Síntesis conceptual

Asignatura: Programación
Unidad: 7. Clases avanzadas y utilización de objetos

Resumen

Asociación, agregación y composición de clases: Para que un programa tenga sentido y pueda funcionar es necesario definir la relación que tiene cada clase dentro del programa, de modo que esté conectado y tenga un diseño lógico y sólido, requiriendo de relaciones como:

- Asociación: Dos clases tienen un enlace que las une conceptualmente, pero pueden existir sin depender la una a la otra.
- Agregación: Dos clases, con una de las clases declarando a la otra en su interior
- Composición de clases: una clase no tiene sentido sin otra, ambas son dependientes.

Herencia: La relación se define entre clase padre y clase hija. La clase padre tienen toda la funcionalidad y las clases heredarán toda la funcionalidad de la clase padre. Permite generalizar conceptos y crear una jerarquía de clases. Los elementos que la afectan son:

- Constructores y herencia: La herencia de constructores es indirecta, cada clase tiene que definir sus constructores obligatoriamente.
- Acceso a campos de la superclase: Los atributos si son heredables, en ocasiones deberemos emplear la palabra reservada *super* para que ocurra.
- Acceso a métodos de la superclase: Al igual que los atributos debemos emplear la palabra reservada *super*.
- Sobreescritura de métodos (Override): Para sobrescribir un método de la clase padre, hay que volver a crearlo con el mismo nombre, parámetros y retorno que la clase padre y además se tendrá que añadir encima de la cabecera del método la palabra reservada @override, los constructores no pueden ser sobrescritos
- Clases y métodos abstractos: Existen clases que no se instancias, ya que contienen conceptos abstractos, las clases abstractas. Estas y sirven para la herencia modo de plantilla para indicar como pueden definirse sus métodos en las clases hijas.
    - Métodos abstractos: Los métodos abstractos deben ser necesariamente sobrescritos por la clase hija.
- Clases y métodos finales: indican una clase de la cual no es posible heredar.
    - Métodos finales: Método no heredable.

Interfaces: Una interfaz es un prototipo de plantilla, que proporciona un conjunto de métodos que simplifican la funcionalidad de una clase. A diferencia de una clase abstracta, una interfaz solo puede tener variables constantes y métodos sin implementar.

Pseudoherencia múltiple: En Java no es posible tener herencias múltiples, aunque estas se pueden simular mediante interfaces.

Polimorfismo: El polimorfismo es la capacidad para ciertos objetos que tienen una relación entre si adoptar diferentes formas en tiempo de ejecución.

- Asignación polimorfa: Cada clase padre tiene la generalización de todas sus clases hijas, con lo cual, la clase padre puede ser instanciada a partir de cualquiera de sus hijas, pero una clase hija no puede contener la instanciación de una clase padre.
- Ejecución polimorfa: El polimorfismo garantiza que se ejecute correctamente el método de la clase real y no la contenedora (padre) de aquellos métodos que existen por igual en ambas clases. En cambio, cuando la clase real tiene métodos exclusivos que no existen en la clase padre no podremos invocarlos. Para poder hacer la invocación de estos métodos previamente habrá que realizar un casting a la clase real para volver a su origen.

Conceptos fundamentales

- **Instancia**: resultado de particularizar una clase, creando un objeto "real" desde una clase abstracta, pueden existir infinitas instancias de una clase.

- **Clase padre:** también llamada superclase, es aquella desde la que se crean otras clases más específicas.

- **Clase hija**: clase creada a partir de otra clase de su mismo ámbito, pero más general, llamada clase padre.

- **Constructores**: subrutina que permite instanciar objetos a partir de una clase.

- **implements:** palabra reservada que permite llamar y hacer uso de la interfaz.