Lecture 76 Method Reference By Using :: (Double colon) Operator – Part1:

Double Colon Operator are used for Constructor and Method Reference.

Consider we have an interface shown below.

**interface** Interf{

**public** **void** m1();

}

Let us see how this interface can be implemented? The implementation can be done in 3 ways.

* Old or Classic way ( By creating a new class and then implementing the interface).

**class** Test **implements** Interf{

@Override

**public** **void** m1() {

// **TODO** Auto-generated method stub

System.***out***.println("Old method ");

}

}

**interface** Interf{

**public** **void** m1();

}

* By using Lambda Expression:

**public** **class** Example1 {

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

Interf interf = () ->{

System.***out***.println("Lambda expression");

};

}

}

**interface** Interf{

**public** **void** m1();

}

* By Method Reference:

**package** com.durgaSoft.section7.lecture76;

**public** **class** Example2 {

**public** **static** **void** m2() {

System.***out***.println("Method Reference");

}

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

Interf interf = Example2::*m2*;

interf.m1();

}

}

In this above implementation we already have a method m2 which is implementing the logic for m1 method available in the Interface Interf. So instead of rewriting the code we can reuse the exiting code using method reference.

Note: While using method reference,

1. Both should have same argument type.
2. Method name, return type, modifier type can be different.

Lecture 78 Method Reference By Using :: (Double colon) Operator – Part2:

Method reference when the implementation is done in static method

Static Method:

Class Name :: Method Name

Instance Method:

Object Reference :: Method Name

Example for instance method:

**package** com.durgaSoft.section7.lecture77;

**public** **class** Example1 {

**public** **void** m2() {

System.***out***.println("Method Reference");

}

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

Example1 example = **new** Example1();

Interf1 interf = example::m2;

interf.m1();

}

}

**interface** Interf1{

**public** **void** m1();

}

Example for implementation of Run method using Lambda Expression:

**package** com.durgaSoft.section7.lecture77;

**public** **class** Example2 {

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

Runnable runnable = () -> {

**for**(**int** i=0;i<10;i++) {

System.***out***.println("Child Thread");

}

};

Thread t = **new** Thread(runnable);

t.start();

**for**(**int** i=0;i<10;i++) {

System.***out***.println("Main Thread");

}

}

}

Example for implementation of Run method using Method Reference:

**package** com.durgaSoft.section7.lecture77;

**public** **class** Example3 {

**public** **void** m1() {

**for** (**int** i = 0; i < 10; i++) {

System.***out***.println("Child Thread");

}

}

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

Example3 example = **new** Example3();

Runnable runnable = example::m1;

Thread t = **new** Thread(runnable);

t.start();

**for** (**int** i = 0; i < 10; i++) {

System.***out***.println("Main Thread");

}

}

}

Lecture 79 Constructor Reference By Using :: (Double colon) :

Normal Lambda Expression:

**package** com.durgaSoft.section7.lecture79;

**public** **class** Example1 {

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

Interf79 interf = () -> **new** Sample();

}

}

**class** Sample{

**public** Sample() {

// **TODO** Auto-generated constructor stub

System.***out***.println("Sample Default Constructor");

}

}

**interface** Interf79{

**public** Sample get();

}

Using Constructor Method: For No Argument Constructor

**package** com.durgaSoft.section7.lecture79;

**public** **class** Example2 {

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

Interf792 interf = SampleExample2 :: **new**;

interf.get();

}

}

**class** SampleExample2{

**public** SampleExample2() {

// **TODO** Auto-generated constructor stub

System.***out***.println("Sample Default Constructor");

}

}

**interface** Interf792{

**public** SampleExample2 get();

}

For Argument Constructor:

**package** com.durgaSoft.section7.lecture79;

**public** **class** Example2 {

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

Interf792 interf = SampleExample2 :: **new**;

interf.get();

Interf793 interf1 = SampleExample2:: **new**;

interf1.get(3);

}

}

**class** SampleExample2{

**int** data =0;

**public** SampleExample2() {

// **TODO** Auto-generated constructor stub

System.***out***.println("Sample Default Constructor");

}

**public** SampleExample2(**int** data) {

System.***out***.println("We are in Single Argument Constructor");

}

}

**interface** Interf792{

**public** SampleExample2 get();

}

**interface** Interf793{

**public** SampleExample2 get(**int** number);

}

JVM checks and gives us the correct argument constructors.