UNIVERSIDAD TECNOLÓGICA DEL PERÚ

Facultad de Ingeniería

Programación Orientada a Objetos

Sesión 13: Introducción a las Expresiones Lambda

Recordando:

¿Que vimos la clase pasada?

Universidad Tecnológica del Perú

Logro de aprendizaje

Al finalizar la sesión, el estudiante soluciona problemas aplicando expresiones lambda usando Java en la resolución de ejercicios.

Universidad Tecnológica del Perú

Utilidad

Las expresiones lambda en Java son una herramienta poderosa que no solo mejora la productividad y claridad del código, sino que también prepara a los estudiantes para trabajar en entornos de desarrollo modernos y avanzados. Dominar esta característica es esencial para cualquier estudiante de sistemas que desee construir una carrera sólida en el desarrollo de software.

Agenda

- Introducción a las Expresiones Lambda.
- Expresiones Lambda en Programación Orientada a Objetos.
- Uso de Expresiones Lambda como Argumento de Métodos

Universidad Tecnológica del Perú

Introducción a las Expresiones Lambda

Definición :

Las expresiones lambda, también conocidas como funciones lambda o funciones anónimas, son una característica de muchos lenguajes de programación que permite definir funciones pequeñas y anónimas in situ. Esto significa que puedes crear funciones sin asignarles un nombre y usarlas directamente en el lugar donde las necesites. Las expresiones lambda son especialmente útiles cuando necesitas una función simple para realizar una tarea específica, como aplicar una operación a una colección de datos.

Introducción a las Expresiones Lambda

Definición :

Por lo general, una expresión lambda consta de los siguientes elementos:

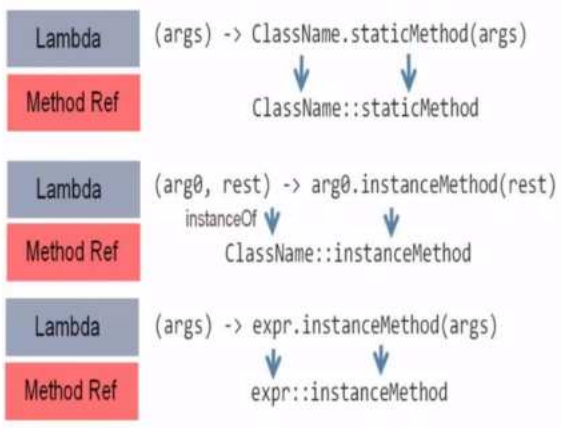
- **Parámetros:** Los valores de entrada que la función lambda acepta.
- **Operador Arrow (->):** Se utiliza para separar los parámetros del cuerpo de la función lambda.
- **Cuerpo de la Función:** El código que se ejecutará cuando se llame a la función lambda.

Ventajas de Usar Expresiones Lambda en Programación

Las expresiones lambda proporcionan varias ventajas en programación:

- **Concisión del Código:**

Las expresiones lambda permiten definir funciones pequeñas de manera concisa en una sola línea de código, lo que mejora la legibilidad y reduce la cantidad de código redundante.



Universidad Tecnológica del Perú

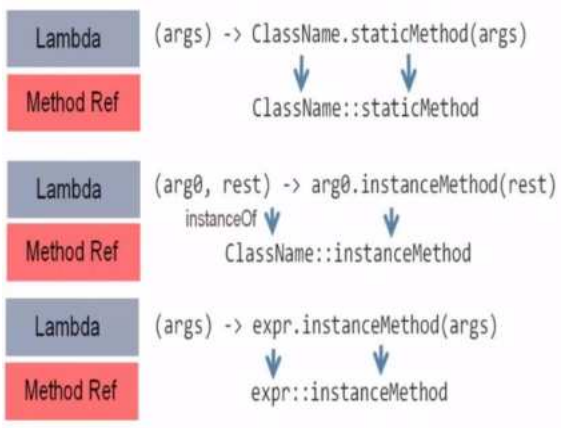
Ventajas de Usar Expresiones Lambda en Programación

- **Programación Funcional:**

Facilitan la adopción de conceptos de programación funcional al permitir operaciones como el mapeo, filtrado y reducción de colecciones de datos de manera más intuitiva y elegante.

