UNIVERSIDAD TECNOLÓGICA DEL PERÚ

Facultad de Ingeniería

Programacion Orientada a Objetos

Sesión 12: Clases genéricas

Recordando:

¿Que vimos la clase pasada?

Universidad Tecnológica del Perú

Logro de aprendizaje

Al finalizar la sesión, el estudiante soluciona problemas aplicando las clases genéricas usando java en la resolución de ejercicios.

Universidad Tecnológica del Perú

Utilidad

- Las clases genéricas son una herramienta fundamental en la programación orientada a objetos y ofrecen múltiples beneficios para los estudiantes de ingeniería de sistemas.
- Para los estudiantes, el conocimiento y uso de clases genéricas es esencial. Les permite escribir código más limpio, eficiente y seguro. Además, fomenta buenas prácticas de programación que son cruciales para el desarrollo de software a gran escala y la gestión de sistemas complejos.

Agenda

- Introducción a las Clases Genéricas.
- Ventajas de usar clases genéricas.
- Sintaxis de Clases Genéricas en Java.
- Métodos Genéricos.

Universidad Tecnológica del Perú

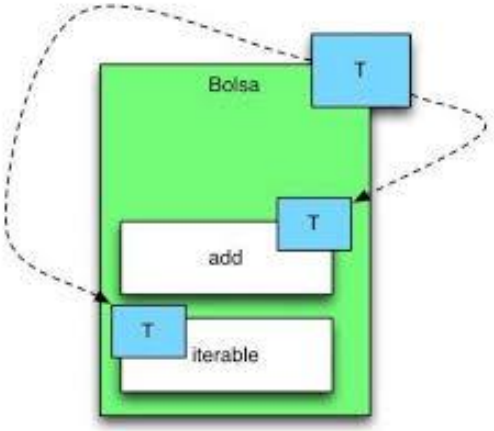
Introducción a las Clases Genéricas

Definición y Propósito:

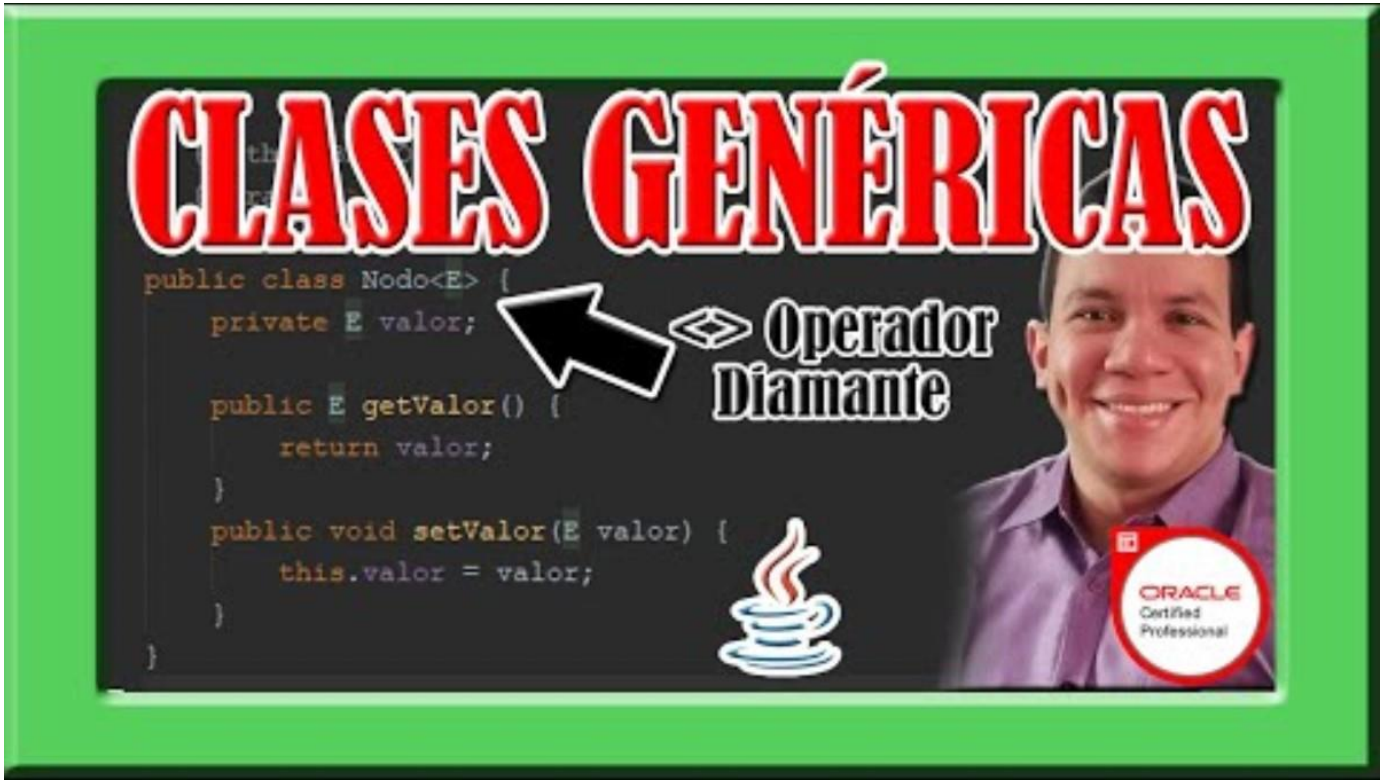
Las clases genéricas en Java son una característica que permite la creación de clases, interfaces y métodos que operan con tipos de datos específicos que se definen en el momento en que se utilizan, en lugar de utilizar tipos de datos concretos.
En otras palabras, te permiten escribir código que es reutilizable y flexible en términos de tipos de datos.

