Programación orientada a objetos

Capítulo 13

Diseñar aplicaciones

No se puede mostrar la imagen en este momento.

13.1.1 El método verbo/sustantivo

Las clases de un sistema se corresponden aproximadamente con los **sustantivos** que aparecen en la descripción del mismo; los métodos se corresponden con los **verbos**.

- Los sustantivos definen cosas (clases y objetos)
- Los verbos describen acciones (Métodos)

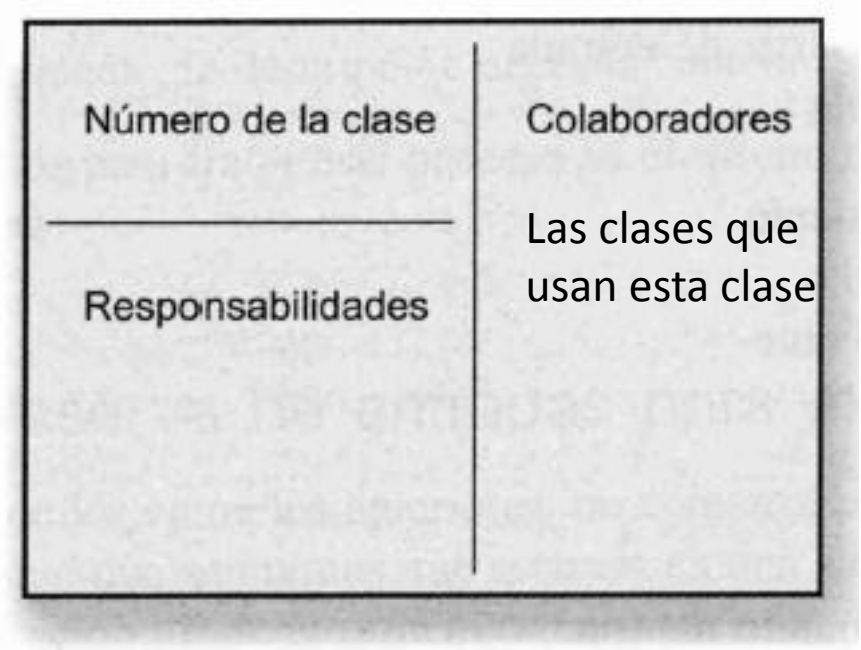
Ejemplo de reserva de entrada de cine

El sistema de reserva de entradas para el cine debe almacenar reservas para varias salas. En cada sala, los asientos están ubicados en filas. Los clientes pueden reservar entradas y se les da un número de fila y un número de asiento. El cliente puede requerir la reserva de varios asientos consecutivos.

Cada entrada es para una función en particular, es decir, para la exhibición de una determinada película en un cierto horario. Las funciones se realizan a determinada fecha y hora en la sala designada para exhibirlas. El sistema almacena el número de teléfono del cliente.

Sustantivos	**Verbos**
sistema de reserva de entradas para el cine	*almacena* (reservas de entradas) *almacena* (número de teléfono)
reserva de entrada sala	*tiene* (asientos)
asiento fila cliente	*reserva* (asientos) *se le da* (número de fila, número de asiento) *solicita* (reserva de entrada)
número de fila número de asiento función	*se designa* (una sala)
película fecha hora número de teléfono	

13.1.4 Usar Tarjetas CRC Clase/Responsabilidad/Colaboradores



Escenarios (casos de uso)

- Un escenario es un ejemplo de actividad que el sistema tiene que llevar adelante o proporcionar
- Hacemos una “simulación” en grupo y en vivo, para comprender mejor el funcionamiento del sistema a diseñar

Se pueden usar los **escenarios** (también conocidos como «casos de uso») para comprender las interacciones de las clases en el sistema.

PCA	
Calcular la matriz eigenvectores y pesos de las imágenes de entrenamiento.	ImageSet
Reconocimiento de un individuo en una imagen de entrada.	neuralNetwork
Guardar o cargar entrenamientos anteriores.	Image
	TabDemo

Ejercicio 13.7 Haga un diseño de clases para un sistema de simulación de control de un aeropuerto. Use tarjetas CRC y escenarios. Aquí está una descripción posible del sistema:

El programa es un sistema de simulación de un aeropuerto. Necesitamos, para nuestro nuevo aeropuerto, conocer si podemos operar con dos pistas de aterrizaje o si necesitamos tres. El aeropuerto funciona de esta manera:

El aeropuerto tiene varias pistas de aterrizaje. Los aviones despegan y aterrizan en pistas de aterrizaje. Los controladores del tráfico aéreo coordinan el tráfico y le dan permisos a los aviones para despegar o aterrizar. Algunas veces, los controladores dan el permiso directamente, otras veces le dicen a los aviones que esperen. Los aviones deben mantener una cierta distancia entre ellos. El propósito del programa es simular el aeropuerto en operación.