- **Mayor Abstracción:**

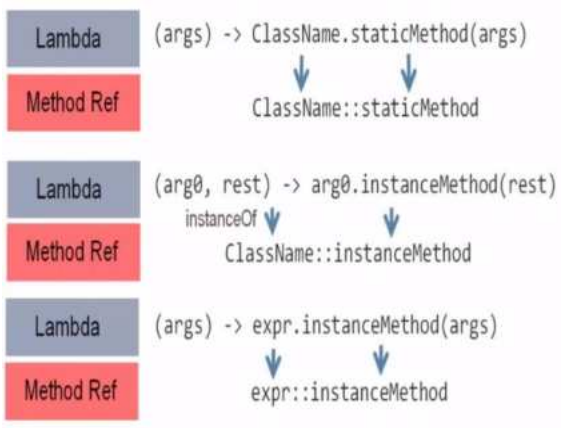
Puedes expresar la intención de tu código de manera más clara y abstracta al usar expresiones lambda.



Universidad Tecnológica del Perú

Ventajas de Usar Expresiones Lambda en Programación

- **Mejora la Productividad:** Al evitar la necesidad de definir funciones separadas con nombres, las expresiones lambda ahorran tiempo y aumentan la productividad.
- **Flexibilidad:** Puedes utilizar expresiones lambda en una variedad de situaciones, como pasándolas como argumentos a funciones u ordenándolas.



Universidad Tecnológica del Perú

Expresiones Lambda en Programación Orientada a Objetos

Cómo se Aplican las Expresiones Lambda en un Entorno Orientado a Objetos:

En un contexto de Programación Orientada a Objetos (POO), las expresiones lambda se utilizan para definir funciones anónimas que pueden ser aplicadas a objetos y colecciones de objetos. A continuación, se detallan los aspectos clave de su aplicación:

Expresiones Lambda en Programación Orientada a Objetos

- **Métodos de Orden Superior:**
  Las expresiones lambda se usan en POO como ejemplos de métodos de orden superior, es decir, funciones que pueden recibir otras funciones como argumentos o devolver funciones como resultado. Esto permite realizar operaciones avanzadas en colecciones de objetos, como filtrado, mapeo y reducción, de manera más eficiente y legible.

Expresiones Lambda en Programación Orientada a Objetos

- **Iteración Mejorada:**
  Las expresiones lambda son especialmente útiles cuando necesitas iterar sobre una colección de objetos y aplicar una operación o función a cada elemento de manera sencilla.
  En lugar de escribir bucles explícitos, puedes usar funciones como map, filter, y reduce junto con expresiones lambda para lograr una iteración más elegante y eficiente.

Expresiones Lambda en Programación Orientada a Objetos

- **Mayor Abstracción:**

  Las expresiones lambda permiten abstraer las operaciones que se aplican a los objetos, lo que hace que el código sea más modular y mantenible.

  Esto significa que puedes cambiar la lógica de una operación sin necesidad de modificar todo el código que la utiliza.

Ejemplos de Uso de Expresiones Lambda en POO

Mapeo de una Lista:

Puedes usar una expresión lambda para mapear una lista de objetos a otra lista, aplicando una transformación a cada elemento. Por ejemplo, si tienes una lista de objetos Persona, puedes utilizar una expresión lambda para mapearla a una lista de sus nombres.

Ejemplos de Uso de Expresiones Lambda en POO

Filtrado de una Lista:

Las expresiones lambda son útiles para filtrar elementos de una lista según un criterio específico.

Por ejemplo, puedes filtrar una lista de Producto para obtener solo los productos que tienen un precio superior a cierto valor.

Ejemplos de Uso de Expresiones Lambda en POO

Reducción de una Lista:
Puedes utilizar expresiones lambda en la reducción de una lista para calcular un valor agregado, como la suma de los valores de una lista de números.

Ejemplos de Uso de Expresiones Lambda en POO

Ordenamiento de una Lista:

Las expresiones lambda también se utilizan para definir un criterio de ordenamiento personalizado cuando necesitas ordenar una lista de objetos de acuerdo a un atributo específico.