Introducción a las Clases Genéricas

El propósito principal de las clases genéricas es aumentar la reutilización del código y mejorar la seguridad de tipos en tiempo de compilación. En lugar de crear múltiples versiones de una clase o método para manejar diferentes tipos de datos, puedes usar una única clase genérica que funcione con diversos tipos.



Clases Genéricas



https://www.youtube.com/watch?v=BJw0jZ1zoj8

Universidad Tecnológica del Perú

Ventajas de usar clases genéricas

- **Reutilización de código:** Las clases genéricas permiten escribir código genérico que se puede usar con múltiples tipos de datos, lo que reduce la duplicación de código.
- **Seguridad de tipos:** Los errores de tipo se detectan en tiempo de compilación en lugar de en tiempo de ejecución, lo que hace que el código sea más seguro y menos propenso a errores.

Ventajas de usar clases genéricas

- **Mejor legibilidad:** El uso de clases genéricas puede hacer que el código sea más legible, ya que se entiende más fácilmente qué tipo de datos se espera en diferentes partes del código.
- **Flexibilidad:** Las clases genéricas son flexibles y se pueden utilizar con una amplia variedad de tipos de datos, lo que facilita la adaptación del código a diferentes necesidades.
- **Rendimiento:** Aunque el rendimiento puede variar según la implementación, en general, el uso de clases genéricas no tiene un impacto negativo significativo en el rendimiento de la aplicación.

Clases Genéricas

https://www.youtube.com/watch?v=GKJl-4oNUWg

Sintaxis de Clases Genéricas en Java

Declaración de una clase genérica:

En Java, la sintaxis para declarar una clase genérica implica el uso de parámetros de tipo (type parameters) dentro de los signos de mayor y menor ("< >"). Aquí tienes un ejemplo de cómo se declara una clase genérica:

Sintaxis de Clases Genéricas en Java

```
public class
MiClaseGenerica<T>{
// Código de la clase
Genérica
}
```

MiClaseGenerica es el nombre de la clase genérica.
<T> es el parámetro de tipo que representa un tipo genérico. Puedes usar cualquier letra o palabra como nombre del parámetro de tipo,pero comúnmente se utiliza T para representar "Tipo".

Uso de parámetros de tipo

Una vez que has declarado una clase genérica, puedes utilizar el parámetro de tipo T en diferentes partes de la clase, como en atributos, métodos y constructores. Por ejemplo:

```
public class MiClaseGenerica<T>{
    private T dato; // Atributo de tipo genérico
    public MiClaseGenerica(T dato) { // Constructor que toma un parámetro genérico
        this.dato = dato;
    }
    public T getDato() { // Método que devuelve un valor de tipo genérico
        return dato;
    }
}
```

Ejemplos de código

Ahora, veamos ejemplos de cómo se puede utilizar esta clase genérica:

```
// Creación de una instancia de  MiClaseGenerica con un tipo específico (por ejemplo, Integer)
MiClaseGenerica<Integer> instancia1 = new
MiClaseGenerica<>(10);
// Obtener el valor almacenado en la instancia
Integer valor1 = instancia1.getDato();
// Creación de otra instancia con un tipo diferente (por ejemplo, String)
MiClaseGenerica<String> instancia2 = new
MiClaseGenerica<>("Hola");
// Obtener el valor almacenado en la segunda instancia
String valor2 = instancia2.getDato();
```

Ejemplos de código

En este ejemplo, hemos creado dos instancias de MiClaseGenerica, una con un tipo Integer y otra con un tipo String.
La flexibilidad de las clases genéricas permite trabajar con diferentes tipos de datos utilizando la misma estructura de clase.

