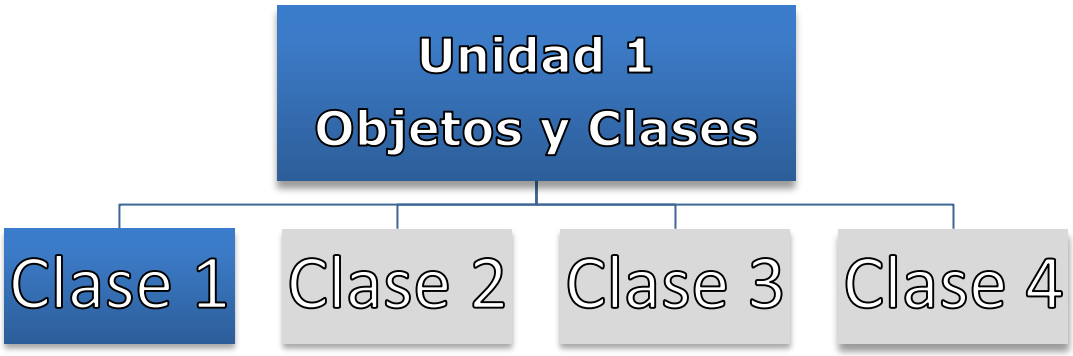
A continuación, le presentamos el material de la clase 1. A partir de estos podrá comenzar a abordar los conceptos sobre los sistemas complejos.

PROGRAMACIÓN ORIENTADA A OBJETOS



Docente titular y autor de contenidos: Prof. Ing. Darío Cardacci

Presentación

La clase 1 corresponde a la unidad 1 de la asignatura. En esta unidad presentaremos los principios básicos de la **programación orientada a objetos**. Los mismos serán necesarios para el desarrollo de software aplicando esta metodología.

Presentaremos los contenidos desde dos perspectivas: en la primera expondremos los **aspectos estáticos**, mientras que, desde la segunda, mostraremos los **elementos dinámicos** de la programación orientada a objetos.

Para lograr lo expuesto, analizaremos el concepto y significado de las ***clases*** en el modelo orientado a objetos, su valor como especificación estática y simplificada de una porción de la realidad perteneciente al dominio del problema tratado y la manera de definirlas correctamente. También explicaremos las formas en que se pueden relacionar, así como las ventajas y desventajas de usar cada una de ellas.

Luego veremos cómo estas especificaciones estáticas, denominadas *clases*, permiten la generación de ***objetos*** en un plano dinámico del modelo, para poder utilizarlos en nuestros programas. Esto nos llevará a analizar las características esenciales de los *objetos*, la forma que tienen para relacionarse y el modo de utilizarlos correctamente.

En esta asignatura no estudiaremos la forma de analizar y diseñar software orientado a objetos por ser temas que exceden el ámbito de la programación orientada a objetos.

No obstante, para evitar que la falta de un análisis y diseño previo afecte la comprensión de algunas de las técnicas que expondremos, le propondremos utilizar la teoría y aplicar la práctica en la resolución de distintas tareas.

También queremos aclarar que, si bien hemos seleccionado una tecnología y una herramienta en particular para el desarrollo de las actividades propuestas, los conceptos teóricos no se limitan a ellas,

son aplicables a otras tecnologías y herramientas que no utilizaremos en este curso.

Para finalizar esta presentación, consideramos oportuno aclarar que la teoría y la técnica que estudiará en esta asignatura sirven como punto de partida hacia un innumerable conjunto de conocimientos asociados, que se presentarán como lecturas o trabajos sugeridos.

Invitamos s Ud. a incursionar en los temas propuestos, ya que le brindarán no sólo un aporte tecnológico a su formación, sino que podrá mejorar su perspectiva profesional sobre la visión que posee acerca de *cómo desarrollar software*.

