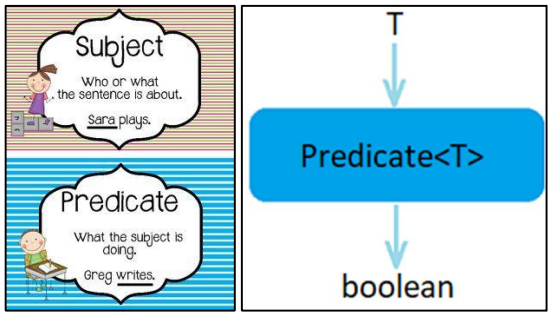
ИНТЕРФЕЙС PREDICATE



Predicate встроенный функциональный интерфейс, добавленный в *Java SE 8* в пакет *java.util.function*. Принимает на вход значение, проверяет состояние и возвращает *boolean* значение в качестве результата. *Predicate* подтверждает какое-то значение как *true* или *false*. Описание интерфейса *Predicate*:

```
package java.util.function;

import java.util.Objects;

/** Represents a predicate (boolean-valued function) of one argument. ...*/
@FunctionalInterface
public interface Predicate<T> {

    /** Evaluates this predicate on the given argument. ...*/
    boolean test(T t);

    /** Returns a composed predicate that represents a short-circuiting logical ...*/
     @Contract(pure = true)  @NotNull
    default Predicate<T> and( @NotNull Predicate<? super T> other) {...}

    /** Returns a predicate that represents the logical negation of this ...*/
     @Contract(pure = true)  @NotNull
    default Predicate<T> negate() { return (t) -> !test(t); }

    /** Returns a composed predicate that represents a short-circuiting logical ...*/
     @Contract(pure = true)  @NotNull
    default Predicate<T> or( @NotNull Predicate<? super T> other) {...}

    /** Returns a predicate that tests if two arguments are equal according ...*/
    static <T> Predicate<T> isEqual(Object targetRef) {...}

    /** Returns a predicate that is the negation of the supplied predicate. ...*/
    /unchecked/
    static <T> Predicate<T> not(Predicate<? super T> target) {...}
}
```

Функциональный дескриптор интерфейса:

```
T -> boolean
```

Рассмотрим пример использования интерфейса *Predicate* для нахождения отрицательных чисел:

```
import java.util.function.Predicate;
class PredicateDemo {
    public static void main(String[] args) {
        Predicate<Integer> negative = i -> i < 0;
        System.out.println(negative.test(-6));
        System.out.println(negative.test(6));
        System.out.println(negative.test(0));
    }}
```

Результат работы программы:

```
true
false
false
```

Predicate интерфейс содержит методы по умолчанию:

```
//Возвращает составной предикат, представляющий собой логическое замыкание И этого предиката и другого
default Predicate<T> and(Predicate<? super T> other);
//Возвращает составной предикат, представляющий собой логическое замыкание
```

```
ИЛИ этого предиката и другого
default Predicate<T> or(Predicate<? super T> other);
//Возвращает предикат, представляющий логическое отрицание этого предиката
default Predicate<T> negate();
```

Следующий пример демонстрирует использование метода *and()* интерфейса *Predicate*:

```
import java.util.function.Predicate;
class PredicateDemo {
    public static void main(String[] args) {
        Predicate<String> containsA = t -> t.contains("A");
        Predicate<String> containsB = t -> t.contains("B");
        System.out.println(containsA.and(containsB).test("ABCD"));
    }}
```

Результат работы программы:

```
true
```