Лабораторная работа

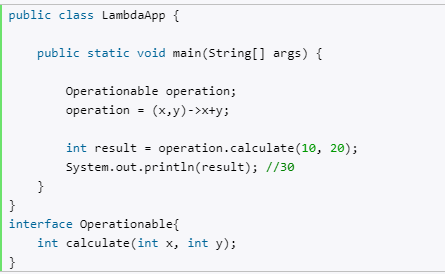
Тема: «Лямбда - выражения»

Интерфейс, который содержит только один абстрактный метод, называется **функциональным интерфейсом**.

В этом случае код можно написать более коротко и понятно, если использовать лямбда-выражения.

Основу лямбда-выражения составляет лямбда-оператор, который представляет стрелку –>. Этот оператор разделяет лямбда-выражение на две части: левая часть содержит список параметров выражения, а правая собственно представляет тело лямбда-выражения, где выполняются все действия.

Лямбда-выражение не выполняется само по себе, а образует реализацию метода, определенного в функциональном интерфейсе. При этом важно, что функциональный интерфейс должен содержать только один единственный метод без реализации.



В роли функционального интерфейса выступает интерфейс **Operationable**, в котором определен один метод без реализации - метод **calculate**. Данный метод принимает два параметра - целых числа, и возвращает некоторое целое число.

По факту лямбда-выражения являются в некотором роде сокращенной формой внутренних анонимных классов, которые ранее применялись в Java. В частности, предыдущий пример мы можем переписать следующим образом:

public class LambdaApp {

     public static void main(String[] args) {

        Operationable op = new Operationable(){

          public int calculate(int x, int y){

                return x + y;  }        };

         int z = op.calculate(20, 10);

         System.out.println(z); // 30    }   }

interface Operationable{     int calculate(int x, int y); }

Чтобы объявить и использовать лямбда-выражение, основная программа разбивается на ряд этапов:

1. Определение ссылки на функциональный интерфейс:

Operationable operation;

1. Создание лямбда-выражения:

operation = (x,y)->x+y;

*Причем параметры лямбда-выражения соответствуют параметрам единственного метода интерфейса Operationable, а результат соответствует возвращаемому результату метода интерфейса. При этом нам не надо использовать ключевое слово return для возврата результата из лямбда-выражения.*

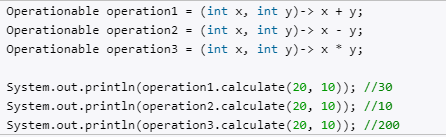
*Так, в методе интерфейса оба параметра представляют тип int, значит, в теле лямбда-выражения мы можем применить к ним сложение. Результат сложения также представляет тип int, объект которого возвращается методом интерфейса.*

1. Использование лямбда-выражения в виде вызова метода интерфейса:

int result = operation.calculate(10, 20);

Так как в лямбда-выражении определена операция сложения параметров, результатом метода будет сумма чисел 10 и 20.

При этом для одного функционального интерфейса мы можем определить множество лямбда-выражений. Например:



Одним из ключевых моментов в использовании лямбд является отложенное выполнение (deferred execution). То есть мы определяем в одном месте программы лямбда-выражение и затем можем его вызывать при необходимости неопределенное количество раз в различных частях программы. Отложенное выполнение может потребоваться, к примеру, в следующих случаях:

* Выполнение кода отдельном потоке
* Выполнение одного и того же кода несколько раз
* Выполнение кода в результате какого-то события
* Выполнение кода только в том случае, когда он действительно необходим и если он необходим.

**Передача параметров в лямбда-выражение**

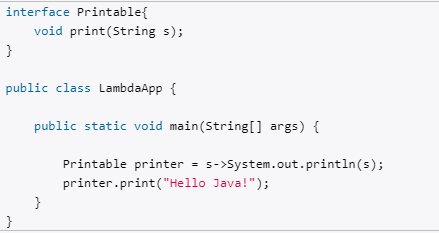
Параметры лямбда-выражения должны соответствовать по тип параметрам метода из функционального интерфейса.

