**When to use and Limitation :**

1. When you have functional interface
2. When you want to pass a functionality as an argument
3. You cannot use Lambdas outside functional interface

**WHAT IS FUNCTIONAL INTERFACE???**

Functional interface is an interface with a function. Basically, the idea is that you create a

Interface with a one and only one function in it.

public interface ArithmeticFunctionalInterface {  
 int arithmetic(int a,int b);  
}

Here we have created an interface called ArthmeticFunctionalInterface. It only contains a single function called arithmetic. We can do an arithmetic operation using this interface;

**Method Referencing**

Method referencing is basically a pass through to a pre-defined functional definition.

That is if you have predefined definition then you can simply use method referencing syntax.

Printable p = DemoReferencing::*display*;  
p.printing("Hello");

public static void display(String s){  
 System.*out*.println(s);  
}

DemoReferencing add = new DemoReferencing();  
Addable a = add::add;  
System.*out*.println(a.add(5,5));

public int add(int x,int y){  
 return x+y;  
}

**Inbuilt-Functional Interface**

Java has given us few inbuilt functional interfaces in java.util.function which we can use in order to create implementation. Three major interfaces are :

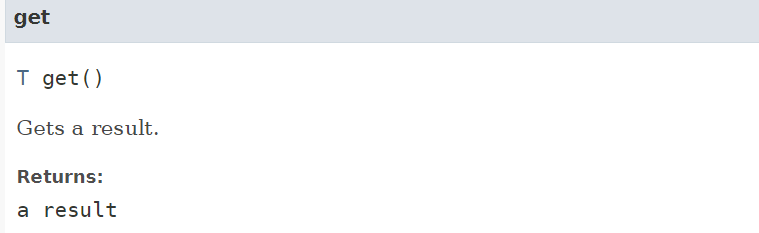
1. **Interface Supplier<T>**

## Interface Consumer<T>

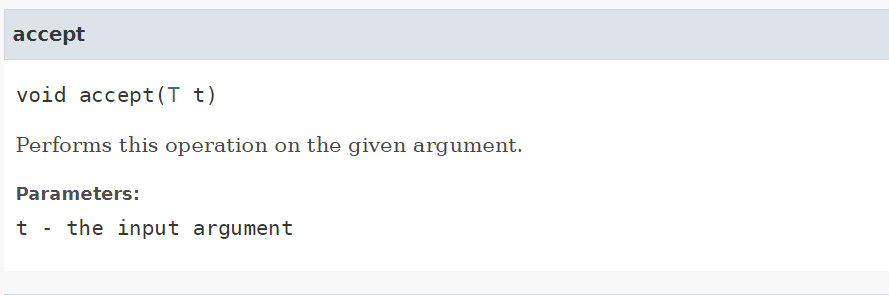
## Interface Predicate<T>

1. Supplier interface takes no argument but returns a value;

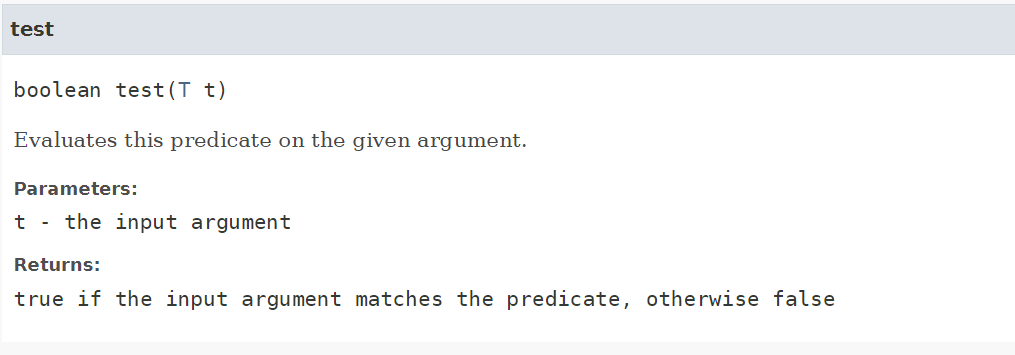
Its functional method is get().



1. Consumer : Consumer takes arguments but returns no value;



1. Predicate : Predicate takes arguments and returns a bool value;



Examples :

Collections.sort() using lambdas :

Collections.sort takes two argument first one is collection and then second is compare interface. So for compare interface we can use lambdas.

// List<Person> people = Arrays.asList(  
// new Person("Charles", "Dickens", 60),  
// new Person("Lewis", "Carrol", 42),  
// new Person("Thomas", "Carlyle", 51),  
// new Person("Charlotte", "Bronte", 45),  
// new Person("Matthew", "Arnold", 38)  
// );

Sol : Collections.sort(people,(p1,p2)-> {  
// return p1.getLastName().compareTo(p2.getLastName());  
// });Collections.sort(people,Comparator.comparingInt(Person::getAge));

Using Lambdas for print values of the list

forEach(System.out::println);

Lambas with Streams examples :

List<String> words = new ArrayList(Arrays.*asList*("Apple","Banana","Cameron","Doge","Etherium"));

1. Add last character of all words into stringbuilder
2. StringBuilder s = new StringBuilder();  
   words.forEach(word-> s.append(word.charAt(1)));
3. Lowercasing the words
4. words.replaceAll(String::toLowerCase);