ICODEAP

{Learn Code Java}

🚀Aprendiendo Expresiones Lambda en Java con Ejemplos

Elivar Largo
Java Backend Developer

www.icodeap.com

https://www.youtube.com/watch?v=ET8K3NIyeM8

Universidad Tecnológica del Perú

Uso de Expresiones Lambda como Argumento de Métodos

En programación, es común que los métodos reciban argumentos, que son valores que se utilizan como entrada para realizar alguna acción dentro del método.
Una de las características más potentes y versátiles de las expresiones lambda es la capacidad de ser pasadas como argumentos a otros métodos.
Esto permite que las expresiones lambda se utilicen para personalizar y parametrizar el comportamiento de un método de manera flexible. A continuación, se describen los conceptos clave de cómo las expresiones lambda pueden ser pasadas como argumentos a métodos:

Uso de Expresiones Lambda como Argumento de Métodos:

Métodos de Orden Superior:

- En programación, los métodos que pueden aceptar funciones (expresiones lambda) como argumentos se denominan "métodos de orden superior."
- Estos métodos se pueden utilizar para aplicar una función específica en una variedad de contextos, lo que los hace muy flexibles y reutilizables.

Uso de Expresiones Lambda como Argumento de Métodos:

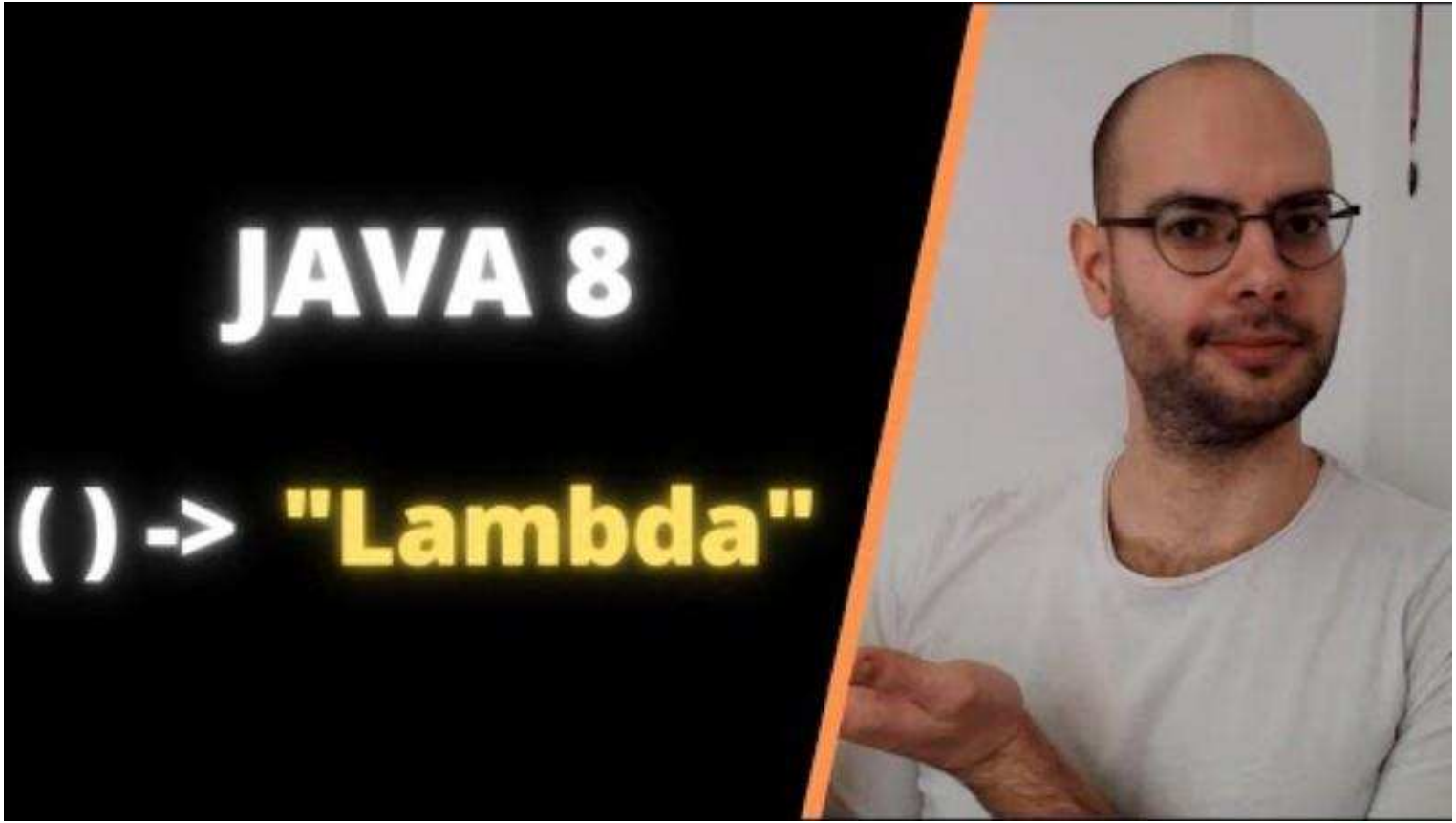
Interfaz Funcional:

- En muchos lenguajes de programación, como Java, se utilizan interfaces funcionales para definir la firma de métodos que aceptan expresiones lambda como argumentos.
- Estas interfaces funcionales contienen un solo método abstracto que debe ser implementado por la expresión lambda. Ejemplos de interfaces funcionales incluyen Predicate, Consumer, Function, entre otros.

Uso de Expresiones Lambda como Argumento de Métodos:

Personalización del Comportamiento:

- Pasar expresiones lambda como argumentos permite personalizar el comportamiento de un método sin tener que modificar su implementación.
- Esto es especialmente útil en situaciones donde deseas realizar operaciones específicas en una colección de datos o aplicar lógica particular a un proceso.

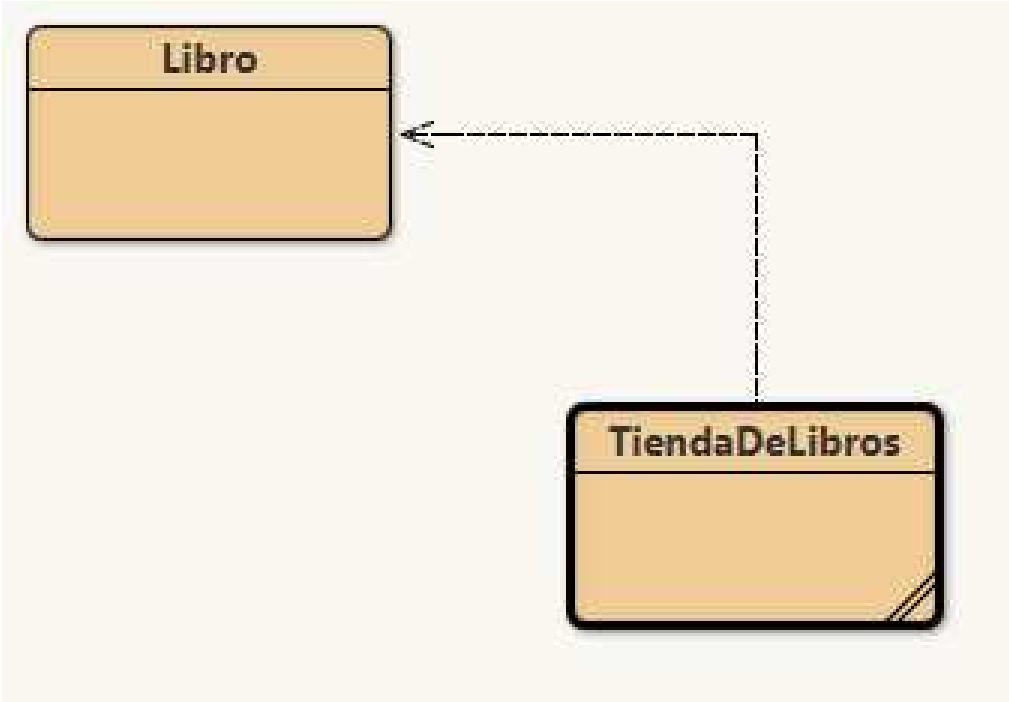


https://www.youtube.com/watch?v=QrptTiBP5yk

Ejercicio:

- Ejercicio simple utilizando un método de reducción en NetBeans para una tienda de venta de libros.
- Supongamos que tienes una lista de libros con información sobre su precio y deseas utilizar el método de reducción para calcular el precio total de todos los libros en el inventario de la tienda.