Если метод не принимает никаких параметров, то пишутся пустые скобки, например: ()-> 30 + 20;

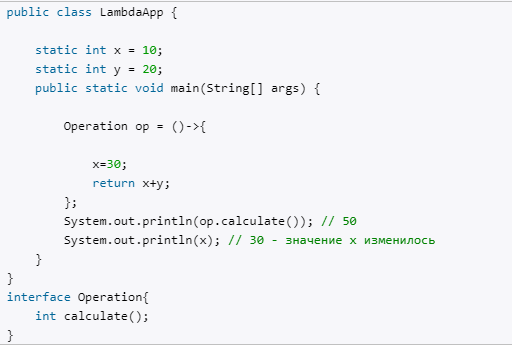
Если метод принимает только один параметр, то скобки можно опустить:

n-> n \* n;

Могут быть и терминальные лямбды, которые не возвращают никакого значения. Например:

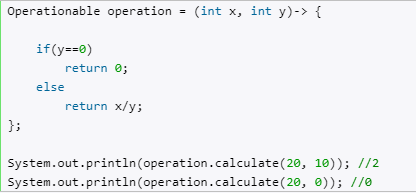


Лямбда-выражение может использовать переменные, которые объявлены во вне в более общей области видимости - на уровне класса или метода, в котором лямбда-выражение определено. Однако в зависимости от того, как и где определены переменные, могут различаться способы их использования в лямбдах. Рассмотрим первый пример - использования переменных уровня класса:

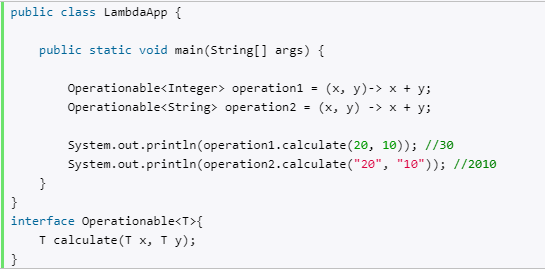


Переменные x и y объявлены на уровне класса, и в лямбда-выражении мы их может получить и даже изменить. Так, в данном случае после выполнения выражения изменяется значение переменной x.

**Блочные выражения** обрамляются фигурными скобками. В **блочных лямбда**-выражениях можно использовать внутренние вложенные блоки, циклы, конструкции if, switch, создавать переменные и т.д. Если блочное лямбда-выражение должно возвращать значение, то явным образом применяется оператор return:



Функциональный интерфейс может быть обобщенным, однако в лямбда-выражении использование обобщений не допускается. В этом случае нам надо типизировать объект интерфейса определенным типом, который потом будет применяться в лямбда-выражении. Например:



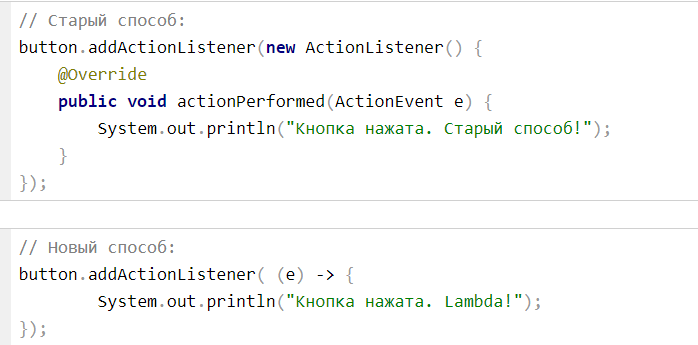
Таким образом, при объявлении лямбд-выражения ему уже известно, какой тип параметры будут представлять и какой тип они будут возвращать.

Встроенные функциональные интерфейсы (<https://metanit.com/java/tutorial/9.3.php>)

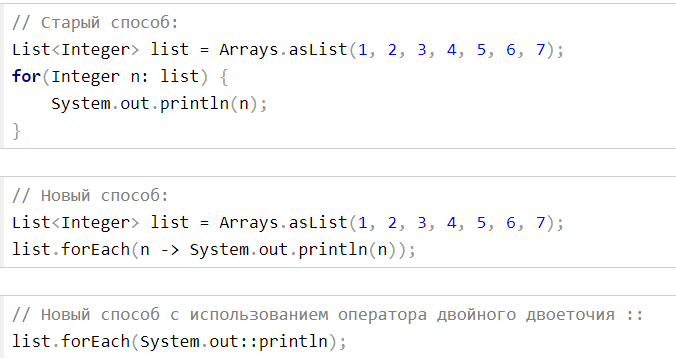
Лямбды, как параметры методов ( <https://metanit.com/java/tutorial/9.2.php>)

<https://java-master.com/stream-and-lambda-in-java/>

Управление событиями в Java 8 также можно осуществлять через Lambda-выражения. Далее представлены два способа добавления обработчика события ActionListener в компонент пользовательского интерфейса:



Простой пример вывода всех элементов заданного массива. Заметьте, что есть более одного способа использования lambda-выражения. Ниже мы создаем lambda-выражение обычным способом, используя синтаксис стрелки, а также мы используем оператор двойного двоеточия (::), который в Java 8 конвертирует обычный метод в lambda-выражение:



В следующем примере мы используем функциональный интерфейс Predicate для создания теста и печати элементов, прошедших этот тест. Таким способом вы можете помещать логику в lambda-выражения и делать что-либо на ее основе.

