**Java-8 – Predefined functional interface function**

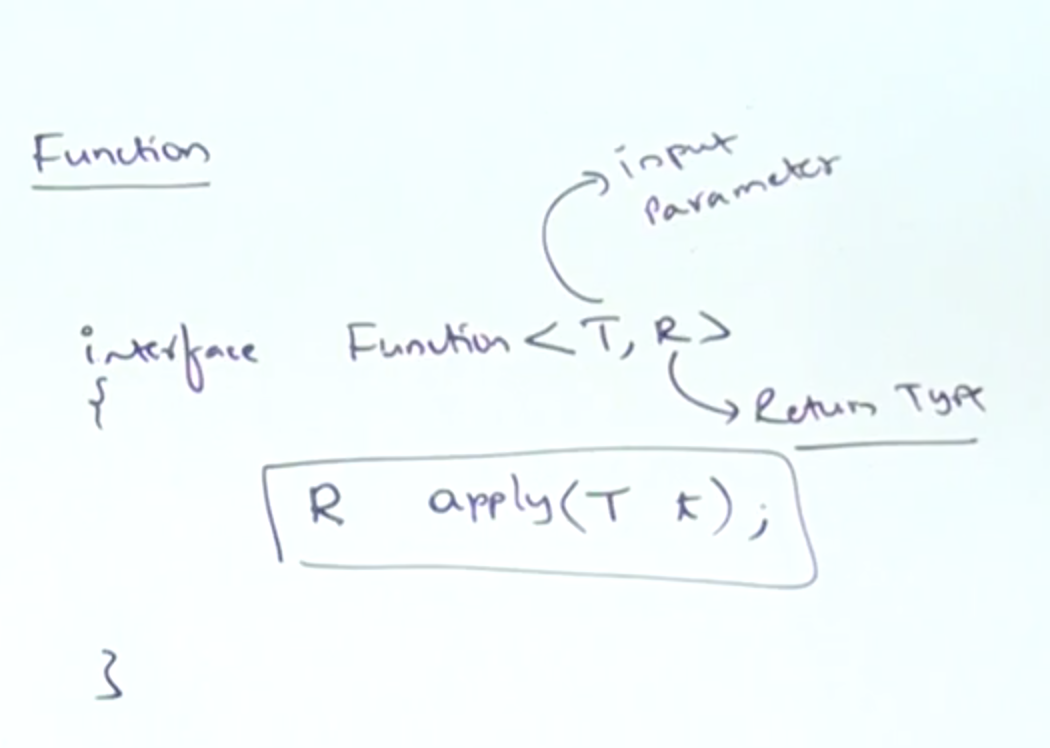
* **Function:**

Function is a mathematical word like predicate.

Predicate vs Function:

Predicate performs a conditional check and it always return boolean either true or false.

Function also performs some operation but it will return any type (int, boolean, Object, etc…)



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

* **Example:**

class FunctionDemo{

public static void main(String[] args){

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

System.out.println(function.apply(“saravana”)); // 8

System.out.println(function.apply(“durga”)); // 5

}

}

class FunctionDemo{

public static void main(String[] args){

Function<Integer, Integer> function = I -> I \* I;

System.out.println(function.apply(5)); // 25

System.out.println(function.apply(100)); // 100

}

}

* **Predicate vs Function:**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Predicate** | **Function** |
| 1 | To implement conditional checks, we should go for Predicate. | To perform certain operation and to return some result we should go for Function. |
| 2 | Predicate can take one type-parameter which represents input argument type.  Predicate<T> | Function can take two parameters. First one represents input argument and the second one represents return type.  Function<T, R> |
| 3 | Predicate interface defines only one abstract method called test() | Function interface defines one abstract method apply() |
| 4 | public boolean test(T t) | public R apply(T t) |
| 5 | Predicate can return only boolean value. | Function can return any type of value. |