Advanced_Backend / Lesen 029 / code / hw_na_les028_01 / src / Main.java

Evgen959 newLes29 5 minutes ago

88 lines (74 loc) · 4.12 KB

Code Blame Raw

```
1   /*
2       2 Сложное (по желанию): Допустим дан List<model.Account>.
3       Класс model.Account определен так же как и в уроке 15:
4           private String iban;
5           private double balance;
6           private model.Person owner;
7           private model.MyDate openDate;
8
9       Ваша задача реализовать следующий функционал:
10          получить List<model.Account> всех счетов с балансом больше заданного числа
11          получить List<model.Account> всех счетов заданного владельца
12
13      Все методы, реализованные в задаче переписать как имплементацию интерфейса predicate.AccountPredic
14  */
15
16  import model.Account;
17  import model.MyDate;
18  import model.Person;
19  import predicate.*;
20
21  import java.util.ArrayList;
22  import java.util.List;
23
24  public class Main {
25      public static void main(String[] args) {
26
27          List<Account> accounts = List.of(
28                  new Account("DE0001", 1000.50, new Person("Jack", 20), new MyDate(10, 5, 2024)),
29                  new Account("DE0002", 8732.55, new Person("John", 28), new MyDate(1, 3, 2023)),
30                  new Account("DE0003", 7640.00, new Person("Bob", 23), new MyDate(19, 5, 2024)),
31                  new Account("DE0004", 12001.00, new Person("Bob", 23), new MyDate(11, 2, 2020)),
32                  new Account("DE0005", 123.00, new Person("Bob", 23), new MyDate(19, 5, 2018)),
33                  new Account("DE0006", 3800.01, new Person("Tom", 10), new MyDate(2, 5, 2020)),
34                  new Account("DE0007", 100.50, new Person("Alice", 16), new MyDate(6, 5, 2021)),
35                  new Account("DE0008", 300012.00, new Person("Nick", 32), new MyDate(7, 5, 2024))
36                  );
37
38          printAccounts(accounts);
39          System.out.println("------getAccountsWithMoreTHanGivenLimit----------");
40          printAccounts(getAccountsWithMoreThanGivenLimit(accounts, 1500));
41          System.out.println();
42          System.out.println("------------getAccountsByOwner------------");
```

```
43              printAccounts(getAccountsByOwner(accounts, new Person("Bob", 23)));
44              System.out.println();
45              //System.out.println("----------getAccountsByYear-----------");
46              //printAccounts(getAccountsByOpenYear(accounts, 2024));
47              System.out.println("---------PredicateByYear--------------");
48              printAccounts(filterAccounts(accounts, new PredicateByYear(2024)));
49
50              System.out.println();
51              System.out.println("---------------By Age------------------");
52              //printAccounts(getAccountsByOwnerAge(accounts, 20));
53              //printAccounts(filterAccounts(accounts, new PredicateAccountsByAge(25)));
54              printAccounts(filterAccounts(accounts, account->account.getOwner().getAge()>25));
55
56              System.out.println("--------------PredicateAccountsByOwnerName----------------");
57              printAccounts(filterAccounts(accounts, new PredicateAccountsByOwnerName("Bob")));
58
59              System.out.println("--------------predicate.PredicateAccountsByAge-----------------");
60              printAccounts(filterAccounts(accounts, new PredicateAccountsByAge(15)));
61
62          }
63          public  static void printAccounts(List<Account> accounts){
64              for (Account account: accounts){
65                  System.out.println(account);
66              }
67          }
68
69          public static  List<Account> getAccountsWithMoreThanGivenLimit(List<Account> list, double limitBal
70              PredicateAccountByGivenLimit predicate = new PredicateAccountByGivenLimit(limitBalance);
71              return filterAccounts(list,predicate);
72          }
73
74          public static List<Account> getAccountsByOwner (List<Account> list, Person owner){
75              PredicateAccountsByOwner predicate = new PredicateAccountsByOwner(owner);
76              return filterAccounts(list,predicate);
77          }
78
79          public static List<Account> filterAccounts(List<Account> list, AccountPredicate predicate){
80              List<Account> result = new ArrayList<>();
81              for (Account account: list){
82                  if (predicate.test(account)){       //!!!!!!!!!!!!!!!
83                      result.add(account);
84                  }
85              }
86              return result;
87          }
88      }
```