**import** java.util.Arrays;

**import** java.util.List;

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

**public** **class** Main {

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

List<Integer> list = Arrays.asList(1, 2, 3, 4, 5, 6, 7);

System.out.print("Выводит все числа: ");

evaluate(list, (n)->true);

System.out.print("Не выводит ни одного числа: ");

evaluate(list, (n)->false);

System.out.print("Вывод четных чисел: ");

evaluate(list, (n)-> n%2 == 0 );

System.out.print("Вывод нечетных чисел: ");

evaluate(list, (n)-> n%2 == 1 );

System.out.print("Вывод чисел больше 5: ");

evaluate(list, (n)-> n > 5 );

}

**public** **static** **void** evaluate(List<Integer> list, Predicate<Integer> predicate) {

**for**(Integer n: list) {

**if**(predicate.test(n)) {

System.out.print(n + " ");

}

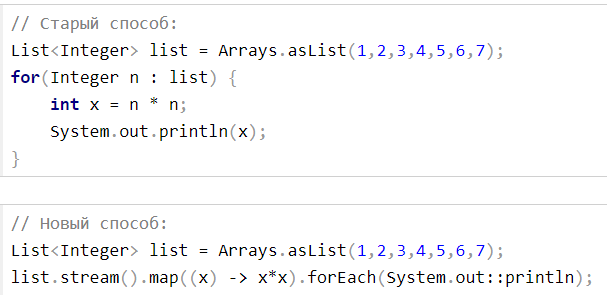
}

System.out.println();

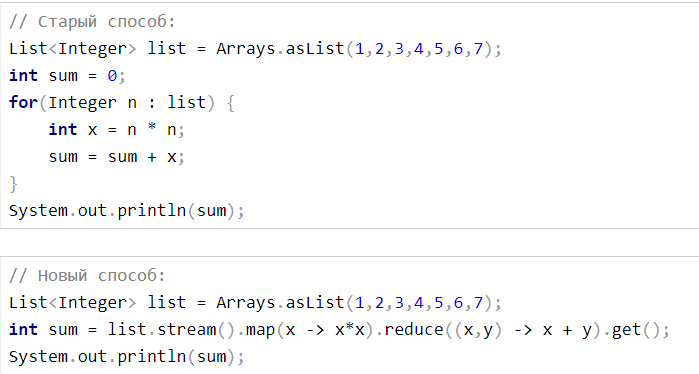
}

}

Можно вывести квадрат каждого элемента списка. Мы используем метод stream(), чтобы преобразовать обычный список в поток (класс Stream (java.util.stream.Stream). Он содержит полезные методы, с которыми можно использовать lambda-выражения. Передаем lambda-выражение x -> x\*x в метод map(), который применяет его ко всем элементам в потоке. После чего используем forEach для печати всех элементов списка.



Дан список, нужно вывести сумму квадратов всех элемента списка. Lambda-выражения позволяет достигнуть этого написанием всего одной строки кода. В этом примере применен метод свертки (редукции) reduce().Используем метод map() для возведения в квадрат каждого элемента, а потом применяем метод reduce() для свертки всех элементов в одно число.



<https://javadevblog.com/polnoe-rukovodstvo-po-lyambda-vy-razheniyam-i-funktsionalnym-interfejsam-v-java-8.html>

<https://java-master.com/stream-and-lambda-in-java/>

<https://urvanov.ru/2016/04/05/java-8-%D0%B2%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5-%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D1%8B-%D0%B8-%D0%BB%D1%8F%D0%BC%D0%B1%D0%B4%D0%B0-%D0%B2%D1%8B%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F/>

САМОСТОЯТЕЛЬНО : Выполнить два задания варианта.

**Задание 1:** Разработать и продемонстрировать метод с использованием лямбда-выражения, возвращающий значение, согласно варианту задания.

Вариант 1. Длину гипотенузы прямоугольного треугольника по заданным длинам катетов.

Вариант 2. Длину катета прямоугольного треугольника по заданным длинам гипотенузы и второго катета.

Вариант 3. Длину катета прямоугольного треугольника по заданным длине гипотенузы и прилежащему углу.

