

 Evgen959 / Advanced_Backend Public[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#)[Advanced_Backend](#) / [Lesen 043](#) / [code](#) / [hw_na_les042_1](#) / [src](#) / [Main.java](#) 

...



Evgen959 newLes43

d71a156 · 9 hours ago



93 lines (78 loc) · 3.57 KB

Code

Blame

Raw



```
1  /*
2  Объявите функциональный интерфейс MyPredicate, в котором должен быть единственный метод boolean
3  Используя ваш интерфейс реализуйте универсальный метод фильтрации списка Person.
4  Т.е. ваш метод должен принимать список Person и интерфейс MyPredicate который задает условие
5  (подсказка: это очень похоже на код StringTransformer написанный в классе)
6  Выполните несколько вариантов отбора Person (по началу имени, по возрасту и т.д.).
7  Попробуйте реализовать MyPredicate с помощью лямбда выражений (стрелочных функций).
8  */
9
10
11  import java.util.ArrayList;
12  import java.util.List;
13  import java.util.function.Predicate;
14
15  public class Main {
16      public static void main(String[] args) {
17          List<Person> people = List.of(
18              new Person("Jack", 19),
19              new Person("Ann", 20),
20              new Person("Garry", 35),
21              new Person("Stann", 10),
22              new Person("Duck", 13),
23              new Person("Nickolaus", 8)
24          );
25
26          List<Person> newList1 = filter(people, new MyPredicateImpl());
27          System.out.println("-----newList1-----");
28          System.out.println(newList1);
29
30          System.out.println("----- 2 -----");
31          List<Person> newList2 = filter(people, new MyPredicate() {
32              @Override
33              public boolean test(Person person) {
34                  return person.getName().charAt(0) < 'J';
35              }
36          });
37          System.out.println(newList2);
38      }
```

```
39     System.out.println("----- 3 -----");
40     List<Person> newList3 = filter(people, new MyPredicate() {
41         @Override
42         public boolean test(Person person) {
43             return person.getName().length() == 3;
44         }
45     });
46     System.out.println(newList3);
47
48     System.out.println("----- 4 -----");
49     List<Person> newList4 = filter(people, person -> person.getName().length() == 3);
50     System.out.println(newList4);
51
52     System.out.println("----- 5 -----");
53     MyPredicate myPredicate = person -> person.getName().length() == 3;
54     System.out.println(filter(people, myPredicate));
55
56     System.out.println("----- 6 -----");
57     System.out.println(filter2(people, p -> p.getName().length() > 4));
58
59     System.out.println(filter2(List.of(1,3,4,-2,3), i-> i%2==0));
60
61
62 }
63
64 ✓ public static List<Person> filter(List<Person> list, MyPredicate predicate){
65     List<Person> resaltList = new ArrayList<>();
66     for (Person p : list){
67         if (predicate.test(p)){
68             resaltList.add(p);
69         }
70     }
71     return resaltList;
72 }
73
74 ✓ public static <E> List<E> filter2(List<E> list, MyPredicate2<E> predicate){
75     List<E> resaltList = new ArrayList<>();
76     for (E p : list){
77         if (predicate.test(p)){
78             resaltList.add(p);
79         }
80     }
81     return resaltList;
82 }
83
84 ✓ public static <E> List<E> filter3(List<E> list, Predicate<E> predicate){
85     List<E> resaltList = new ArrayList<>();
86     for (E p : list){
87         if (predicate.test(p)){
88             resaltList.add(p);
89         }
90     }
91     return resaltList;
92 }
```

```
93      }
```