

## Diseño de Software Orientado a Objetos

## ¿Qué es Orientación a Objetos?

- Definición de clases y herencia.
- Pasaje de mensajes.
- Métodos estáticos y virtuales.
- Etc.
- Pero:
  - usar un lenguaje orientado a objetos no es condición necesaria ni suficiente para hacer un programa orientado a objetos.

2

## Diseño de Software

- El diseño del software consiste en crear un modelo del software deseado de modo que satisfaga los requisitos de buena forma.

Requisitos



¿qué?

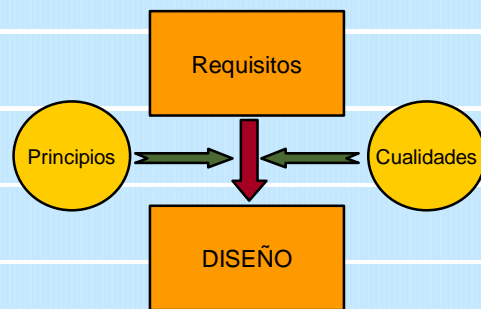
Diseño



¿cómo?

3

## Proceso de Diseño



4

## Métodos Estructurados de Diseño

- Diseño Estructurado
  - Yourdon, 1979
- Análisis Estructurado de Sistemas
  - Gane & Sarson, 1979
- Desarrollo de Sistemas Jackson
  - Jackson, 1983
- Diseño Orientado a Objetos
  - Robinson, 1992
  - Booch, 1994
  - UML, 1997

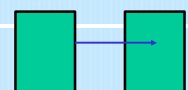


Más que métodos, son notaciones y guías para el diseño de software. La creatividad y el rigor siguen siendo la esencia del diseño.

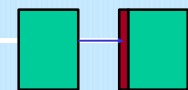
5

## Diseño por Responsabilidades

- Dar a alguien una responsabilidad implica:
  - se espera un cierto comportamiento,
  - se le dará independencia para hacer su tarea.
- Los principios de delegación y no interferencia son esenciales.
- Encapsulamiento y modularidad como un arte.
- Base para la reutilización.



Programación imperativa



Programación por responsabilidades

6

## Sistemas Chicos y Grandes

- Programación en chico:
  - código desarrollado por uno o unos pocos programadores,
  - una persona entiende todas las facetas del sistema.
- Programación en grande:
  - muchas personas participan en la programación,
  - los especificadores, diseñadores y programadores pueden ser distintas personas,
  - nadie abarca todo el conocimiento del sistema,
  - el manejo de la comunicación es un asunto importante y complejo.

7

## Modelar el Comportamiento

- En general es más fácil comprender la funcionalidad del sistema antes que otros aspectos (datos, sincronización, etc.)
- Diseño Dirigido por la Responsabilidad.

8

## Ejemplo: Ayudante de Cocina

- Producto a desarrollar:
  - Ayudante de Cocina Interactivo e Inteligente (ACII).
- Las especificaciones son poco más que el nombre del producto.
- Se supone que éste es el primero de una línea de productos.

9

## Requisitos

- El ACII es una aplicación para PC que reemplaza los libros de recetas de cocina.
- No sólo contiene recetas de cocina, sino que ayuda en la planificación de la comida por un cierto período (una semana).
- El usuario puede interactuar con el ACII para planear el menú de cada día.
- El sistema escala los menús de la semana según el número de comensales e imprime una lista de ingredientes a comprar.

10

## Identificar Componentes

- El diseño de software complejo se inicia identificando componentes.
- Una componente es una entidad abstracta que puede desarrollar una tarea, o sea, asumir una responsabilidad.
- Características de una componente (en el diseño):
  - tener un conjunto pequeño y bien definido de responsabilidades;
  - interactuar mínimamente con otras componentes.

11

## Tarjetas CRC

- Las tarjetas CRC (Componente - Responsabilidad - Colaborador) son una técnica para organizar y registrar las responsabilidades de las componentes.
- Cada tarjeta contiene:
  - nombre de la componente,
  - lista de responsabilidades,
  - otras componentes con las que interactúa.
- Cada miembro del equipo de desarrollo tiene una o más tarjetas y está encargado de simular un escenario de ejecución, pasándole el control a otro objeto cuando corresponda.