Вариант 4. Длину катета прямоугольного треугольника по заданным длине гипотенузы и противолежащему углу.

Вариант 5. Длину гипотенузы прямоугольного треугольника по заданным длине катета и прилежащему углу.

Вариант 6. Длину гипотенузы прямоугольного треугольника по заданным длине катета и противолежащему углу.

Вариант 7. Длину стороны треугольника по длинам двух других сторон и углу между ними.

Вариант 8. Площадь треугольника по его заданным сторонам.

Вариант 9. Площадь треугольника по заданным длинам двух сторон и углу между ними.

Вариант 10. Высоту равнобокой трапеции по заданным её сторонам.

Вариант 11. Площадь трапеции по заданным её сторонам.

Вариант 12. Длину окружности заданного радиуса.

Вариант 13. Площадь круга заданного радиуса.

Вариант 14. Площадь параллелограмма по заданным сторонам и углу между ними.

Вариант 15. Факториал.

Вариант 16. Объем цилиндра по радиусу основания и высоте.

Вариант 17. Объем конуса по радиусу основания и высоте.

**Задание 2:** С использованием лямбда-выражений выполнить задание согласно варианту.

Вариант 1. Выбрать из списка чисел элементы, меньшие заданного, и сортировать их по убыванию.

Вариант 2. Выбрать из списка чисел нечетные элементы, большие заданного, и сортировать их по возрастанию.

Вариант 3. Выбрать из списка чисел отрицательные элементы, кратные пяти, и сортировать их по убыванию.

Вариант 4. Выбрать из списка чисел положительные элементы, кратные трём, сортировать их по возрастанию и выбрать первые четыре.

Вариант 5. Выбрать из списка строк элементы заданной длины и сортировать их по возрастанию.

Вариант 6. Выбрать из списка строк элементы, длиной больше заданной величины, сортировать их по убыванию и вывести последние три.

Вариант 7. Отобрать из заданного списка векторов относящиеся к первому квадранту и сортировать их по возрастанию модуля.

Вариант 8. Отобрать из заданного списка векторов относящиеся ко второму октанту и сортировать их по убыванию модуля.

Вариант 9. Отобрать из заданного списка структур элементы, имеющие значение ключа, большее заданного, и сформировать список строковых полей этих структур.

Вариант 10. Отобрать из заданного списка структур элементы, имеющие значение ключа, меньшее заданного, и сформировать список строковых полей этих структур, сортированный по убыванию.

Вариант 11. Отобрать из заданного списка структур элементы, имеющие значение ключа, находящееся в заданном диапазоне, сформировать список строковых полей этих структур и отобрать из него элементы заданной длины.

Вариант 12. Отобрать из заданного списка структур элементы, имеющие значение ключа, находящееся за пределами заданного диапазона, сформировать список строковых полей этих структур и отобрать из него элементы, содержащее заданное слово.

Вариант 13. Отобрать из заданного списка структур элементы, имеющие значение ключа, находящееся в заданном диапазоне, сформировать список полей типа Дата этих структур и отобрать из него элементы, большие заданной даты.

Вариант 14. Отобрать из заданного списка структур элементы, имеющие значение ключа, меньшее заданной величины, сформировать список полей типа Дата этих структур, отобрать из него элементы, меньшие заданной даты, и сортировать по убыванию.

Вариант 15. Отобрать из заданного списка структур элементы, имеющие значение ключа, большее заданной величины, сформировать список полей типа Дата этих структур, отобрать из него элементы, большие заданной даты, сортировать по возрастанию и отобрать последние пять.

Вариант 16. Отобрать из заданного списка структур элементы, имеющие значение ключа, находящееся в заданном диапазоне, сформировать список полей типа Дата этих структур, отобрать из него элементы с заданным месяцем, сортировать по убыванию и отобрать первые три.

Вариант 17. Отобрать из заданного списка структур элементы, имеющие значение ключа, не входящее в заданный диапазон, сформировать список полей типа Дата этих структур, отобрать из него элементы, с заданным месяцем, и сортировать по возрастанию.

КОНТРОЛЬНЫЕ ВОПРОСЫ:

1. Дать определение функционального интерфейса.
2. Определение лямбда-выражений.
3. Как передаются параметры в лямбда-выражение, привести пример.
4. Блочные лямбда выражения, привести пример.
5. Встроенные функциональные интерфейсы в лямбда-выражении, привести пример.
6. Методы класса stream, которые можно использовать в лямбда –выражении.