Por todo lo expresado hasta aquí, esperamos que usted, a través del estudio de esta unidad logre:

- Abstraer realidades complejas a escenarios simplificados.
- Reconocer elementos intervinientes en un problema.
- Definir los elementos significativos de un dominio en términos de características cualitativas y cuantitativas.
- Definir los elementos significativos de un dominio en términos de las acciones que puede desarrollar.
- Construir especificaciones de clases.
- Diferenciar clases y objetos.
- Conocer las particularidades y formas de relacionarse que poseen las clases.
- Conocer las particularidades y formas de relacionarse que poseen los objetos.
- Comprender la orientación a objetos como un medio para poder desarrollar software de alta calidad utilizando una metodología sólida que hace que los desarrollos posean un nivel aceptable de extensibilidad y flexibilidad.
- Identificar y explicar los conceptos y las características que deben poseer los objetos y las formas en que se pueden relacionar.

- Comprender el concepto de clase en el modelo orientado a objeto.
- Identificar las relaciones que se pueden dar entre clases para desarrollar una arquitectura orientada a objetos en las aplicaciones.

Los siguientes **contenidos** conforman el marco teórico y práctico de esta unidad. A partir de ellos lograremos alcanzar el resultado de aprendizaje propuesto. En negrita encontrará lo que trabajaremos en la clase 1.

- **El modelo orientado a objetos.**
- **Jerarquías "Es - Un" y "Todo - Parte".**
- **Concepto de Clase y Objeto.**
- Características básicas de un objeto: estado, comportamiento e identidad.
- Ciclo de vida de un objeto.
- Modelos. Modelo estático. Modelo dinámico. Modelo lógico. Modelo físico.
- Concepto de análisis diseño y programación orientada a objetos.
- Conceptos de encapsulado, abstracción, modularidad y jerarquía.
- Concurrencia y persistencia.
- Concepto de clase.
- Definición e implementación de una clase
- Campos y Constantes
- Propiedades. Concepto de Getter() y Setter(). Propiedades de solo lectura. Propiedades de solo escritura. Propiedades de lectura-escritura. Propiedades con indizadores. Propiedades autoimplementadas. Propiedades de acceso diferenciado.

- Métodos. Métodos sin parámetros. Métodos con parámetros por valor. Métodos con parámetros por referencia. Valores de retorno de referencia.
- Sobrecarga de métodos.
- Constructores. Constructores predeterminados. Constructores con argumentos.
- Finalizadores.
- Clases anidadas.

A continuación, le presentamos un detalle de los contenidos y actividades que integran esta unidad. Usted deberá ir avanzando en el estudio y profundización de los diferentes temas, realizando las lecturas requeridas y elaborando las actividades propuestas, algunas de desarrollo individual y otras para resolver en colaboración con otros estudiantes y con su profesor tutor.

1. Los sistemas complejos (Teoría)

Lectura obligatoria

Cardacci Dario y Booch, Grady. Orientación a Objetos. Teoría y Práctica. Buenos Aires, Argentina. Pearson Argentina, 2013. Capítulo 1.

Links a temas de interés

Descarga de visual studio:

https://visualstudio.microsoft.com/es/downloads/

Enlace al lenguaje c#

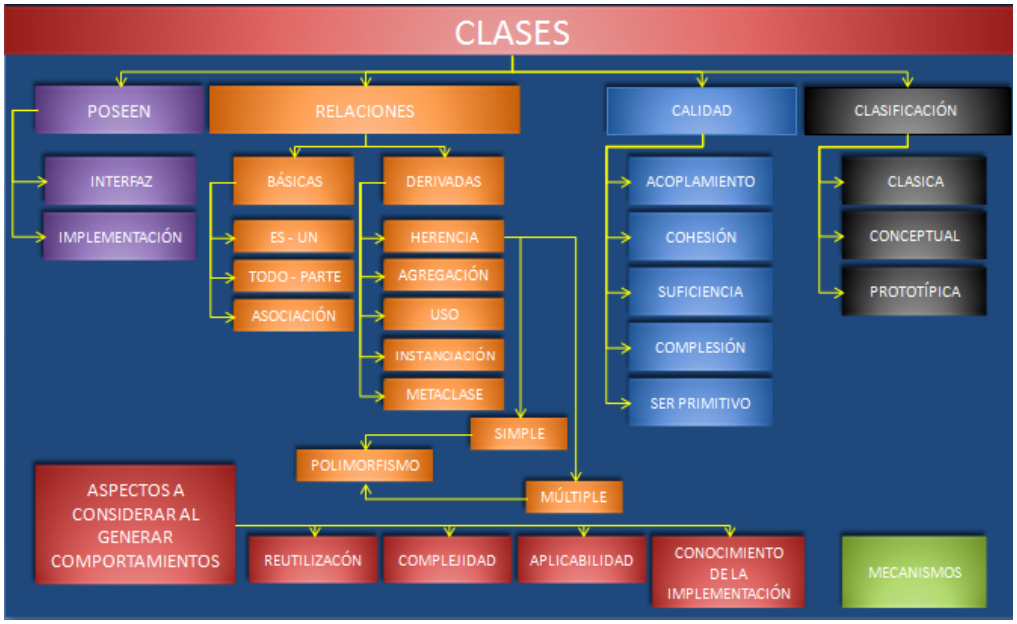
https://docs.microsoft.com/es-es/dotnet/csharp/

Enlace a la object Management Group

www.omg.org

Organizador Gráfico

El siguiente esquema le permitirá visualizar la interrelación entre los conceptos que a continuación abordaremos.



Lo invitamos a comenzar con el estudio de los contenidos de la clase 1.

1. Los sistemas complejos

La **complejidad** es una cualidad que no le es ajena a lo humano. Lo que nos rodea cotidianamente es básicamente complejo, aunque en muchas oportunidades tendemos a mirar una dimensión de las cosas cuya abstracción nos permite convivir con ello.

También lo **sistémico** es inherente a lo humano. De hecho, el mismo ser humano puede ser observado como un sistema, en particular, como un sistema colaborativo de partes perfectamente organizadas para lograr un objetivo particular.

El software también es complejo, especialmente en los tiempos que vivimos.

En general, los requerimientos solicitan como mínimo que este producto represente significativamente y acompañe, organice y simplifique, los procesos empresariales en el caso de los sistemas de información y los procesos sociales, en el caso de los sistemas de esta naturaleza.

Por supuesto, pueden existir algunos sistemas de software que no son complejos, donde todas las actividades ligadas a él, desde la concepción hasta el mantenimiento, son efectuadas por una única persona. Generalmente, su ciclo de vida es limitado, así como los objetivos que persiguen.

Indudablemente, no son estos desarrollos los que nos convocan a analizar y aprender todo lo que veremos en esta asignatura sino los que son complejos y requieren nuestra atención.

Ahora bien, concentrémonos en el grupo de sistemas que nos interesan. Estos en general poseen un tamaño mayor a los simples, no suelen producirse por un individuo sino por grupos de trabajo. Estas dos características, *dimensión y cantidad de trabajadores*, colaboran para que la complejidad sea aún mayor.

Con el avance de la ingeniería del software y el desarrollo de prácticas más especializadas se ha logrado **administrar la complejidad**. Con esto queremos decir que la mantenemos en niveles aceptables, aunque siempre está allí y nunca se logra eliminarla por completo, permanece latente, al acecho que profesionales descuidados abandonen las buenas prácticas y sufran sus consecuencias.

Una pregunta muy frecuente es aquella que plantea si el software es inherentemente complejo o los desarrolladores lo transforman en algo complejo. Nos atrevemos a considerar que, si bien no hay que restarle méritos a las malas

prácticas, los elementos constitutivos básicos que se necesitan para concebir y desarrollar un software son de por sí complejos.

Cuando nos referimos a los elementos complejos queremos representar o visualizar a la complejidad desde distintos ángulos, a saber.

a. La complejidad del dominio del problema.
b. La complejidad de gestionar un proyecto de desarrollo de software.
c. La complejidad que genera la no estandarización de los componentes de software. Cabe aclarar que se avanzó en este sentido, no obstante falta mucho camino por recorrer.
d. La complejidad resultante del cambio dinámico introducido por las constantes modificaciones en el dominio del problema y la tecnología.