Métodos Genéricos

Los métodos genéricos en Java son métodos que pueden trabajar con tipos de datos genéricos, lo que significa que pueden aceptar y devolver valores de diferentes tipos sin especificarlos de antemano.
Esto proporciona flexibilidad y reutilización de código al igual que las clases genéricas.

Universidad Tecnológica del Perú

Declaración de métodos genéricos

Para declarar un método genérico, utilizamos parámetros de tipo (type parameters) en la firma del método, similar a cómo lo hacemos en las clases genéricas. Aquí hay un ejemplo de cómo se declara un método genérico:

En este ejemplo:

<T> es un parámetro de tipo que representa un tipo genérico.

T se utiliza para el tipo del parámetro de entrada (valor) y el tipo de retorno del método.

```
public <T> T miMetodoGenerico(T valor) {
// Código del método
return valor;
}
```

Restricciones y límites en tipos genéricos

Puedes aplicar restricciones a los tipos genéricos para limitar qué tipos de datos se pueden utilizar en el método genérico.
Las restricciones se aplican utilizando la palabra clave extends y se pueden usar para asegurarse de que los tipos cumplan con ciertos criterios. Por ejemplo:

```
public <T extends Number> double promedio(T[] valores) {
    // Código para calcular el promedio de los valores
}
```

En este caso, el método promedio solo aceptará tipos que extiendan la clase Number, como Integer, Double, etc.

Ejemplos de métodos genéricos

Veamos algunos ejemplos de métodos genéricos en acción:

```
// Método genérico para imprimir cualquier tipo de array
public <T> void imprimirArray(T[] array) {
    for (T elemento : array) {
        System.out.print(elemento + " ");
    }
    System.out.println();
}
// Método genérico para encontrar el máximo entre dos valores comparables
public <T extends Comparable<T>> T encontrarMaximo(T a, T b) {
    if (a.compareTo(b) > 0) {
        return a;
    } else {
        return b;
    }
}
```

Ejemplos de métodos genéricos

En el primer ejemplo, el método imprimirArray puede imprimir cualquier tipo de array.
En el segundo ejemplo, el método encontrarMaximo encuentra el máximo entre dos valores comparables. Las restricciones en este caso aseguran que los valores sean comparables.

