DoublePredicate Functional Interface

Previously we have discussed [Predicate Interface](http://data-structure-learning.blogspot.com/p/1.html) and all its methods with examples. I would highly recommend that you read about that first before proceeding to read about DoublePredicate. Predicate Interface can accept any type but DoublePredicate can only accept input of double primitive type only.

DoublePredicate is a double valued predicate which deals with single boolean argument. So DoublePredicate is a specialized version (only for double argument) of Predicate interface.

|  |
| --- |
| **DoublePredicate Interface Declaration**  **public** **interface** DoublePredicate  DoublePredicate interface only accepts double argument |
| **test() method**  **boolean** test(**double** value);  test method is used to evaluate this predicate on given argument.  The below code returns true if parameter is greater than 10 else returns false.  DoublePredicate doublePred = (x) -> x > 10.0;  System.***out***.println(doublePred.test(11)); //Outputs true  System.***out***.println(doublePred.test(9)); //Outputs false |
| **and() method**  **default** DoublePredicate and(DoublePredicate other) {  Objects.*requireNonNull*(other);  **return** (value) -> test(value) && other.test(value);  }  and() returns the composed version of two different DoublePredicates. Both the predicates are evaluated in sequence. The composed version represents the short-circuiting logical and of both the predicates. This method will throw NullPointerException is *other* predicate is null.  We will write two different predicates. One checks for number greater than 10 and another check for number less than 20. So it will return true if number is greater than 10 **and** less than 20.  DoublePredicate doublePredGreater = (x) -> x > 10.0;  DoublePredicate doublePredLesser = (x) -> x < 20.0;  **DoublePredicate combine=doublePredGreater.and(doublePredLesser);**  System.***out***.println(combine.test(17)); //Outputs true  System.***out***.println(combine.test(25)); //Outputs false |
| **negate() method**  **default** DoublePredicate negate() {  **return** (value) -> !test(value);  }  negate() method returns the logical negation of this predicate.  Let us take simple example to demonstrate this method.  We will write a DoublePredicate that will return true if number is greater than 10.  DoublePredicate doublePredGreater = (x) -> x > 10.0;  System.***out***.println(doublePredGreater.test(11)); //Outputs true  System.***out***.println(doublePredGreater.test(9)); //Outputs false  Now we will apply negate method on this predicate.  DoublePredicate negation=doublePredGreater.negate();  System.***out***.println(negation.test(11)); //Outputs false  System.***out***.println(negation.test(9)); //Outputs true  The output is now reversed. |
| **or() method**  **default** DoublePredicate or(DoublePredicate other) {  Objects.*requireNonNull*(other);  **return** (value) -> test(value) || other.test(value);  }  or() method returns the composed predicate that represents the sort-circuiting logical OR for this predicate and another. While evaluating the composed version of predicate, if this predicate is true then the other predicate is not evaluated. This method will throw NullPointerException is *other* predicate is null.  We will write two different predicates. One checks for number greater than 10 and less than 20 and another check for number greater than 40 and less than 50. So it will return true if number is greater than 10 **and** less than 20 **or** number is greater than 40 **and** less than 50.  DoublePredicate doublePred1 = (x) -> x > 10.0 && x < 20.0;  DoublePredicate doublePred2 = (x) -> x > 40.0 && x < 50.0;    **DoublePredicate or=doublePred1.or(doublePred2);**  System.***out***.println(or.test(15)); //Outputs true. Number is between 10 and 20  System.***out***.println(or.test(47)); //Outputs true. Number is between 40 and 50  System.***out***.println(or.test(7)); //Outputs false. Number < 10 and Number < 40  System.***out***.println(or.test(32)); //Outputs false. Number > 20 and Number < 40 |

That’s all on DoublePredicate Interface