Nombre de la Componente	Colaboradores
Lista de Responsabilidades	

12

## Ciclo “Qué - Quién”

- Cada actividad, tarea o funcionalidad identificada en la especificación de requisitos debe ser asignada a una componente como parte de sus responsabilidades.
- Si una componente tiene muchas responsabilidades, es posiblemente mejor separarla en dos o más sub-componentes.
- El documento de diseño puede empezar a escribirse en este momento:
  - aún el análisis se hace a nivel del sistema completo,
  - las decisiones de diseño están frescas en la mente de los desarrolladores.

13

## Componentes y Comportamiento

- Al comenzar la ejecución del ACII, se presentará una pantalla de Bienvenida con las siguientes opciones:
  - recorrer las recetas existentes,
  - agregar una nueva receta a la base de datos,
  - modificar una receta existente,
  - revisar un plan de comidas existente,
  - crear un nuevo plan de comidas.
- Las actividades pueden clasificarse en dos grupos:
  - acciones sobre las recetas,
  - acciones sobre los planes de comidas.

14

## CRC de Bienvenida

Bienvenida	Colaboradores
Responsabilidades	<ul style="list-style-type: none"> <li>• Base de Datos de Recetas</li> <li>• Planes de Comida</li> </ul>
<ul style="list-style-type: none"> <li>• Desplegar mensaje inicial</li> <li>• Ofrecer las opciones de acción</li> <li>• Pasar el control a:                             <ul style="list-style-type: none"> <li>• Base de Datos de Recetas</li> <li>• Planes de Comida.</li> </ul> </li> </ul>	

15

## CRC de Base de Datos de Recetas

Base de Datos de Recetas	Colaboradores
Responsabilidades	<ul style="list-style-type: none"> <li>• Bienvenida</li> <li>• Planes de Comida</li> </ul>
<ul style="list-style-type: none"> <li>• Recorrer de las recetas,</li> <li>• Modificación de recetas,</li> <li>• Incluir nuevas recetas.</li> </ul>	

16

## El Cambio es Inevitable

- Los cambios deben afectar el menor número posible de componentes.
- Concentrar los cambios probables en una o dos componentes:
  - interfaces, formatos de comunicación y salida.
- Minimizar las dependencias del hardware (portabilidad).
- Minimizar el acoplamiento entre componentes.
- Documentar las razones de las decisiones de diseño como apoyo para futuros cambios.

17

## Posponer Decisiones

- ¿Hay que presentarle al usuario una lista de categorías de recetas a recorrer?
  - ¿Podrá el usuario dar palabras clave como una serie de ingredientes y buscar las recetas con esos ingredientes?
  - ¿Deben usarse barras de scroll o el cursor para moverse a través de la lista de recetas?
- è Estas decisiones no son fundamentales porque afectan solamente la componente Bienvenida (anticipación del cambio).

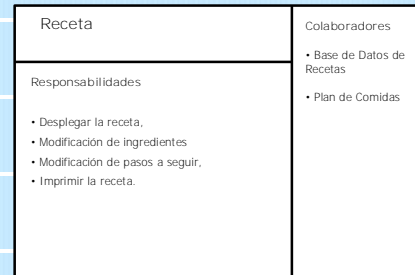
18

## Recetas

- Cada Receta será también una componente.
- Debe contener una cierta información:
  - ingredientes necesarios,
  - pasos a seguir.
- Las actividades que debe realizar una receta son:
  - desplegar la receta en pantalla,
  - permitir modificar ingredientes y/o pasos,
  - imprimir la receta.
- Este es un modelo de formato para cada una de las recetas.
- Cuando la Base de Datos ofrece incluir una nueva receta:
  - se crea una nueva componente de Receta en blanco;
  - se asignan los valores de la información con los datos proporcionados por el usuario.
- La Base de Datos da el inicio pero la componente Receta realiza las tareas.

19

## CRC de Receta



20

## Otras Componentes

- El Plan de Comidas permite (responsabilidad):
  - crear un nuevo plan,
  - editar y modificar un plan existente,
  - imprimir un plan de comidas,
  - calcular la lista de compras necesarias.
- Un nuevo plan tiene una Fecha asociada.
- Cada Fecha es también una componente:
  - consultar la comida de una Fecha particular,
  - imprimir la comida del día.

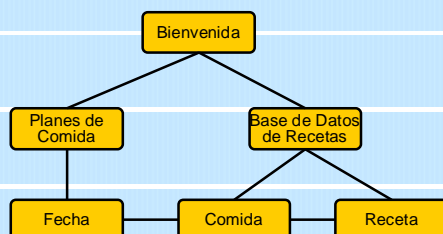
21

## Más Componentes

- Una Comida es la descripción de todas las Recetas a servirse en una determinada Fecha:
  - número de comensales,
  - desplegar la Comida,
  - editar/modificar recetas de la Comida,
  - imprimir la Comida.

