Tema 1
Introducción a la programación orientada a objetos

Programación II

Alicia Garrido Alenda

Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante

Paradigma de programación

Paradigma de programación

Proporciona el enfoque y los métodos a un programador para la construcción de un programa.

- Diferentes paradigmas generan diferentes estilos de programación y diferentes formas de construir las soluciones para los problemas.

Paradigmas de programación

Evolución en los paradigmas de programación

Programación imperativa ⇒ Programación estructurada ⇒ Programación basada en objetos ⇒ POO

Programación imperativa

- Programa
  - Conjunto de instrucciones a ejecutar secuencialmente.
  - Se incluyen saltos, bucles, condiciones.
- Las variables representan el contenido de las posiciones de memoria, que se modifica mediante asignaciones o lectura.
- Es el paradigma de programación más ajustado al funcionamiento del ordenador.

Programación estructurada I

Teorema del programa estructurado

- Todo programa puede escribirse utilizando únicamente las tres instrucciones de control siguientes:
  1. Secuencia.
  2. Instrucción condicional.
  3. Iteración o bucle.
- Todas tienen un solo punto de entrada y un solo punto de salida.
- Anidamiento: sólamente con estas 3 estructuras se pueden escribir todos los programas.
- Hoy en día se utilizan las técnicas de programación estructurada de forma conjunta con las de programación modular.

Programación estructurada II

La programación estructurada a veces no puede modelar correctamente el mundo real porque se basa en algoritmos (acciones) mientras que el mundo real se basa en objetos (elementos que realizan acciones).

- La programación estructurada no representa bien las relaciones entre datos y operaciones.
- Los datos:
  - Se asocian a funciones o procedimientos (al revés que en el mundo real).
  - Son globales al problema.

Programación basada en objetos

- Principios:
  - Ocultación de la información
  - Tipos abstractos de datos

- Técnica de programación: Consiste en ampliar el lenguaje existente con nuevas operaciones y tipos de datos definidos por el usuario.
- Técnica de diseño: Consiste en dedicar módulos separados a la realización de cada tipo abstracto de datos y cada función importante.

EXTENSIÓN DE LA PROGRAMACIÓN ESTRUCTURADA

Ocultación de información

- Objetivos:
  - Encapsular la información que se puede modificar para que quede oculta al resto de la aplicación.
  - El acceso a la información se realiza indirectamente a través de interfaces bien definidas.
- Beneficios:
  - Las interfaces permiten que un cambio en la estructura interna de los datos no afecte al resto de la aplicación.
  - Favorece el mantenimiento del *software*.

Tipos abstractos de datos

- Los tipos permiten:
  1. Ocultar la representación interna de los datos.
  2. Ocultar la implementación de las operaciones.
  3. Comprobar la corrección del código en tiempo de compilación.
     - Los valores de un tipo no pueden ser asignados a variables de otro tipo.
     - No pueden realizarse operaciones inconsistentes.
- Los lenguajes proporcionan un conjunto reducido de tipos pero facilitan mecanismos que permiten definir nuevos tipos que modelan entidades más complejas.

Tipo Abstracto de Datos: un tipo de dato que define un conjunto de valores y un conjunto de operaciones que se pueden realizar sobre dichos valores.

Paradigma Orientado a Objetos (OO)

Cambia:

- El concepto de ejecución de programa. Llamadas a funciones → Paso de mensajes entre objetos.
- El concepto de dato (*pasivo*) por el de objeto (*activo*).
- El modo de organización del programa. Funciones+variables → Clases.

POO: Usa objetos y sus interacciones para diseñar aplicaciones de manera que su estructura sea lo más parecida posible al problema real que está modelando.

Un modo de ver el mundo I

- Un mundo estructurado en:
  1. Objetos y clases
  2. Responsabilidades
  3. Mensajes y Métodos
  4. Jerarquías de clases

Un modo de ver el mundo II

- Objetos y clases:

Objeto: Encapsulación de un estado y un comportamiento.

Clase: Categoría que agrupa un conjunto de objetos.

- Un objeto es una instancia de una clase.

Un modo de ver el mundo III

- Responsabilidades:

El comportamiento de cada objeto se describe en términos de responsabilidades.

- Un programa OO solicita a sus estructuras de datos (objetos) que realicen un servicio. La realización del servicio es la responsabilidad del objeto.

Un modo de ver el mundo IV

- Mensajes y métodos:
  - A un objeto se le envían mensajes para que realice una determinada acción.
  - El objeto selecciona un método apropiado para realizar dicha acción.
  - A este proceso se le denomina Paso de mensajes.

Un modo de ver el mundo V

- Jerarquía de clases: **generalización** (herencia)
  - Organización del conocimiento en términos de jerarquías.
  - El conocimiento de una categoría más general es aplicable a una categoría más específica → generalización. Implementación en POO de este concepto → herencia.
  - Las clases pueden ser organizadas en una estructura jerárquica de herencia. Una clase hijo heredará propiedades de una clase padre más alta en la jerarquía (más general).

Características de los LOO I

- Básicas (Alan Kay, 1993):
  1. Todo es un **objeto**
  2. Cada objeto tiene su propia memoria configurada a partir de otros objetos.
  3. Todo objeto es **instancia** de una **clase**.
  4. Todos los objetos de la misma clase pueden recibir los mismos mensajes (realizar las mismas acciones). La clase es el lugar donde se define el **comportamiento** de los objetos y su estructura interna.
  5. Las clases se organizan en una estructura arbórea de raíz única, llamada **jerarquía de herencia**.
  6. Un programa es un conjunto de objetos que se comunican mediante el envío de mensajes.

Características de los LOO II

- Opcionales:
  1. **Polimorfismo**: Capacidad para referenciar distintos elementos en distintos instantes usando el mismo elemento de programa.
  2. **Genericidad**: Definición de clases parametrizadas (definen tipos genéricos).
  3. **Gestión de errores**: Manejo de excepciones.
  4. **Recogida de basura** (garbage collection): Permite eliminar automáticamente la memoria de aquellos objetos que ya no se utilizan.
  5. **Concurrencia**: Permite que diferentes objetos actúen al mismo tiempo.
  6. **Persistencia**: Propiedad por la cual la existencia de un objeto trasciende en el tiempo (normalmente implica el uso de bases de datos).
  7. **Reflexión**: Capacidad de un programa de manipular su propio estado, estructura y comportamiento.

Objetivos de la POO I

- Fiabilidad: corrección + robustez
  - **Corrección**: Capacidad de los elementos software de realizar con exactitud sus tareas, tal y como se definen en las especificaciones.

La corrección está relacionada directamente con el comportamiento de un sistema en los casos previstos por su especificación.

  - **Robustez**: Capacidad de los elementos software de responder adecuadamente ante situaciones inesperadas.

La robustez se caracteriza por el comportamiento de un sistema en los casos no previstos por la especificación.

Objetivos de la POO II

- Modularidad: extensibilidad + reutilización
  - **Extensibilidad**: Facilidad para adaptar los productos de software a los cambios de especificación.
  - **Reutilización**: Capacidad de los elementos software de servir para la construcción de muchas aplicaciones diferentes.

Producir aplicaciones más fáciles de cambiar $\Rightarrow$ Facilidad de mantenimiento