Comodínes en Generics

El ? (signo de interrogación) representa el elemento comodín. Significa cualquier tipo. Si escribimos <?extiende Número>, significa cualquier clase secundaria de Número, por ejemplo, Entero, Flotante y doble. Ahora podemos llamar al método de la clase Number a través de cualquier objeto de clase hijo.

Podemos usar un comodín como **tipo de parámetro, campo, tipo de devolución o variable local.Sin embargo, no está permitido usar un comodín como argumento de tipo para la invocación de un método genérico, la creación de una instancia de clase genérica o un supertipo**

Comodínes con límite superior

El propósito de los comodines de límite superior es disminuir las restricciones en una variable. Restringe el tipo desconocido para que sea un tipo específico o un subtipo de ese tipo. Se utiliza declarando un carácter comodín ("?") seguido de la palabra clave extends (en el caso de clase) o implements (en el caso de interfaz), seguida de su límite superior.

Sintaxis

Lista<? **extends Number**>
Aquí, **?** es un carácter comodín.

extends , es una palabra clave.

Número , es una clase presente en el paquete java.lang

Supongamos que queremos escribir el método para la lista de Número y sus subtipos (como Entero, Doble). Usando

List<? extends Number> es adecuado para una lista de tipo Number o cualquiera de sus subclases, mientras que

List<Number> funciona solo con la lista de tipo Number.

Entonces,**List<? extends Number>** es menos restrictivo que **List<Number>**.

Comodínes sin límite en Generics

El tipo comodín ilimitado representa la lista de un tipo desconocido como List<?>. Este enfoque puede ser útil en los siguientes escenarios: -

- Cuando el método dado se implementa mediante el uso de la funcionalidad proporcionada en la clase Object.
- Cuando la clase genérica contiene los métodos que no dependen del parámetro de tipo.

Comodínes de límite inferior Generics

El propósito de los comodines de límite inferior es restringir el tipo desconocido para que sea un tipo específico o un supertipo de ese tipo. Se usa declarando un carácter comodín ("?") seguido de la palabra clave super, seguida de su límite inferior.

Sintaxis

Lista<? su**per Integer**>
? es un carácter comodín.

súper, palabra clave.

Integer, es una clase contenedora.

Supongamos que queremos escribir el método para la lista de Entero y su supertipo (como Number, Object). Usando

List<? super Integer> es adecuado para una lista de tipo Integer o cualquiera de sus superclases, mientras que

List<Integer> funciona solo con la lista de tipo Integer. Entonces, **Lista<? super Integer>** es menos restrictivo que **List<Integer>** .