Ejercicio:



```
public class Libro {
    private String titulo;
    private String autor;
    private double precio;

    // Constructor
    public Libro(String titulo, String autor, double precio) {
        this.titulo = titulo;
        this.autor = autor;
        this.precio = precio;
    }

    // Métodos getters
    public String getTitulo() {
        return titulo;
    }

    public String getAutor() {
        return autor;
    }

    public double getPrecio() {
        return precio;
    }
```

Ejercicio: class Libro

Universidad Tecnológica del Perú

Ejercicio: class Libro

```
// Métodos setters (opcional, solo si necesitas cambiar los valores después de la creación)
    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }

    public void setAutor(String autor) {
        this.autor = autor;
    }

    public void setPrecio(double precio) {
        this.precio = precio;
    }
}
```

Ejercicio:

Crearemos una lista de libros y utilizaremos una expresión lambda y el método de reducción reduce para calcular el precio total de todos los libros en el inventario.

```
import java.util.ArrayList;
import java.util.List;
public class TiendaDeLibros {
    public static void main(String[] args) {
        // Crear una lista de libros
        List<Libro> inventario = new ArrayList<>();
        inventario.add(new Libro("El Gran Gatsby", "F. Scott Fitzgerald", 15.99));
        inventario.add(new Libro("Cien Años de Soledad", "Gabriel García Márquez", 19.99));
        inventario.add(new Libro("To Kill a Mockingbird", "Harper Lee", 12.99));
```

Ejercicio:

```
// Utilizar una expresión lambda y el método reduce para calcular el precio total
        double precioTotal = inventario.stream()
            .map(Libro::getPrecio) // Extraer los precios de los libros
            .reduce(0.0, (subtotal, precio) -> subtotal + precio); // Sumar los precios

        // Imprimir el precio total de los libros en el inventario
        System.out.println("Precio total de los libros en el inventario: $" + precioTotal);
    }
}
```

Explicación del Ejercicio

Explicación paso a paso:

1. `inventario.stream()`:

   - `inventario` es una colección (por ejemplo, una lista) que contiene objetos `Libro`.
   - `stream()` convierte esta colección en un stream, lo cual permite operaciones funcionales sobre los elementos de la colección.

Explicación del Ejercicio

2. `.map(Libro::getPrecio)`:

   - `.map()` es una operación de stream que transforma cada elemento del stream según la función proporcionada.
   - `Libro::getPrecio` es una referencia a método que indica que estamos aplicando el método `getPrecio()` de la clase `Libro` a cada objeto `Libro` del stream.
   - Esto extrae el precio de cada libro del inventario, convirtiendo el stream de libros en un stream de precios (`Stream<Double>`).

Explicación del Ejercicio

3. `.reduce(0.0, (subtotal, precio) -> subtotal + precio)`:

   - `.reduce()` es otra operación de stream que reduce los elementos del stream a un solo valor.
   - `0.0` es el valor inicial del subtotal, en este caso, el precio total inicializado en 0.0.
   - `(subtotal, precio) -> subtotal + precio` es la expresión lambda que define cómo se debe combinar los elementos del stream.
     - `subtotal` es el acumulador parcial que mantiene el resultado parcial de la reducción.
     - `precio` es el próximo elemento del stream que se está combinando con el subtotal.
     - La lambda simplemente suma el subtotal actual con el precio actual para calcular el nuevo subtotal.

Esta operación reduce todos los precios de los libros en el stream a un único valor, que es el precio total de todos los libros en el inventario.

Explicación del Ejercicio

- En este ejercicio, hemos creado una lista de objetos Libro en el inventario de la tienda y luego utilizamos una expresión lambda y el método reduce para sumar los precios de los libros y calcular el precio total.
- La operación reduce toma un valor inicial (en este caso, 0.0) y una función que describe cómo combinar los valores (en este caso, la suma de precios).

Explicación del Ejercicio

- En resumen, esta expresión lambda y el método reduce se utilizan para calcular el precio total de todos los libros en el inventario.
- Primero, se mapean los precios de los libros utilizando **.map(Libro::getPrecio),** y luego se utiliza **.reduce(0.0, (subtotal, precio) -> subtotal + precio)** para sumar todos estos precios y obtener el precio total.
- Este enfoque es eficiente y conciso gracias al uso de expresiones lambda y operaciones de stream en Java.

Conclusiones:

- Hemos explorado cómo las expresiones lambda pueden mejorar la programación orientada a objetos y cómo se pueden utilizar como argumentos de métodos.
- Al comprender estos conceptos, los programadores podrán escribir un código más limpio y eficiente, lo que les permitirá desarrollar aplicaciones de mayor calidad.
- Las expresiones lambda son una herramienta poderosa que puede mejorar
- significativamente la productividad y la capacidad de mantenimiento del código en proyectos de software.

Universidad Tecnológica del Perú

Practica

Implementa el ejercicio visto.

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