**Java-8 – Primitive interfaces for function**

* **Primitive Function:**

As like Predicate even Function has primitive version. Analogy wise its same as Predicate, if we make use of normal Function for Arithmetic operations, there we will be n-number of autoboxing and auto unboxing.

* **IntFunction:**

import java.util.function.IntFunction;

class IntFunctionDemo{

public static void main(String[] args){

IntFunction<Integer> f = I -> I \* I;

System.out.println(f.apply(10));

}

}

Note: Here the Integer in the declaration defines the output type not the input type parameter.

* **ToIntFunction:**

Let us say we have a requirement to return the length of the given String. Below is the code to do it using normal function.

Function<String, Integer> f = s -> s.length();

System.out.println(f.apply(“saravana”));

The above approach has to do conversion from primitive to Object in the return statement. It will impact the performance. There is an alternative to achieve the above functionality using ToIntFunction interface.

import java.util.function.ToIntFunction;

class ToIntFunctionDemo{

public static void main(String[] args){

ToIntFunction<String> f = s -> s.length();

System.out.println(f.applyAsInt(“Saravana”)); // 8

}

}

* **IntToDouble:**

Requirement: Square root of given integer

With normal function:

Function<Integer, Double> f = I -> Math.sqrt(i);

System.out.println(f.apply(10));

Problem: Performance issue because of conversions.

With IntToDoubleFunction:

import java.util.function.IntToDoubleFunction:

class Test{

public static void main(String[] args){

IntToDouble f = i -> Math.sqrt(i);

System.out.println(f.applyAsDouble(5));

}

}

Note: If we use this primitive functional interfaces it will improve the performance.

* **All 15 primitive functional interfaces for Function:**

1. IntFunction<R>

public R apply(int i);

1. LongFunction<R>

public R apply(long l);

1. DoubleFunction<R>

public R apply(double d);

1. ToIntFunction<T>

public int applyAsInt(T t);

1. ToLongFunction<T>

public long applyAsLong(T t);

1. ToDoubleFunction<T>

public double applyAsDouble(T t);

1. IntToLongFunction

public long applyAsLong(int i);

1. IntToDoubleFunction

public double applyAsDouble(int i);

1. LongToIntFunction

public int applyAsInt(long l);

1. LongToDoubleFunction

public double applyAsDouble(long l);

1. DoubleToIntFunction

public int applyAsInt(double d);

1. DoubleToLongFunction

public long applyAsLong(double d);

1. ToIntBiFunction<T, U>

public int applyAsInt(T t, U u);

1. ToLongBiFunction<T, U>

public long applyAsLong(T t, U u);

1. ToDoubleBiFunction<T, U>

public double applyAsDouble(T t, U u);

* **This document contains contents for videos 69 & 70.**