The method without any name (anonymous function) is called lambda expression. We refer lambda expression using functional interfaces.

Java provided so many functional interfaces like below

1. Predicate
2. Function
3. Consumer
4. Supplier
5. And so many…

Predicate 🡺 Predicate is a predefined functional interface introduced in java 1.8. It contains only one method “test”. It performs some conditional check and returns true or false. Predicate is a Boolean valued function

if you want to check some Boolean condition using lambda, you can use Predicate functional interface.

You can write it once, but use it anywhere you want

Let’s see Predicate interface from java.util.function (its interface provided by java and you no need to write it)

**package** java.util.function;

**import** java.util.Objects;

@FunctionalInterface

**public** **interface** Predicate<T> {

**boolean** test(T t);

**default** Predicate<T> and(Predicate<? **super** T> other) {

Objects.*requireNonNull*(other);

**return** (t) -> test(t) && other.test(t);

}

**default** Predicate<T> negate() {

**return** (t) -> !test(t);

}

**default** Predicate<T> or(Predicate<? **super** T> other) {

Objects.*requireNonNull*(other);

**return** (t) -> test(t) || other.test(t);

}

**static** <T> Predicate<T> isEqual(Object targetRef) {

**return** (**null** == targetRef) ? Objects::*isNull* : object -> targetRef.equals(object);

}

}

You can use predicate directly (test is the single abstract method in Predicate interface). You can use test method to check Boolean condition.

Predicate method is used for conditional checking

**import** java.util.ArrayList;

**import** java.util.Collection;

**import** java.util.List;

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

**public** **class** PredicateTest1 {

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

Predicate<String> strLengthPredicate = v1 -> (v1.length() > 5);

System.***out***.println(strLengthPredicate.test("Hello World!"));

System.***out***.println(strLengthPredicate.test("Hello"));

System.***out***.println("=========================");

Predicate<Integer> intGrtPredicate = v1 -> (v1 >= 10);

System.***out***.println(intGrtPredicate.test(12));

System.***out***.println(intGrtPredicate.test(2));

System.***out***.println("=========================");

Predicate<Collection<String>> colectionPredicate = c -> (c.isEmpty());

List<String> namesList1 = **new** ArrayList<String>();

System.***out***.println(colectionPredicate.test(namesList1));

List<String> namesList2 = **new** ArrayList<String>();

namesList2.add("John");

System.***out***.println(colectionPredicate.test(namesList2));

System.***out***.println("=========================");

}

}

Predicate joining

**import** java.util.ArrayList;

**import** java.util.Collection;

**import** java.util.List;

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

**public** **class** PredicateTest1 {

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

Predicate<String> strLengthPredicate = v1 -> (v1.length() > 5);

System.***out***.println(strLengthPredicate.test("Hello World!"));

System.***out***.println(strLengthPredicate.test("Hello"));

System.***out***.println("=========================");

Predicate<Integer> intGrtPredicate = v1 -> (v1 >= 10);

System.***out***.println(intGrtPredicate.test(12));

System.***out***.println(intGrtPredicate.test(2));

System.***out***.println("=========================");

Predicate<Collection<String>> colectionPredicate = c -> (c.isEmpty());

List<String> namesList1 = **new** ArrayList<String>();

System.***out***.println(colectionPredicate.test(namesList1));

List<String> namesList2 = **new** ArrayList<String>();

namesList2.add("John");

System.***out***.println(colectionPredicate.test(namesList2));

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Predicate Joining \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.***out***.println("==========and===============");

Predicate<Integer> intGrtThan10Predicate = v1 -> (v1 >= 10);

Predicate<Integer> isEvenPredicate = v1 -> (v1 % 2 == 0);

System.***out***.println(intGrtThan10Predicate.and(isEvenPredicate).test(64));

System.***out***.println(intGrtThan10Predicate.and(isEvenPredicate).test(65));

System.***out***.println("==========or===============");

System.***out***.println(intGrtThan10Predicate.or(isEvenPredicate).test(64));

System.***out***.println(intGrtThan10Predicate.or(isEvenPredicate).test(5));

System.***out***.println("==========negate===============");

System.***out***.println(intGrtThan10Predicate.negate().test(5));

System.***out***.println(intGrtThan10Predicate.negate().test(90));

System.***out***.println("==========isEqual===============");

Predicate<String> i = Predicate.*isEqual*("Hello");

System.***out***.println(i.test("Hello"));

System.***out***.println(i.test("World"));

}

}

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

**public** **class** PredicateTest3 {

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

User user1 = **new** User();

user1.setUsrId("John");

user1.setPwd("Secret");

User user2 = **new** User();

user2.setUsrId("John");

user2.setPwd("dummy");

Predicate<User> usrPredicate = u -> "John".equals(u.getUsrId()) && "Secret".equals(u.getPwd());

System.***out***.println("Login success " + usrPredicate.test(user1));

System.***out***.println("Login success " + usrPredicate.test(user2));

}

}

**class** User {

**private** String usrId;

**private** String pwd;

**public** String getUsrId() {

**return** usrId;

}

**public** **void** setUsrId(String usrId) {

**this**.usrId = usrId;

}

**public** String getPwd() {

**return** pwd;

}

**public** **void** setPwd(String pwd) {

**this**.pwd = pwd;

}

}

*isEqual*

Override equals and hascode

**class** User {

**private** String usrId;

**private** String pwd;

User() {

}

User(String usrId, String pwd) {

**this**.usrId = usrId;

**this**.pwd = pwd;

}

**public** String getUsrId() {

**return** usrId;

}

**public** **void** setUsrId(String usrId) {

**this**.usrId = usrId;

}

**public** String getPwd() {

**return** pwd;

}

**public** **void** setPwd(String pwd) {

**this**.pwd = pwd;

}

@Override

**public** **int** hashCode() {

**final** **int** prime = 31;

**int** result = 1;

result = prime \* result + ((pwd == **null**) ? 0 : pwd.hashCode());

result = prime \* result + ((usrId == **null**) ? 0 : usrId.hashCode());

**return** result;

}

@Override

**public** **boolean** equals(Object obj) {

**if** (**this** == obj)

**return** **true**;

**if** (obj == **null**)

**return** **false**;

**if** (getClass() != obj.getClass())

**return** **false**;

User other = (User) obj;

**if** (pwd == **null**) {

**if** (other.pwd != **null**)

**return** **false**;

} **else** **if** (!pwd.equals(other.pwd))

**return** **false**;

**if** (usrId == **null**) {

**if** (other.usrId != **null**)

**return** **false**;

} **else** **if** (!usrId.equals(other.usrId))

**return** **false**;

**return** **true**;

}

}

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

**public** **class** PredicateTest3 {

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

User user1 = **new** User();

user1.setUsrId("John");

user1.setPwd("Secret");

User user2 = **new** User();

user2.setUsrId("John");

user2.setPwd("dummy");

Predicate<User> usrPredicate = u -> "John".equals(u.getUsrId()) && "Secret".equals(u.getPwd());

System.***out***.println("Login success " + usrPredicate.test(user1));

System.***out***.println("Login success " + usrPredicate.test(user2));

Predicate<User> isSameUser = Predicate.*isEqual*(**new** User("John", "Secret"));

System.***out***.println(isSameUser.test(user1));

System.***out***.println(isSameUser.test(user2));

}

}