La **abstracción** es un mecanismo que permite comprender con mayor facilidad sistemas complejos. En alguna oportunidad se ha mencionado que la estética cumple un rol fundamental en el entendimiento de los sistemas complejos, refiriéndonos a *estética como el descubrimiento de nuevas estructuras y relaciones que colaboren en la resolución de los problemas complejos que debemos enfrentar*.

Retomando el concepto de abstracción podríamos imaginarla como la acción por la cual luego de observar un elemento rescatamos aspectos relevantes. En caso que la observación suceda sobre varios elementos además de los relevantes se rescatarían los comunes a ellos.

A modo de ejemplo podemos imaginar que estamos en presencia de un gran número de vehículos de transporte terrestre. Si intentásemos analizarlos íntegramente, no existe duda que nos enfrentaríamos a una complejidad significativa. Esto es debido a que cada uno de ellos posee características motrices diferentes (motores diesel, motores a nafta, motores eléctricos, motores híbridos etc.).

También encontraríamos que en algunos casos ciertos mecanismos funcionan de manera eléctrica en un conjunto de vehículos, mientras en otros lo hacen en forma hidráulica o mecánica. Sus terminaciones podrían ser de cuero en algunos casos, pana en otros y plásticos o sintéticos para otro grupo.

De esta forma, la descripción podría ser infinitamente grande inclusive llegando al extremo de analizar atómicamente los compuestos de cada parte. Si bien esto no estaría mal, probablemente perdamos el rumbo en cuanto a conceptualizar lo que es un vehículo de transporte terrestre.

Para ello es que se recurre a la abstracción y es este sentido que se ejercita la posibilidad de capturar aquellas cosas que hacen a un vehículo de transporte terrestre. Podemos mencionar que posee ruedas y un mecanismo de encendido, que su forma de tracción nos permite ir a distintas velocidades y en dos sentidos opuestos, que su sistema de dirección nos permite cambiar el rumbo hacia donde nos dirigimos y que posee mecanismos para disminuir la velocidad hasta detenerlo y apagar el funcionamiento del motor.

Con la abstracción realizada seguramente no tendremos mucha información pormenorizada del vehículo, pero habremos comprendido para qué sirve y cuáles son sus funciones más importantes.

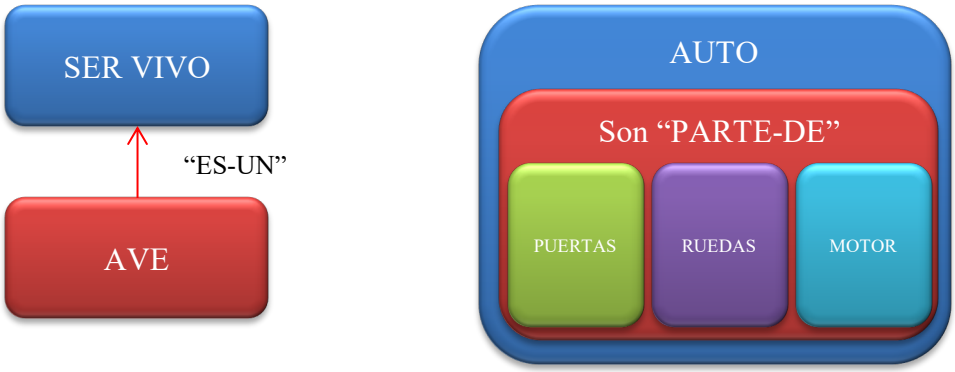
Ahora bien, imaginemos un conductor experto que haya manejado una cantidad significativa de vehículos terrestres. Consciente o inconscientemente ha realizado abstracciones de ellos que permanecen en su memoria. Dada la circunstancia en la cual deba enfrentarse a la conducción de un nuevo vehículo, estas le darán las pautas para poder hacerlo y con un breve período de instrucción lo realizará sin problemas. Esto sucede gracias a las abstracciones que permitieron que comprendiera un sistema complejo rápidamente, sustentándose en sus peculiaridades más significativas.

