Tema 1: conceptos básicos de orientación a objetos:

Orientación a objetos

Reutilización
Responsabilidades y contratos
Abstracción, ocultación y encapsulamiento
Herencia
Polimorfismo

Abstracción
Concepto
Mero diseño orientado a objetos provee la abstracción
Polimorfismo

Ocultación
Principio de ocultación
Ámbitos

Encapsulamiento
Empaquetar información y funcionalidad, (datos y procesamiento)
Permite mecanismos de sincronización

Visión general:

En nuestros programas tendremos objetos de diferentes clases que se relacionan entre ellos con mensajes que piden servicios a otros objetos. Cada objeto servirá a los mensajes que le lleguen siguiendo el guión marcado por un método. Los objetos son entes con información y funcionalidad.

El diseño orientado a objetos consistirá en

1 Delimitar las clases de objetos que existirán en el universo de nuestro programa, y cómo se pueden relacionar los objetos de las diferentes clases entres sí.
2 Implementar los métodos y los atributos necesarios.
3 Construir un programa con las instrucciones que instancian los objetos necesarios para iniciar la ejecución (situación inicial del universo), inicializando estos objetos a los estados necesarios para que el sistema comience a funcionar correctamente.

Según Budd, se puede ver la ejecución de un programa en orientación a objetos como una simulación: se generan los objetos iniciales y se les deja que interactúen entre ellos. Simular el ciclo de vida de los objetos iniciales.

Resumen del paradigma de orientación a objetos:

- No sólo nuevas características sobre estructurada sino nueva forma de modelar
- Programa: colección de agentes independientes (objetos) responsables de tareas específicas. Ejecución: evolución de un sistema en que interactúan algunos de esos objetos (simulación)

- Objeto: encapsula estado (valores de datos) y comportamiento (operaciones)
- Los objetos son de una clase y las clases son plantillas que definen un comportamiento común a todos los objetos de la misma
- Un objeto muestra su comportamiento mediante la invocación de métodos en respuesta a mensajes. (En programación multihebra, aquí hay que matizar un poco)
- Los objetos amplían el concepto de TDA con la inclusión de la herencia
- La reducción de interdependencia entre componentes software permite el desarrollo de sistemas software reutilizable.
- Mayor nivel de abstracción: se puede programar con una visión de mensajes y descripción de las tareas que ellos realizan, pasando por alto los detalles de implementación.

Conceptos concretos

Ocultación: ámbitos

Principio de ocultación

- Ámbitos
    - Público
    - Privado
    - Protegido (lo veremos con la herencia)
    - (otros: paquetes, namespaces)

Clase

Clase como patrón

Responsabilidades/contratos

Diferencia entre objeto y clase

Clase abstracta

- Interfaz: métodos públicos
- Atributos
    - De clase
    - De instancia (objeto)
- Métodos
    - De clase
    - De instancia
    - Abstractos (Clase abstracta)
    - Finales
- Ámbitos
    - Público
    - Privado
    - Protegido
    - (otros)

Clases estáticas (*singleton*)

Objeto

- Datos y procesamiento
- Concreción de clase
- Estado concreto y propio (no compartido)

- Se comparten los atributos de clase (especie de globales)
- Instanciación: con referencias o con estructuras: variable de objeto ¿qué es?
    - Copia de objetos
    - Comparación de objetos
- Consciencia de sí mismo: **this**
- Consciencia de su clase
- Ciclo de vida
    - Constructor
    - Vida
    - Destructor

Mensaje

Invocación de método (llamada a función). Cambio de nombre sólo por conceptualización.

Método

- Fracción de código que indica cómo debe responder un objeto a un mensaje
    - De clase
    - De instancia
    - Abstracto (clase abstracta)
    - Final

Atributo

- Dato que ayuda a mantener el estado de un objeto.
    - De clase
    - De instancia
    - Por defecto

Errores comunes cuando se diseña:

- Clases que hacen modificaciones directas a otras clases
- Clases con demasiada responsabilidad
- Clases sin responsabilidad
- Clases con responsabilidades que no se usan
- Nombres engañosos
- Responsabilidades desconectadas
- Uso inadecuado de la herencia
- Funcionalidad repetida