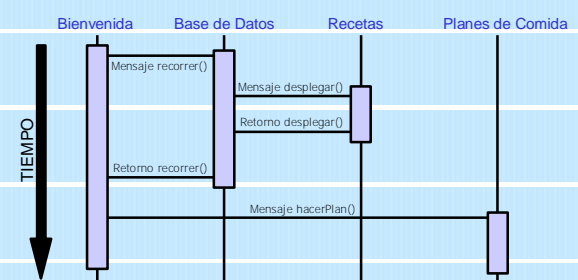
22

## Estructura de ACII



23

## Diagrama de Secuencia



24

## Estado y Comportamiento

- El Comportamiento de una componente es el conjunto de acciones que puede realizar.
- El Estado de una componente es la información contenida dentro de ella.
- El conjunto de todas las acciones posibles de una componente se llama el *protocolo* de la componente.
- El Estado de una componente no es estático sino que puede cambiar con el tiempo.

u No es necesario que una componente mantenga variables de Estado.

25

## Instancias y Clases

- Existen varias Recetas en la Base de Datos:
  - probablemente tienen distintos ingredientes y distintos pasos a seguir (estado);
  - pero todas tendrán las mismas responsabilidades (comportamiento).
- Iniciando la especificación de un sistema, nos centramos en el comportamiento común de las Recetas.
- Una Clase es un conjunto de objetos con el mismo comportamiento.
- Un individuo de una Clase es una Instancia; la caracteriza su estado.

26

## Interfaz e Implementación

- Definir el comportamiento de una componente permite que otro programador pueda usarla sin conocer su implementación.
- Esto se llama ocultamiento de la información (information hiding).
- Decimos que la componente encapsula su comportamiento y muestra sólo los detalles necesarios para usarla.
- La Interfaz describe lo que la componente puede hacer:
  - es la cara de la componente hacia otros programadores.
- La Implementación describe cómo la componente realiza sus tareas:
  - es la cara de la componente que ve el programador trabajando en ella.

27

## Interfaz e Implementación

- La separación de la Interfaz y la Implementación es quizás uno de los principales principios de la Ingeniería de Software (separación de intereses).
- El ocultamiento de la información es especialmente útil en desarrollos donde participan muchas personas:
  - el factor crítico aquí es la comunicación;
  - las componentes se desarrollan en forma independiente.
- Es también muy importante para la reutilización de componentes.

28

## Principios de Parnas

- El desarrollador de una componente de software debe proveer al usuario de toda la información necesaria para hacer uso efectivo de sus servicios, y ninguna otra información.
- El desarrollador de una componente de software será provisto de toda la información necesaria para llevar a cabo las responsabilidades que le son asignadas a la componente, y ninguna otra información.
- Varias implementaciones pueden probarse para la misma interfaz.

29

## Siguiendo el Diseño

- Estructura general del sistema:
  - Componentes sin estado interno, pueden implementarse como una función;
  - Componentes con estado y varias responsabilidades pueden implementarse como clases,
    - cada responsabilidad será un método.
  - Todos los datos necesarios deben identificarse:
    - los parámetros de los métodos se identifican con su nombre y su origen,
    - los datos internos se identifican.

30

## Elección de Nombres

- Guías para elección de nombres [Keller, 1990]:
  - usar nombres fácilmente pronunciables;
  - usar mayúsculas o guiones para unir palabras;
  - ser cuidadoso con las abreviaturas;
  - evitar nombres con diferentes interpretaciones;
  - evitar los números como parte de un nombre;
  - las funciones o variables con valores lógicos, deben tener nombres que sugieran este tipo, e.g., estáLista.
- Reescribir las tarjetas CRC con los nombres seleccionados para cada variable y responsabilidad.

31

## CRC de Fecha

Fecha	Colaboradores
Responsabilidades	• Comida • Planes de Comida
• Mantener información de un día específico. Fecha (año, mes, día) - crea una fecha DesplegarEditar() - despliega la información de un día en una ventana permitiendo que el usuario la modifique. CrearListaCompras () - agregar ingredientes de la comida a la lista de compras.	

32

## Diseño Detallado

- Luego de detallar los nombres precisos de cada variable de estado y responsabilidad, cada componente puede desarrollarse separadamente.
- En el diseño detallado:
  - se eligen las estructuras de datos internas,
  - se implementan los métodos definidos en la interfaz.

33