Java-8

* Lambda Expression
* Functional Interface
* Stream API
* Default method in interface
* Static method in interface
* Optional class
* Collectors class
* ForEach() Method/Loop
* Parallel array sorting
* Method,constructor references- ::
* So -on….

Need of java8

* Concise and minimal code
* Add features of functional programming
* More compatible code with multi-core processor

Lambda Expression-

Anonymous function

* Not having name
* Not having return type
* Not having modifier
* -> needs to be added

Cleaning rules-

1. If body have just one statement then we can remove {}
2. Use type inference, compiler guess the situation and context
3. No return keyword
4. If only one parameter remove small brackets

Example1-

Private void add(int a,int b)

{

SOP(a+b)

}

Cleaned lambda expression

(a,b)->SOP(a+b);

Example 2-

Private int getstrlength(String str)

{

Return str.length();

}

Cleaned lambda expression

str->str.length();

Benefits-

1.To enable functional programming

2.To make code more readable,maintainable and concise code

3. To enable parallel processing

4.Reduce jar file size

5. Eliminate unnecessary variables

public class LambdaMathExample {

public static void main(String[] args) {

// Lambda expression for addition

MathOperation add = (a, b) -> a + b;

// Lambda expression for subtraction

MathOperation subtract = (a, b) -> a - b;

// Perform addition

System.out.println("5 + 3 = " + operate(5, 3, add));

// Perform subtraction

System.out.println("5 - 3 = " + operate(5, 3, subtract));

}

// Functional interface for math operations

interface MathOperation {

int operate(int a, int b);

}

// Method to perform operation using lambda expression

private static int operate(int a, int b, MathOperation operation) {

return operation.operate(a, b);

}

}

Functional Interface

* Have only and only 1 abstract method
* Earlier known as SAM
* Any number of default and static
* Need to put @FunctionalInterface to make sure only one abstract method is present.
* Functional interface act as a data type of lambda expression
* Functional Interface can be used to hold lambda expression

package Itep10;

@FunctionalInterface

interface Exp{

void show();

default void data1(){

System.out.println("data.....");

}

}

public class demo {

public static void main(String[] args) {

Exp imp = ()-> System.out.println("we are child");

imp.show();

imp.data1();

System.out.println(imp);

}

}