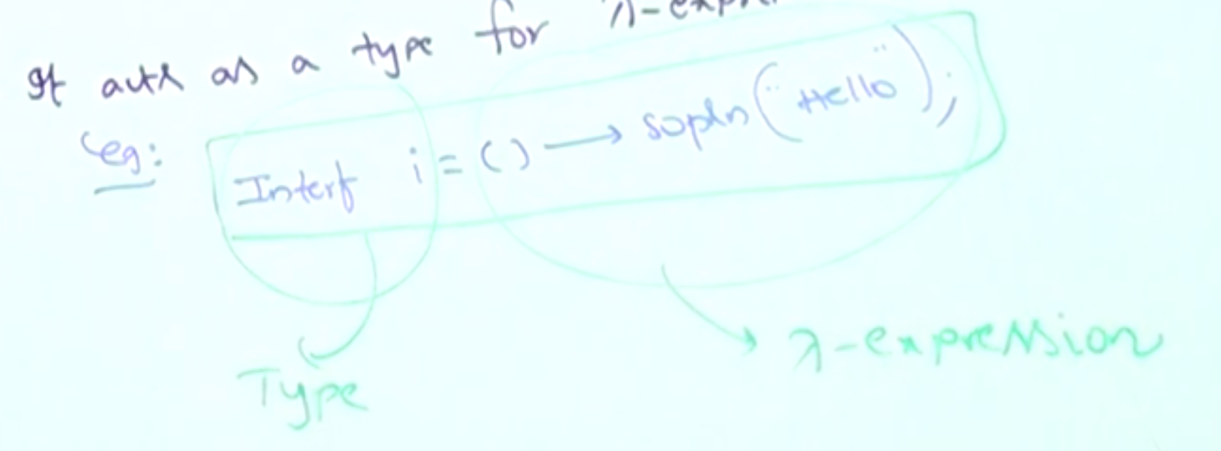
**Java-8 – Invoking Lambda Expression Using Functional Interface:**

* **Functional Interface Summary:**

1. It should contain exactly one abstract method. SAM (Single Abstract Method).
2. It can contain any number of default and static methods.
3. It can act as a type for Lambda Expression.



1. It can be used to invoke lambda expression.

Ex: i.m1();

* **Questions:**

**Case\_01: Why Functional Interface should contain only one abstract method?**

interface Interf{

public void m1(int i);

}

Interf I = i -> System.out.println(i\*i);

I.m1(10);

I.m1(20);

Lambda expression get all the information from the interface method signature. Like i is the parameter, and int is its type, and the expression is the implementation for method m1().

So, if the functional interface contains more than one abstract method, then the compiler can’t get all the above information.

interface Interf{

public void m1(int i);

public void m2(int i);

}

Interf I = i -> System.out.println(i\*i);

In this case we will get the below compile time error.

Incompatible types: Interf is not a functional interface

Multiple, non-overriding abstract methods in

interface Interf

**Case\_02: What is the advantage of @FunctionalInterface annotation?**

Functional Interface must contain only one abstract method, @FunctionalInterface is used to indicate that the interface is Functional Interface, so that the developer will not try to add new method to that interface.