Lamda Expressions

# What are Lambda Expressions?

* In Java, Lambda expressions express instances of functional interfaces (An interface with a single abstract method is called a functional interface)
* Lambda expressions were introduced in Java 8
* A lambda expression is a short block of code that takes in parameters and returns a value.
* Lambda expressions are similar to methods, but they do not need a name and they can be implemented right in the body of a method.

*parameter* -> *expression*

for more than one parameter

*(parameter1, parameter2)* -> *expression*

# Example of how to implement a Lambda expression

* Let us create a class with Main() method
* Add an interface with a single method which is called a functional interface
* We can add a Functional Interface tag (@FunctionalInterface) for the interface with a single abstract method.

Example:

@FunctionalInterface

**interface** A

{

**void** printText();

}

* In order to call the method, it should first be implemented in a class like below

**class** B **implements** A

{

**public** **void** printText()

{

System.***out***.println("Hello World");

}

}

Now this can be called in main

A obj = **new** B();

obj.printText();

* Now, instead of having a separate class (in this case, Class B) to implement the method from functional Interface, I can directly add it in Main with the help of Lambda Expressions.

A obj = () -> System.***out***.println("Hello World");

obj.printText();

# Lambda Expressions with parameters and return type

Example of a lambda expression with parameters:

@FunctionalInterface

**interface** A

{

**int** addNumbers(**int** num1, **int** num2);

}

A obj = (num1,num2) -> num1 + num2;

System.***out***.println(obj.addNumbers(10,20));

--------------------Example 2 (With multiple statements-------------------------------------

@FunctionalInterface

**interface** Printing{

    String print(String message);

}

**public** **class** LambdaExpressionExample{

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

        Printing forPrinting = (message)-> {

            String str1 = "Java is a  ";

            String str2 = str1 + message;

**return** str2;

        };

            System.out.println(person.say("programming language"));

    }

}

# Method References

Java provides a new feature called method reference in Java 8. Method reference is used to refer method of functional interface. It is a compact and easy form of lambda expression.

When using method reference, the functional interface signature must match the method signature.

Types of Method References

There are the following types of method references in Java:

1. Reference to a static method.

ContainingClass::staticMethodName

1. Reference to an instance method.

containingObject::instanceMethodName

1. Reference to a constructor.

ClassName::**new**

**Example for a static reference:**

@FunctionalInterface

**interface** B

{

**void** printSomething();

}

**public** **class** TryWithResources {

**public** **static** **void** printHello()

{

System.***out***.println("Print Hello!!!");

}

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

B obj2 = TryWithResources::*printHello*;

obj2.printSomething();

}

**Reference to an instance method**

**interface** Sayable{

**void** say();

}

**public** **class** InstanceMethodReference {

**public** **void** saySomething(){

        System.out.println("Hello, this is non-static method.");

    }

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

        InstanceMethodReference methodReference = **new** InstanceMethodReference(); // Creating object

        // Referring non-static method using reference

            Sayable sayable = methodReference::saySomething;

        // Calling interface method

            sayable.say();

}

}

**Reference to a constructor**

**interface** Messageable{

    printMessage(String msg);

}

**class** Message{

    Message(String msg){

        System.out.print(msg);

    }

}

**public** **class** ConstructorReference {

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

        Messageable hello = Message::**new**;

        hello.getMessage("Hello");

    }

}