Herencia en Java

La herencia en Java es un mecanismo en el que un objeto adquiere todas las propiedades y comportamientos de un objeto padre. Es una parte importante de los OOP (sistema de programación orientado a objetos).

La idea detrás de la herencia en Java es que puede crear nuevas clases basadas en las clases existentes. Cuando hereda de una clase existente, puede reutilizar métodos y campos de la clase principal. Además, también puede agregar nuevos métodos y campos en su clase actual.

La herencia representa la relación **IS-A,** que también se conoce como relación *padre-hijo* .

¿Por qué usar la herencia en Java?

- Para la sobreescritura del método (por lo que se puede lograr el polimorfismo en tiempo de ejecución).
- Para la reutilización del código.

Términos utilizados en Herencia

- **Clase:** una clase es un grupo de objetos que tienen propiedades comunes. Es una plantilla o plano a partir del cual se crean los objetos.
- **Subclase / Clase secundaria:** Subclase es una clase que hereda la otra clase. También se denomina clase derivada, clase extendida o clase secundaria.
- **Super Class / Parent Class:** Superclass es la clase de donde una subclase hereda las características. También se llama clase base o clase padre.
- **Reusabilidad:** como su nombre lo especifica, la reusabilidad es un mecanismo que le facilita reutilizar los campos y métodos de la clase existente cuando crea una nueva clase. Puede usar los mismos campos y métodos ya definidos en la clase anterior.

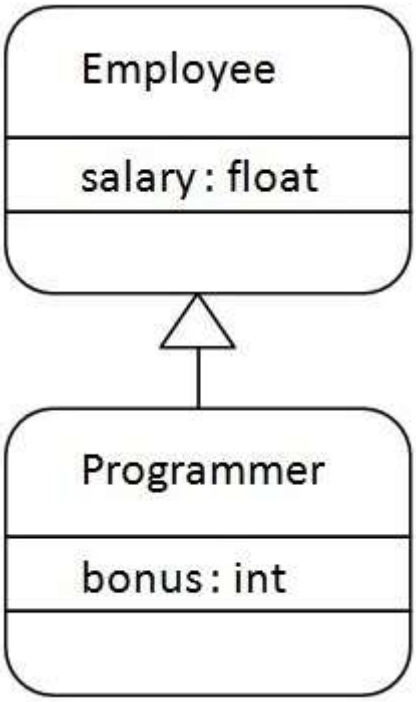
La sintaxis de la herencia de Java

```
clase  Subclase-nombre extends Superclase-nombre
{
   // métodos y campos
}
```

La **palabra clave extend** indica que está creando una nueva clase que deriva de una clase existente. El significado de "se extiende" es aumentar la funcionalidad.

En la terminología de Java, una clase que se hereda se llama padre o superclase, y la nueva clase se llama hija o subclase.