Al analizar las características de la abstracción subyace el concepto de **descomposición**. Esta descomposición es la que nos permite comprender los conceptos que intentamos *administrar* y al mismo tiempo se establece una *jerarquía*. En principio podemos asociar esta jerarquía diciendo que los elementos más abstractos son de mayor jerarquía que aquellos elementos más específicos. Si bien existen muchas formas de jerarquías existen dos en particular que nos interesan.

- Una de ellas se denomina **jerarquía estructural** y se la reconoce por la forma "parte-de". Por ejemplo, podemos expresar que rueda es "parte-de" vehículo o que llanta es "parte de" rueda. Se observa claramente que auto posee más jerarquía que rueda y rueda posee más jerarquía que llanta. También un observador al mirar podría decir que observa un vehículo, y estaría estableciendo un determinado nivel de abstracción, pero si agudiza su vista vería ruedas con lo cual estaría estableciendo un nivel de abstracción que reviste mayor detalle que el anterior. He aquí la explicación sobre la cual sustentamos que existe una relación entre los distintos niveles de abstracción y los distintos niveles de jerarquías.
- La otra se denomina **jerarquía de tipos** y se la reconoce por la forma "es-un". Esta forma de jerarquía es la que se establece por la

naturaleza de las cosas. Así podemos decir que un águila "es un" ave y ave "es un" ser vivo.

Estas dos jerarquías sustentarán toda la **estructura de clase** (*jerarquía estructural* y *jerarquía de tipos*) que utilizaremos de aquí en adelante. Los objetos que instanciemos responderán a estas formas estructurales y de tipos.



Antes de la orientación a objetos se trabajaba con una forma algorítmica para *descomponer problemas*. Esta forma descompone en pasos lógicos un problema ordenándolo en una secuencia estructurada de manera que representen el **dominio del problema**.

En orientación a objetos se visualizan "objetos" del dominio del problema, se determina como se relacionan y se establecen comportamientos en términos de "responsabilidades de hacer" y "colaboraciones que espera recibir" para hacer lo que se le pide.

Pensemos en cómo funciona un auto. Bajo un esquema algorítmico descompondríamos el problema en una secuencia de pasos. Por ejemplo:

Bajo la orientación a objetos trabajaríamos comenzando con abstraer el dominio del problema, "Auto". Luego descomponemos auto de acuerdo a los principios de la "jerarquía estructural (parte-de)". Allí obtenemos partes especializadas del mismo, a saber: alarma, puerta, llave, sistema de encendido etc. Les determinamos características distintivas a cada uno y acciones que debe desarrollar y finalmente los relacionamos.

MATERIAL DE REPASO CONCEPTUAL

PROGRAMACIÓN ORIENTADA A OBJETOS

Teoría de Objetos

Titular: Dario Guillermo Cardacci

¿Por qué objetos?

- Crecimiento de los sistemas.
- Sistemas más complejos.
- Realidad cambiante.
- Requisitos altamente dinámicos.
- Problemas de mantenimiento.
- Equipos de trabajo numerosos.
- Arquitecturas complejas.
- etc

El rol del Observador

- Determinar el dominio del problema.
- Observar objetivamente.
- Abstraer y establecer modelos simplificados.
- Modelos sustentados en características y comportamientos.

El resultado está afectado por las experiencias previas que posee el observador.

UAI El futuro sos vos.
PROGRAMACIÓN ORIENTADA A OBJETOS
Titular: Dario G. Cardacci

Por qué el software es complejo

- La complejidad del dominio del problema.
- La dificultad de gestionar el proceso de desarrollo.
- La flexibilidad que se puede alcanzar a través del software.
- Los problemas de caracterizar el comportamiento de sistemas discretos.

