**Lambda expression**

Lambda expression were introduced in Java 1.8. This is one of the best features of Java (of course lambda expression is already part of many programming languages, but in Java they are introduced in Java 1.8).

We can write java code in functional style using Lambda expression.

Anonymous function syntax:

|  |  |  |
| --- | --- | --- |
| Parameter | Expression | Body |
| () | -> | { } |
| (I,j) | -> | {return i+j}  Or {i+j} |

It’s an expression through which we can represent an anonymous function.

Examples:

|  |  |
| --- | --- |
| Old Traditional style | Lambda Style |
| **public** **class** RunnableImpl **implements** Runnable {  **public** **void** run() {  System.***out***.println("Task is running");  }  }  Thread t1 = **new** Thread(**new** RunnableImpl());  t1.start(); | Runnable task = () -> {  System.***out***.println("Task is running");  };  Thread t2 = **new** Thread(task);  t2.start();  or  Thread t3 = **new** Thread(() -> {  System.***out***.println("Task is running");  });  t3.start(); |
| **public** **interface** LambdaExample2Intr {  **public** **void** m1();  }  **public** **class** LambdaExample2a **implements** LambdaExample2Intr {  @Override  **public** **void** m1() {  System.***out***.println("m1 method implementation");  }  **public** **static** **void** main(String[] args) {  LambdaExample2a inter = **new** LambdaExample2a();  inter.m1();  }  } | @FunctionalInterface  **public** **interface** LambdaExample2Intr {  **public** **void** m1();  }  **public** **class** LambdaExample2a1 {  **public** **static** **void** main(String[] args) {  LambdaExample2Intr example =()->{  System.***out***.println("m1 method implementation");};  example.m1();  }  } |

It’s easy and simple in lambda way (just on line)

Thread t3 = **new** Thread(() ->System.***out***.println("Task is running"));

t3.start();

**Collections Sorting using Lambda**

**public** **class** Product {

**private** **int** id;

**private** String name;

**private** String category;

**private** **int** qty;

**private** **double** pricePerKg;

**private** **double** totalValue;

**public** Product() {

}

**public** Product(**int** id, String name, String category, **int** qty, **double** pricePerKg) {

**this**.id = id;

**this**.name = name;

**this**.category = category;

**this**.qty = qty;

**this**.pricePerKg = pricePerKg;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getCategory() {

**return** category;

}

**public** **void** setCategory(String category) {

**this**.category = category;

}

**public** **int** getQty() {

**return** qty;

}

**public** **void** setQty(**int** qty) {

**this**.qty = qty;

}

**public** **double** getPricePerKg() {

**return** pricePerKg;

}

**public** **void** setPricePerKg(**double** pricePerKg) {

**this**.pricePerKg = pricePerKg;

}

**public** **double** getTotalValue() {

**return** totalValue;

}

**public** **void** setTotalValue(**double** totalValue) {

**this**.totalValue = totalValue;

}

}

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** ProductUtil {

**public** List<Product> getProducts() {

List<Product> prodList = **new** ArrayList<Product>();

prodList.add(**new** Product(1, "buckwheat", "cereals", 20, 32.50));

prodList.add(**new** Product(2, "Hercules", "cereals", 20, 12.50));

prodList.add(**new** Product(3, "millet", "cereals", 15, 14.60));

prodList.add(**new** Product(4, "cookies cracker", "biscuit", 40, 15.00));

prodList.add(**new** Product(5, "Wafer chocolate", "Wafer", 25, 23.50));

prodList.add(**new** Product(6, "vanilla wafers", "Wafer", 25, 23.20));

prodList.add(**new** Product(7, "favorite cookies", "biscuit", 45, 20.80));

prodList.add(**new** Product(8, "ravioli", "pasta", 20, 25.00));

prodList.add(**new** Product(9, "spaghetti", "pasta", 20, 27.00));

prodList.add(**new** Product(10, "vermicelli", "pasta", 20, 25.00));

prodList.add(**new** Product(11, "vareniki", "pasta", 20, 25.00));

prodList.add(**new** Product(12, "Hercules", "cereals", 23, 55.00));

prodList.add(**new** Product(13, "buckwheat", "cereals", 21, 15.00));

prodList.add(**new** Product(14, "pearl barley", "cereals", 50, 78.12));

prodList.add(**new** Product(15, "semolina", "cereals", 12, 75.00));

prodList.add(**new** Product(16, "swallow", "cereals", 11, 12.00));

prodList.add(**new** Product(17, "Karakum", "candies", 78, 2.00));

prodList.add(**new** Product(18, "vermicelli", "candies", 17, 1.00));

prodList.add(**new** Product(19, "leopard", "candies", 19, 8.00));

prodList.add(**new** Product(20, "Mars", "candies", 2, 11.00));

**return** prodList;

}

}

OLD WAY

**import** java.util.Collections;

**import** java.util.Comparator;

**import** java.util.List;

**public** **class** LambdaExample2c1 {

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

List<Product> prodList = ProductUtil.*getProducts*();

System.***out***.println("-----------Before Sort-----------");

**for** (Product p : prodList) {

System.***out***.println(p.getName());

}

System.***out***.println("-----------After Sort-----------");

Collections.*sort*(prodList, **new** MyComparator());

**for** (Product p : prodList) {

System.***out***.println(p.getName());

}

}

}

**class** MyComparator **implements** Comparator<Product> {

@Override

**public** **int** compare(Product o1, Product o2) {

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

**return** o1.getName().compareTo(o2.getName());

}

}

Lambda approach

**import** java.util.List;

**public** **class** LambdaExample2c {

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

List<Product> prodList = ProductUtil.*getProducts*();

System.***out***.println("-----------Before Sort-----------");

**for** (Product p : prodList) {

System.***out***.println(p.getName());

}

System.***out***.println("-----------After Sort-----------");

prodList.sort((Product o1, Product o2) -> o1.getName().toLowerCase().compareTo(o2.getName().toLowerCase()));

**for** (Product p : prodList) {

System.***out***.println(p.getName());

}

}

}

=======================-===========================================

We can learn more on how to use lambda expressions in Stream API