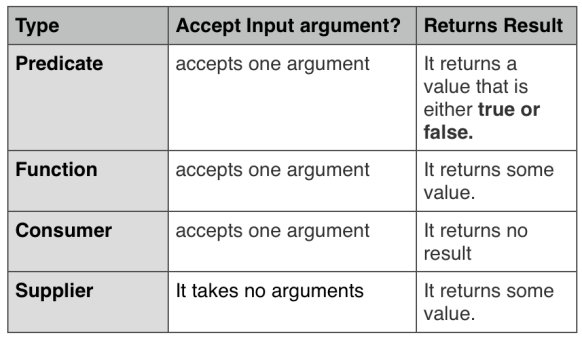
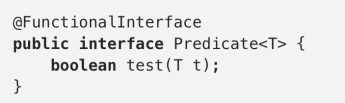
**Functional Interfaces: Predicate, Consumer, Function, and Supplier**

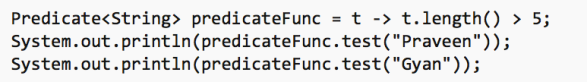


**Predicate**

A predicate is a statement that may be true or false depending on the values of its variables. It can be thought of as a function that returns a value that is either true or false.

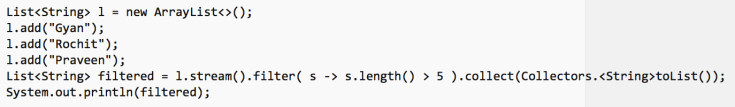


Let us execute the below code:



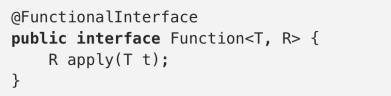
**Output:**  
true  
false

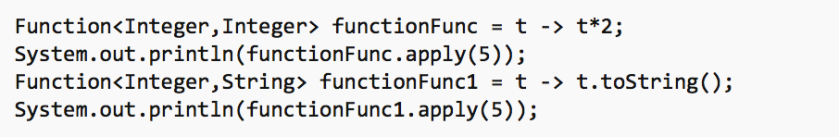
Predicates are also used to filter collections, for example:

  
**Output:**  
[Rochit, Praveen]

#### Function

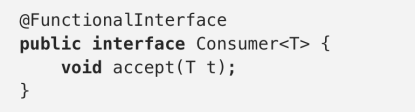
This functional interface represents a function that accepts one argument and produces a result. One use, for example, it’s to convert or transform from one object to another. Since it’s a functional interface, you can pass a lambda expression wherever a Function is expected.  
The input parameter type and the return type of the method can either be same or different.

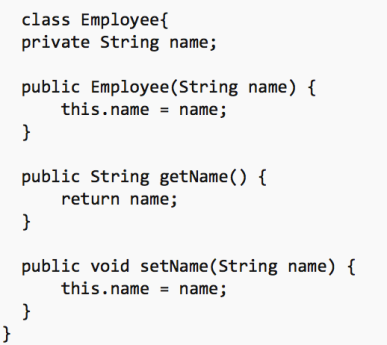


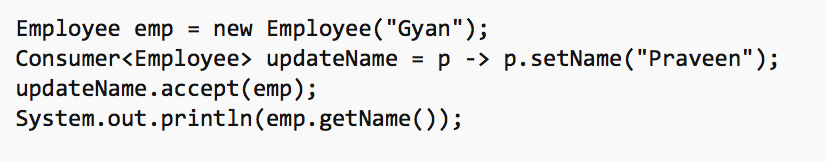
  
**Output**:  
10  
5

#### Consumer

This functional interface represents an operation that accepts a single input argument and returns no result. The real outcome is the side-effects it produces. Since it’s a functional interface, you can pass a lambda expression wherever a Consumer is expected.





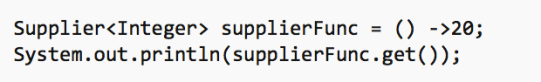


The side-effect here, it’s the updating of the employee’s name, so the output is:  
**Praveen**

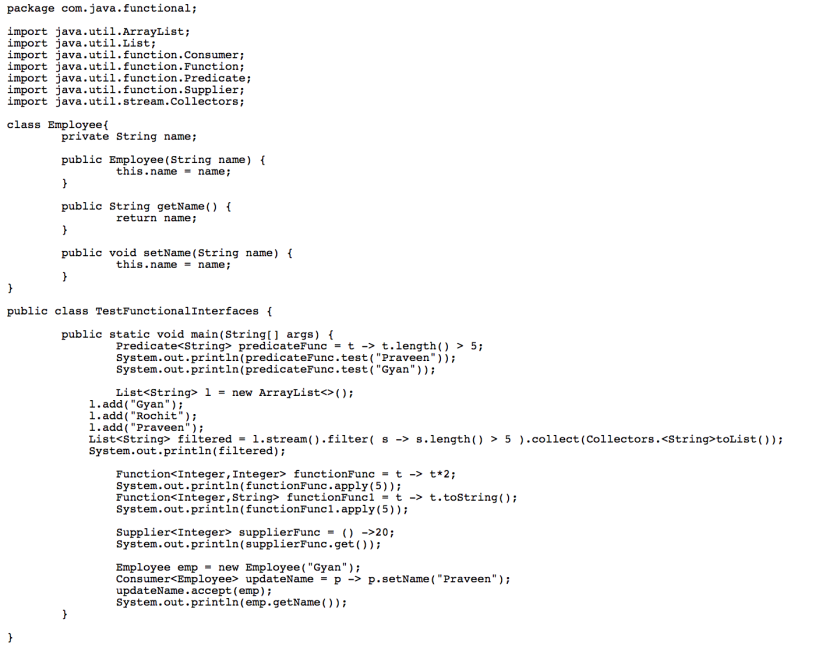
#### Supplier

This functional interface does the opposite of the Consumer, it takes no arguments but it returns some value. It may return different values when it is being called more than once. Since it’s a functional interface, you can pass a lambda expression wherever a Supplier is expected.



  
**Output**:  
20

Let us look into the entire program:



**Output**:  
true  
false  
[Rochit, Praveen]  
10  
5  
20  
Praveen

**Example:-1**

**import** java.util.function.Predicate;

**public** **class** PredicateInterfaceExample1 {

**public** **static** **void** main(String[] args) {

// Creating predicate

Predicate<Integer> lesserthan =i->(i<18);

// Calling Predicate method

System.***out***.println(lesserthan.test(10));

}}

**Example:-2**