Atributos de un sistema complejo

1. Frecuentemente, la complejidad toma la forma de una jerarquía, por lo cual un sistema complejo se compone de subsistemas, y así sucesivamente, hasta que se alcanza algún nivel ínfimo de componentes elementales.

UAI El futuro sos vos.
PROGRAMACIÓN ORIENTADA A OBJETOS Titular: Dario G. Cardacci

Atributos de un sistema complejo

2. La elección de qué componentes de un sistema son primitivos es relativamente arbitraria y queda en gran medida a decisión del observador.

Atributos de un sistema complejo

3. Los enlaces internos de los componentes suelen ser más fuertes que los enlaces entre componentes.

Este hecho tiene el efecto de separar la dinámica de la estructura interna de los componentes de la dinámica que involucra la interacción entre los componentes.

UAI El futuro sos vos.
PROGRAMACIÓN ORIENTADA A OBJETOS Titular: Dario G. Cardacci

Atributos de un sistema complejo

4. Los sistemas jerárquicos están compuestos usualmente de solo unas pocas clases diferentes de subsistemas en varias combinaciones y disposiciones.

Atributos de un sistema complejo

5. Se encontrará invariablemente que un sistema complejo que funciona ha evolucionado de un sistema simple que funcionaba. Un sistema complejo diseñado desde cero nunca funciona y no puede parchearse para conseguir que lo haga. Hay que volver a empezar, partiendo de un sistema simple que funcione.

UAI El futuro sos vos.
PROGRAMACIÓN ORIENTADA A OBJETOS Titular: Dario G. Cardacci

Jerarquías

- "parte – de" (part of) Estructura de Clases
- "es – un" (is a) Estructura de Objetos

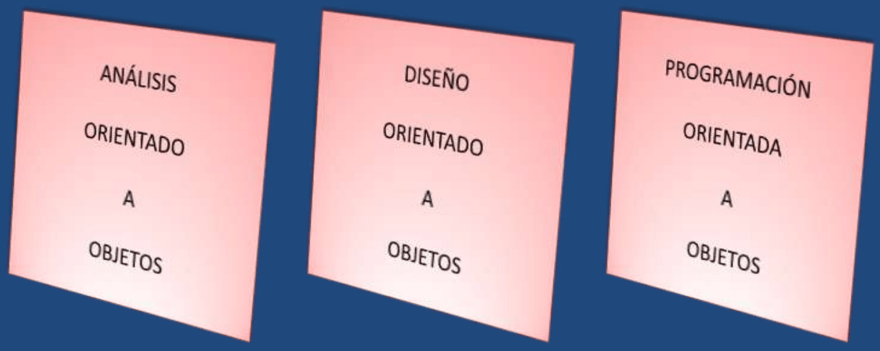
Descomposición

Divide y Vencerás

- Algorítmica.
- Orientada a Objetos.

UAI El futuro sos vos.
PROGRAMACIÓN ORIENTADA A OBJETOS
Titular: Dario G. Cardacci

Metodología Orientada a Objetos



Elementos del Modelo de Objetos

Elementos Fundamentales

- Abstracción.
- Encapsulamiento.
- Modularidad.
- Jerarquía.

Elementos Secundarios

- Tipos.
- Concurrencia.
- Persistencia.

UAI El futuro sos vos.
PROGRAMACIÓN ORIENTADA A OBJETOS
Titular: Dario G. Cardacci

Abstracción

- Una abstracción denota las características esenciales de un objeto que lo distinguen de todos los demás tipos objetos y proporciona así fronteras conceptuales nítidamente definidas respecto a la perspectiva del observador.

Encapsulamiento

- Oculta los detalles de implementación de un objeto.
- Es el proceso de almacenar en un compartimiento los elementos de una abstracción que constituyen su estructura y su comportamiento; sirve para separar la interfaz contractual de una abstracción y su implantación.

UAI El futuro sos vos.
PROGRAMACIÓN ORIENTADA A OBJETOS
Titular: Dario G. Cardacci

Modularidad

- La modularidad es la propiedad que posee un sistema que ha sido descompuesto en un conjunto de módulos cohesivos y débilmente acoplados.