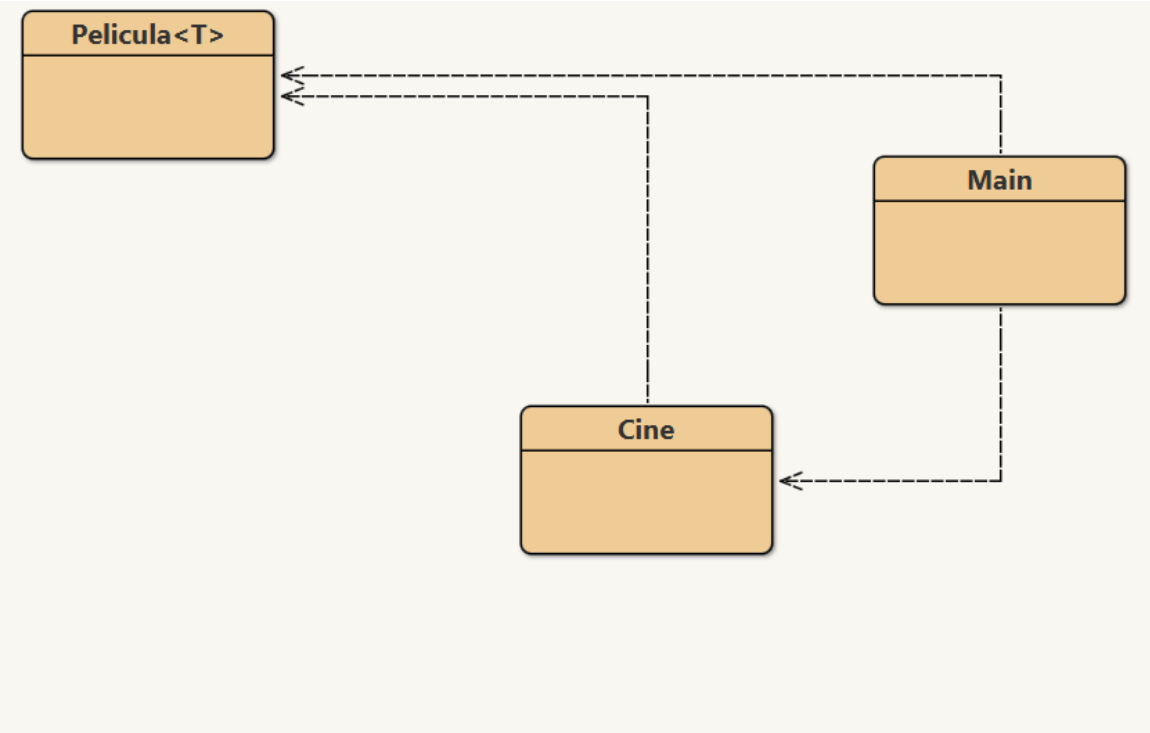
Uso de métodos genéricos

Los métodos genéricos se utilizan de la siguiente manera:

Los métodos genéricos en Java permiten escribir funciones que trabajan con tipos de datos genéricos, lo que proporciona flexibilidad y reutilización de código.
Puedes aplicar restricciones para limitar qué tipos de datos se pueden utilizar en el método genérico, lo que aumenta la seguridad de tipos.
Estos métodos son una parte fundamental de la programación genérica en Java y son ampliamente utilizados en bibliotecas y aplicaciones Java.

```
Integer[] numeros = { 1, 2, 3, 4, 5 };
imprimirArray(numeros);
Double maximo = encontrarMaximo(3.5, 1.2);
```

Ejercicio: Clase genérica llamada “Película”



Ejercicio: Clase genérica llamada “Película”

```
public class Pelicula<T> {
    private String titulo;
    private T genero;
    public Pelicula(String titulo, T genero) {
        this.titulo = titulo;
        this.genero = genero;
    }
    public String getTitulo() {
        return titulo;
    }
    public T getGenero() {
        return genero;
    }
}
```

Ejercicio: clase genérica llamada “Cine”

```
class Cine {
    public static <T> void
imprimirDetallesPelicula(Pelicula<T> pelicula) {
        System.out.println("Título: " + pelicula.getTitulo());
        System.out.println("Género: " + pelicula.getGenero());
        System.out.println();
    }
}
```

Wildcards y Captura de Tipos

En este ejemplo, hemos utilizado una colección genérica List<Pelicula<?>> para almacenar diferentes tipos de películas en una lista.

```
import java.util.ArrayList;
import java.util.List;
public class Main {
    public static void main(String[] args) {
        List<Pelicula<?>> listaDePeliculas = new ArrayList<>();
        Pelicula<String> pelicula1 = new Pelicula<>("Interestelar", "Ciencia Ficción");
        Pelicula<String> pelicula2 = new Pelicula<>("El Señor de los Anillos","Fantasía");
        Pelicula<Integer> pelicula3 = new Pelicula<>("Matrix", 2); // Género representado como número

        listaDePeliculas.add(pelicula1);
        listaDePeliculas.add(pelicula2);
        listaDePeliculas.add(pelicula3);

        for (Pelicula<?> pelicula : listaDePeliculas) {
            Cine.imprimirDetallesPelicula(pelicula);
        }
    }
}
```

Practica

Implementa el ejercicio visto.

Conclusiones

- Las clases genéricas en Java son una herramienta poderosa para escribir código más reutilizable y seguro en términos de tipos.
- Permiten crear clases y métodos que funcionan con diferentes tipos de datos sin sacrificar la seguridad en tiempo de compilación.

Cierre

¿Qué hemos aprendido hoy?

Bibliografía

- MORENO PÉREZ, J. “Programación orientada a objetos”. RA-MA Editorial. https://tubiblioteca.utp.edu.pe/cgi-bin/koha/opac-detail.pl?biblionumber=31933
- Vélez Serrano, José. “Diseñar y programar, todo es empezar: una introducción a la Programación Orientada a Objetos usando UML y Java”. Dykinson. https://tubiblioteca.utp.edu.pe/cgi-bin/koha/opac-detail.pl?biblionumber=36368

Universidad Tecnológica del Perú