13.2 Diseño de clases

- Por las tarjetas CRC sabemos que clases vamos a necesitar
- Diseñar interfaces de clases
  - Decidir los métodos públicos que tendrá la clase
    - El escenario ahora contempla las llamadas a métodos, parámetros y valores de retorno
  - Métodos esqueletos (stubs)
    - Un método esqueleto (stubs) es un método que tiene la signatura correcta y el cuerpo vacía
- Diseñar el interfaz de usuario
  - IGU (Interfaz Gráfica de Usuario)
  - El modo que el usuario interactuará con el sistema
- 13.3 Documentación

13.4 Cooperación

- Programación por parejas
  - La **Programación en Pareja** (o *Pair Programming* en inglés) requiere que dos programadores participen en un esfuerzo combinado de desarrollo en un sitio de trabajo. Cada miembro realiza una acción que el otro no está haciendo actualmente: Mientras que uno codifica las pruebas de unidades el otro piensa en la clase que satisfará la prueba, por ejemplo.

Programación por parejas *Tradicionalmente, la implementación de las clases se hace a solas. La mayoría de los programadores trabajan en sus propias clases escribiendo el código y se contratan a otras personas solamente después de que se terminó la implementación, para que prueben o revisen el código.*

Recientemente, se ha sugerido la programación por parejas como una alternativa que intenta producir código de mejor calidad (código con mejor estructura y menos fallos). La programación por parejas es también uno de los elementos de una técnica que se conoce como programación extrema. Busque en Internet «programación por parejas» o «programación extrema» para encontrar más información.

- La **programación extrema** o *eXtreme Programming* (XP) es una metodología de desarrollo de la ingeniería de software formulada por Kent Beck, autor del primer libro sobre la materia, *Extreme Programming Explained: Embrace Change* (1999). Es el más destacado de los procesos ágiles de desarrollo de software. Al igual que éstos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad

13.5 Prototipos

Armar un prototipo sifnifica construir un sistema que funciona parcialmente en el que se simulan algunas de las funciones de la aplicación. Sirve para proporcionar mayor compresión, en una etapa temprana del proceso de desarrollo, de la manera en que funcionará el sistema.

13.6 Crecimiento del software

- Modelo en cascada

En el modelo de cascada, las distintas fases del desarrollo del software se realizan siguiendo una secuencia determinada:

- análisis del problema
- diseño del software
- implementación de los componentes del software
- prueba unitaria
- prueba integral
- entrega del sistema al cliente

- Modelo de desarrollo iterativo
  - El software crece
  - Diseñar un sistema pequeño
  - Ir agregando funcionalidades

13.7 Usar patrones de diseño

Un patrón de diseño describe un problema común, que ocurre regularmente en el desarrollo del software y luego describe una solución general del problema que se puede usar en varios contextos diferentes. La solución de los patrones de diseño de software consiste, típicamente, en la descripción de un conjunto de clases y sus respectivas interacciones.

Un **patrón de diseño** es la descripción de un problema computacional común y la descripción de un pequeño conjunto de clases y su estructura de interacción que ayuda a resolver dicho problema.

Primero:

- Documentan buenas soluciones a problemas planteados

Segundo:

- Los patrones de diseño tienen nombres y de esa manera establecen vocabularios que ayudan a los diseñadores a hablar sobre sus diseños

13.7.1 Estructura de un patrón

La descripción de un patrón incluye como mínimo:

- un *nombre* que se puede utilizar para hablar sobre el patrón convenientemente;
- una descripción del **problema** que resuelve el patrón (frecuentemente dividido en secciones como intento, motivación, pertinencia);
- una descripción de la **solución** (frecuentemente se describe la estructura, los participantes y los colaboradores);
- las **consecuencias** del uso del patrón, incluyendo los resultados y lo que se deja de lado.

- Decorador
- Singleton
- Método Fábrica
- Observador

Resumen de conceptos

- **sustantivo/verbo** En un sistema, las clases se corresponden aproximadamente con los sustantivos de la descripción del problema; los métodos se corresponden con los verbos.
- **escenarios** Los escenarios (conocidos también como «casos de uso») se pueden usar para comprender las interacciones en un sistema.
- **prototipo** La construcción de un prototipo es la construcción de un sistema que funciona parcialmente, en el que algunas funciones de la aplicación están simuladas. Sirve para brindar comprensión sobre cómo funcionará realmente el sistema en las fases iniciales del proceso de desarrollo.
- **patrón de diseño** Un patrón de diseño es la descripción de un problema computacional común y la descripción de un pequeño conjunto de clases y su estructura de interacción que ayuda a resolver dicho problema.