Jerarquía

- La jerarquía es una clasificación u ordenación de abstracciones.

UAI El futuro sos vos.
PROGRAMACIÓN ORIENTADA A OBJETOS
Titular: Dario G. Cardacci

Tipos

- Los tipos son la puesta en vigor de la clase de los objetos, de modo que los objetos de tipos distintos no pueden intercambiarse o, como mucho, pueden intercambiarse solo de formas muy restringidas.

Tipos
Ligaduras

- Ligadura Estática o Temprana
- Ligadura Dinámica o Tardía

UAI El futuro sos vos.
PROGRAMACIÓN ORIENTADA A OBJETOS Titular: Dario G. Cardacci

Concurrencia

- La concurrencia permite a diferentes objetos actuar al mismo tiempo.
- La concurrencia es la propiedad que distingue un objeto activo de uno que no está activo.

Persistencia

- La persistencia es la propiedad de un objeto por la que su existencia trasciende el tiempo (es decir, el objeto continúa existiendo después de que su creador deja de existir) y/o el espacio (es decir, la posición del objeto varía con respecto a espacio de direcciones en el que fue creado).

UAI El futuro sos vos.
PROGRAMACIÓN ORIENTADA A OBJETOS
Titular: Dario G. Cardacci

UAI El futuro sos vos.
PROGRAMACIÓN ORIENTADA A OBJETOS
Titular: Dario G. Cardacci

FIN

Actividades asincrónicas

Guía de preguntas de repaso conceptual

1. Enumere y explique los aspectos más relevantes que hacen que un software de gran magnitud sea complejo.

2. ¿Cuáles son los cinco atributos de un sistema complejo?

3. ¿Cuáles son las dos jerarquías más importantes que consideramos en la orientación a objetos para sistemas complejos?

4. ¿Con qué podemos enfrentar a la complejidad para obtener partes cada vez más pequeñas y simplificadas del dominio del problema?

5. ¿Cuáles son las dos formas de descomposición más conocidas?

6. ¿Explique en qué se diferencia la descomposición algorítmica y la orientada a objetos?

7. ¿Qué rol cumple la abstracción en la orientación a objetos?

8. ¿Qué rol cumple la jerarquía en la orientación a objetos?

9. ¿Consideraría Ud. al diseño orientado a objetos un desarrollo evolutivo o revolucionario? Justifique.

10. ¿Cuántos y cuáles son los modelos básicos que se manejan en el diseño orientado a objetos?

11. ¿Qué es la programación orientada a objetos?

12. ¿Qué es el diseño orientado a objetos?

13. ¿Qué es el análisis orientado a objetos?

14. ¿Cuáles son los elementos fundamentales en el modelo de objetos?

15. ¿Cuáles son los elementos secundarios del modelo de objetos?

16. Explique el significado de la abstracción.

17. Explique el significado del encapsulamiento.

18. Explique el significado de la modularidad.

19. Explique el significado de la jerarquía.

20. Explique el significado de la tipificación.

21. Explique el significado de la concurrencia.

22. Explique el significado de la persistencia.

23. ¿Cómo se denotan las características esenciales de un objeto que lo distinguen de todos los demás tipos de objetos y proporciona así fronteras conceptuales nítidamente definidas respecto a la perspectiva del observador?

24. ¿A qué denominamos un objeto cliente?

25. ¿A qué denominamos un objeto servidor?

26. ¿A qué denomina Meyer el modelo contractual de programación?

27. ¿Qué establece un contrato entre objetos?

28. ¿Cómo se denomina a las formas en que un objeto puede actuar y/o reaccionar, constituyendo estas formas la visión externa completa, estática y dinámica de la abstracción?

29. ¿Cómo se denomina al conjunto completo de operaciones que puede realizar un cliente sobre un objeto, junto con las formas de invocación u órdenes que admite?

30. ¿A qué nos referimos cuando decimos que un concepto central de la idea de abstracción es el de invariancia?