Predefined Functional Interfaces

What is the use of Predefined Funcational Interfaces?

To make lambda expression as common coding activity, Java 8 provides some predefined functional interfaces.

Popular Funtional Interfaces :

1. Predicate
2. Function
3. Consumer
4. Supplier

Two argument predefined functional interface

1. BiPredicate
2. BiFunction
3. BiConsumer

Primitive Functional Interface

1. IntPredicate
2. IntFunction
3. IntConsumer

Predicate Functional Interface:

This functional interface contains SAM named test(T t), its(i.e. method test) return type is boolean, it will return boolean true/false value, based on expression evaluation.

See below example for more clarity.

Predicate<T> :

```
public Boolean test(Integer I){
        if(I%2==0){
                        System.out.println(“true”);
        }
        Else{
                        System.out.println(“false”);
```

```
    }
}
```

Program Implementation of Predicate functional interface and its test method

```java
import java.util.function.*;

public class TestPredicate {
    public static void main(String[] args) {
        Predicate<Integer> p = i->i%2==0;
        System.out.println(p.test(10));
        System.out.println(p.test(15));
    }
}
```

Predicate Joining:

To check very complex conditional expressions.

For e.g.

P1.and(p2).test(evaluatingExpression); :: To check p1 and p2 both returns true.

P1.or(p2) :: Either P1 or P2 returns true.

P1.negate(P2) :: Opposite of P1

Imp Points:

1. Whenever we have to do conditional checks then we can use their Predicate Functional Interface.
2. Predicate can take only one argument.