Los datos genéricos son valores o tipos de datos que no están específicamente definidos y que pueden ser aplicables a diferentes tipos de datos o escenarios. En programación, el término "genérico" se refiere a la capacidad de escribir código que puede manejar datos de diferentes tipos sin necesidad de especificar el tipo exacto por adelantado.

Por ejemplo, en lenguajes como Java o C#, puedes crear clases, métodos o estructuras de datos genéricos que pueden funcionar con cualquier tipo de dato. Esto es útil porque permite reutilizar el código sin tener que duplicarlo para diferentes tipos de datos.

1. **Tipo genérico único (T, U, etc.):**

* Un tipo genérico simple que representa un solo tipo de dato. Se suele denotar con letras como T, U, E, K, V, etc.
* Ejemplo: public class Caja<T> { ... } donde T puede ser cualquier tipo de dato (como Integer, String, etc.).

1. **Parámetros de tipo múltiple (T, U, etc.):**

* Se pueden usar múltiples tipos genéricos en una misma clase o método.
* Ejemplo: public class Par<K, V> { ... } donde K y V pueden ser de cualquier tipo.

1. **Genéricos con límites (T extends ClassName):**

* Permiten restringir el tipo genérico a ser una subclase de una clase específica o implementar una interfaz específica.
* Ejemplo: public <T extends Number> T suma(T a, T b) solo permite que T sea un tipo que extienda Number, como Integer, Double, etc.

1. **Wildcards (?):**

* Representan un tipo desconocido dentro de un genérico. Se usa comúnmente en listas o colecciones cuando el tipo exacto no es relevante.
* Ejemplo: List<?> listaDesconocida indica que la lista puede contener cualquier tipo de datos.

1. **Wildcards con límites (? extends Type y ? super Type):**

* ? extends Type: Indica que el tipo desconocido debe ser un subtipo de Type.
* ? super Type: Indica que el tipo desconocido debe ser un supertipo de Type.
* Ejemplo: List<? extends Number> numeros permite que numeros sea una lista de cualquier tipo que extienda Number, como Integer o Double.

1. **Tipos genéricos en métodos:**

* Se pueden declarar genéricos en los métodos, no solo en las clases.
* Ejemplo: public static <T> void imprimir(T elemento) donde T es un tipo genérico usado solo dentro del método.

Ejemplo:

|  |
| --- |
| // Clase genérica que puede aceptar cualquier tipo de dato T  public class Caja<T> {      private T contenido;      public void guardar(T objeto) {          contenido = objeto;      }      public T obtener() {          return contenido;      }  } |

En este ejemplo, Caja es una clase genérica que puede almacenar y devolver cualquier tipo de dato, ya sea un Integer, String, Double, etc. Cuando creas una instancia de Caja, puedes especificar el tipo de dato que va a contener:

|  |
| --- |
| Caja<Integer> cajaEntero = new Caja<>();  cajaEntero.guardar(123);  Caja<String> cajaCadena = new Caja<>();  cajaCadena.guardar("Hola"); |

Los datos genéricos permiten crear soluciones más flexibles y reutilizables en diferentes contextos, sin atarse a un tipo de dato específico.