Ejemplo de herencia de Java

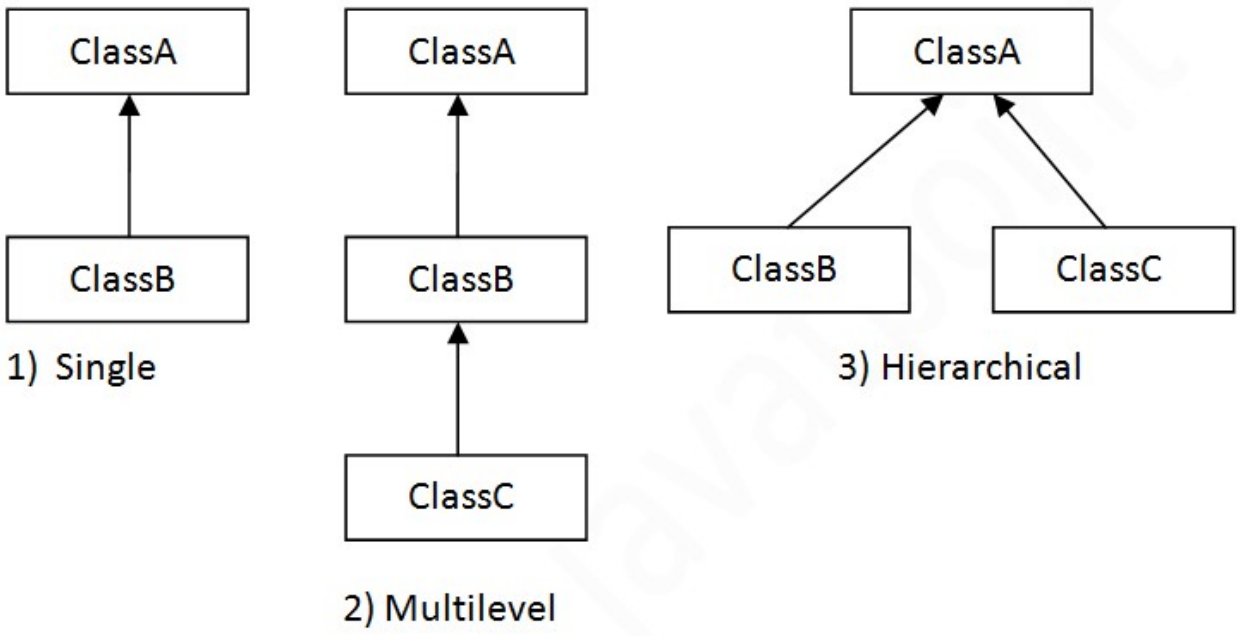


Como se muestra en la figura anterior, Programador es la subclase y Empleado es la superclase. La relación entre las dos clases es **Programador IS-A Empleado** . Significa que el programador es un tipo de empleado.

Tipos de herencia en java

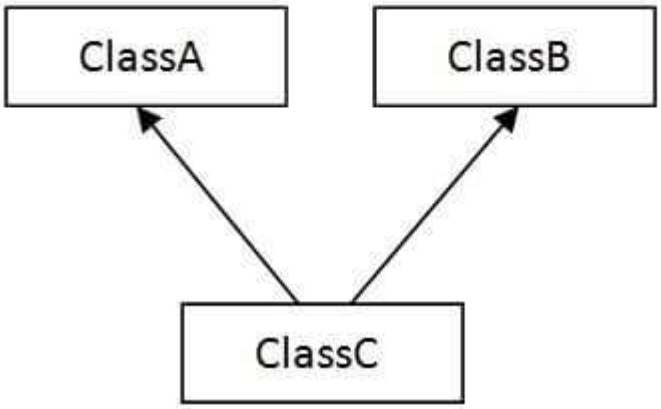
Sobre la base de la clase, puede haber tres tipos de herencia en Java: individual, multinivel y jerárquico.

En la programación de Java, la herencia múltiple e híbrida solo se admite a través de la interfaz. Aprenderemos sobre las interfaces más adelante.

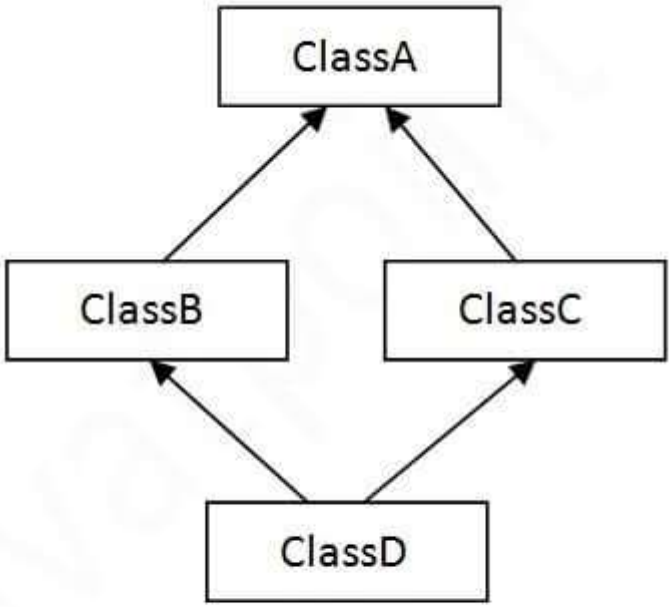


Nota: La herencia múltiple no es compatible en Java.

Cuando una clase hereda varias clases, se conoce como herencia múltiple. Por ejemplo:



4) Multiple



5) Hybrid

.

P) ¿Por qué no se admite la herencia múltiple en Java?

Para reducir la complejidad y simplificar el lenguaje, no se admite la herencia múltiple en Java.

Considere un escenario donde A, B y C son tres clases. La clase C hereda las clases A y B. Si las clases A y B tienen el mismo método y lo llama desde un objeto de clase hijo, habrá ambigüedad para llamar al método de clase A o B.

Dado que los errores en tiempo de compilación son mejores que los errores en tiempo de ejecución, Java genera un error en tiempo de compilación si hereda 2 clases. Entonces, ya sea que tenga el mismo método o diferente, habrá un error de tiempo de compilación.