Programación Funcional

Lambdas

Lucero Guadalupe Domínguez Pérez

Ejemplos Lambdas

Supplier

01

```java
import java.util.Random;
import java.util.function.Supplier;
public class SupplierNumeroAleatorio {
        public static void main(String[] args) {

            // Definimos la funcion Supplier que ejecuta numeros
entre 1 a 100
            Supplier<Integer> numeroAleatorioSupplier = () -> {
                // Creamos una instancia de Random para crear
numeros aleatorios
                Random random = new Random();
                //Creamos numero aleatorios
                return random.nextInt(100) ;
            };

            // Ejecutamos lambda, para obtener el  número
aleatorio
            int numeroAleatorio = numeroAleatorioSupplier.get();
            System.out.println("Numero aleatorio: " +
numeroAleatorio);

        }
    }
```

Predicate

02

```java
import java.util.Arrays;
import java.util.List;
import java.util.function.Predicate;

public class PredicateImparesPares {

        public static void main(String[] args) {
            List<Integer> numeros = Arrays.asList(1, 2, 3, 4, 5,
6, 7, 8, 9, 10);

            // Definimos la funcion Predicate que verifica si un
numero es par
            Predicate<Integer> esPar = num -> num % 2 == 0;

            // Ejecutamos la funcion Predicate LAMBDA
            System.out.println("Números pares:");
            for (Integer num : numeros) {
                if (esPar.test(num)) {
                    System.out.print(num + " ");
                }
            }
        }
    }
```

BiPrediccate

03

```java
import java.util.function.BiPredicate;
public class BiPredicateSize {
        public static void main(String[] args) {

            // Definimos una funcion BiPredicate que
verifica si una cadena y un numero tienen el mismo
tamaño
            BiPredicate<String, Integer>
mismaLongitud = (cadena, numero) -> cadena.length()
== numero;

            // Ejecutamos la funcion BiPredicate
LAMBDA

 System.out.println(mismaLongitud.test("LUCERO",
6));

 System.out.println(mismaLongitud.test("GUADALUPE",
5));

        }
}
```

Ejemplos Lambdas

Consumer

04

```java
import java.util.Arrays;
import java.util.List;
import java.util.function.Consumer;

public class ConsumerNamesSP {

    public static void main(String[] args) {

        System.out.println("Algunos integrantes de Super Junior: ");

        //Creando un arreglo de nombres
        List<String> nombres = Arrays.asList("Siwon", "Yesung", "Heechul", "Donghae");

        // Definimos el funcional Consumer(LAMBDA)
        Consumer<String> sp = nombre -> System.out.println(nombre);

        // Ejecutamos el funcional Consumer (LAMBDA)
        nombres.forEach(sp);

    }

}
```

BiConsumer

05

```java
import java.util.Arrays;
import java.util.function.BiConsumer;
import java.util.List;
public class BiConsumerSP {
        public static void main(String[] args) {
            // Creamos dos listas de diferentes tipos, String y Integer
            List<String> nombreSP = Arrays.asList("Siwon ", "Yesung ", "Eunhyuk ", "Ryeowook");
            List<Integer> edadeSP = Arrays.asList(35, 37, 35, 34);

            // Definimos el funcional BiConsumer para imprimir
            BiConsumer<String, Integer> imprimirPersona = (nombre, edad) ->
                    System.out.println(nombre + " tiene " + edad + " años.");

            // Ejecutamos el funcional Biconsumer (LAMBDA)
            for (int i = 0; i < nombreSP.size(); i++) {

             imprimirPersona.accept(nombreSP.get(i), edadeSP.get(i));
            }
        }
    }
```

Function

06

```java
import java.util.function.Function;
public class ConvertirCaFFunction {
        public static void main(String[] args) {

            // Definimos la Function que convierte grados Celsius a grados Fahrenheit, usando la formula
            // F = C x 9/5 + 32
            Function<Double, Double> celsiusToFahrenheit = celsius -> (celsius * 9/5) + 32;

            // Ejecutamos la lambda function, para aplicar la conersion
            double celsius = 24.67;
            double fahrenheit = celsiusToFahrenheit.apply(celsius);
            System.out.println(celsius + " grados Celsius son " + fahrenheit + " grados Fahrenheit.");

        }
    }
```

Ejemplos Lambdas

BiFunction

07

```java
import java.util.function.BiFunction;
public class BiFunctionArea {
    public static void main(String[] args) {

        // Definimos la funcion BiFunction para calcular el area del rectangulo
        BiFunction<Integer, Integer, Integer> calculateRectangleArea = (width, height) -> width * height;

        // Definimos las variables de ancho y altura
        int ancho = 5;
        int altura = 10;

        // Ejecutamos la funcion apply de la instancia de BiFunction para calcular el area del retangulo
        int area = calculateRectangleArea.apply(ancho, altura);

        // Imprimimos el resultado
        System.out.println("Area del rectangulo: \nancho " + ancho + " y altura " + altura + "\nes: " + area);
    }

}
```

UnaryOperator

08

```java
import java.util.function.UnaryOperator;
public class UnaryOperatorDuplicar {
    public static void main(String[] args) {

        // Definimos la funcion UnaryOperator
        UnaryOperator<Integer> duplicadoNumero = num -> num * 2;

        // Creamos una variable de referencia
        int numero = 5;

        // Ejecutamos la función apply de la funcion UnaryOperator para obtener el resultado
        int resultado = duplicadoNumero.apply(numero);

        // Imprimimos el rsultado, con el numero antiguo y nuevo
        System.out.println("Numero Original: " + numero);
        System.out.println("Numero duplicado: " + resultado);
    }

}
```

BinaryOperator

09

```java
import java.util.function.BinaryOperator;
public class BinaryOperatorPotencia {
    public static void main(String[] args) {

        // Declaramos la funcion BinaryOperator, que recibe como parametros 2 valores de referencia
        BinaryOperator<Integer> powerFunction = (base, exponente) -> {
            int resultado= 1;
            for (int i = 0; i < exponente; i++) {
                resultado *= base;
            }
            return resultado;
        };

        // Definimos la base y el exponente para calcular la potencia
        // se multiplica por si mismo el numero de veces que tiene el exponente
        int base = 5;
        int exponente = 2;

        // Ejecutamos la funcion apply de BinaryOperator
        int potenciaResultado = powerFunction.apply(base, exponente);

        // Imprimimos por pantalla la base, el exponente y el resultado de la potencia.
        System.out.println("Base: " + base);
        System.out.println("Exponent: " + exponente);
        System.out.println("Power Result: " + potenciaResultado);
    }

}
```