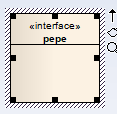
1. La herencia en Java es un concepto de programación orientada a objetos que permite a una clase hija heredar atributos y métodos de otra clase padre.

Beneficios:

* Reutilizar código
* Polimorfismo
* Abstracción y generalización
* Código mas organizado

1. se representa con una flecha que va de la clase hija a la clase padre
2. por ejemplo es mejor en los casos en los que vamos a utilizar código con mismo atributos o que están relacionados y de esta forma poder optimizar
3. una interfaz es una colección de métodos abstractos y constantes que pueden ser implementados por cualquier clase que desee cumplir con esa interfaz.
4. En un diagrama UML, una interfaz se representa de manera similar a una clase, pero con un estereotipo especial de interfaz.



1. Se recomienda utilizar cuando se quiere lograr un bajo acoplamiento, facilitar la interoperabilidad, soportar múltiples implementaciones, lograr el polimorfismo o definir una API pública clara y consistente.
2. Las interfaces son más adecuadas para definir contratos y permitir una mayor flexibilidad en la implementación, mientras que las clases abstractas son útiles para proporcionar una estructura común y algo de implementación a las subclases.
3. Asociación, Agregación, Composición, Herencia, interfaces

* Asociación: Estudiante -----> Universidad
* Agregación: Coche ----◇ Motor
* Composición: Casa ----◆ Habitación
* Subclase ----▷ Superclase
  1. Asociación:

Úsala cuando dos clases estén relacionadas de alguna manera, pero ninguna sea parte integral de la otra. Por ejemplo, una clase "Estudiante" podría estar asociada con una clase "Curso", ya que un estudiante puede inscribirse en varios cursos.

También úsala para modelar relaciones bidireccionales entre clases.

* 1. Agregación:

Empléala cuando una clase "contenga" a otra clase, pero la clase contenida pueda existir independientemente de la clase contenedora. Por ejemplo, una clase "Coche" podría tener una agregación con una clase "Motor", ya que un motor puede ser instalado o retirado de un coche.

Usa agregación cuando haya una relación de "todo a parte", pero la parte pueda ser compartida por múltiples objetos o clases.

* 1. Composición:

Utilízala cuando una clase "contenga" a otra clase y la clase contenida sea parte integral de la clase contenedora. En otras palabras, la vida de la clase contenida está completamente ligada a la vida de la clase contenedora. Por ejemplo, una clase "Casa" podría tener una composición con una clase "Habitación", ya que una habitación no puede existir sin una casa.

Usa composición cuando haya una relación de "todo a parte", y la parte no pueda existir sin el todo.

* 1. Herencia:

Emplea la herencia cuando una clase (subclase) pueda heredar atributos y métodos de otra clase (superclase). Utiliza este tipo de relación cuando haya una relación de "es un" entre las clases. Por ejemplo, una clase "Perro" podría heredar de una clase "Animal", ya que un perro es un animal.

Usa la herencia para modelar relaciones de generalización/especialización.