Evgen959 newHWLes28 1 minute ago

120 lines (102 loc) · 5.25 KB

Code Blame Raw

```
/*
    2 Сложное (по желанию): Допустим дан List<Account>.
    Класс Account определен так же как и в уроке 15:
        private String iban;
        private double balance;
        private Person owner;
        private MyDate openDate;

    Ваша задача реализовать следующий функционал:
        получить List<Account> всех счетов с балансом больше заданного числа
        получить List<Account> всех счетов заданного владельца

    Все методы, реализованные в задаче переписать как имплементацию интерфейса AccountPredicate
*/

import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) {

        List<Account> accounts = List.of(
                new Account("DE0001", 1000.50, new Person("Jack", 20), new MyDate(10, 5, 2024)),
                new Account("DE0002", 8732.55, new Person("John", 28), new MyDate(1, 3, 2023)),
                new Account("DE0003", 7640.00, new Person("Bob", 23), new MyDate(19, 5, 2024)),
                new Account("DE0004", 12001.00, new Person("Bob", 23), new MyDate(11, 2, 2020)),
                new Account("DE0005", 123.00, new Person("Bob", 23), new MyDate(19, 5, 2018)),
                new Account("DE0006", 3800.01, new Person("Tom", 10), new MyDate(2, 5, 2020)),
                new Account("DE0007", 100.50, new Person("Alice", 16), new MyDate(6, 5, 2021)),
                new Account("DE0008", 300012.00, new Person("Nick", 32), new MyDate(7, 5, 2024))
                );

        printAccounts(accounts);
        System.out.println("------getAccountsWithMoreTHanGivenLimit----------");
        printAccounts(getAccountsWithMoreTHanGivenLimit(accounts, 1500));
        System.out.println("------Д/3-----PredicateAccountsWithMoreTHanGivenLimit----------");
        printAccounts(filterAccounts(accounts, new PredicateAccountsWithMoreTHanGivenLimit(1500)));


        System.out.println();
        System.out.println("------------getAccountsByOwner------------");
        printAccounts(getAccountsByOwner(accounts, new Person("Bob", 23)));
        System.out.println("------Д/3--------PredicateAccountsByOwner--------------");
        printAccounts(filterAccounts(accounts, new PredicateAccountsByOwner(new Person("Bob", 23))));

        System.out.println();
        System.out.println("----------getAccountsByYear-----------");
        printAccounts(getAccountsByOpenYear(accounts, 2024));
        System.out.println("------Д/3--------PredicateAccountsByYear--------------");
```

```
50          printAccounts(filterAccounts(accounts, new PredicateAccountsByYear(2024)));
51
52          System.out.println();
53          System.out.println("---------------getAccountsByOwnerAge----------------------");
54          printAccounts(getAccountsByOwnerAge(accounts, 20));
55
56          System.out.println("--------------PredicateAccountsByOwnerName----------------");
57          printAccounts(filterAccounts(accounts, new PredicateAccountsByOwnerName("Bob")));
58
59          System.out.println("--------------PredicateAccountsByAge-----------------");
60          printAccounts(filterAccounts(accounts, new PredicateAccountsByAge(25)));
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 limitBalance){
70          List<Account> result = new ArrayList<>();
71                                                          // Перебор, итерирование
72          for (Account account: list){
73                                                          // отбор элементов, условие, фильтр
74              if (account.getBalance()>limitBalance){
75                  result.add(account);                    // действие
76              }
77          }
78          return result;
79      }
80
81      public static List<Account> getAccountsByOwner (List<Account> list, Person owner){
82          List<Account> result = new ArrayList<>();
83          for (Account account: list){
84              if (account.getOwner().equals(owner)){
85                  result.add(account);
86              }
87          }
88          return result;
89      }
90
91      public static List<Account> getAccountsByOpenYear (List<Account> list, int year){
92          List<Account> result = new ArrayList<>();
93          for (Account account: list){
94              if (account.getOpenDate().getYear() == year){
95                  result.add(account);
96              }
97          }
98          return result;
99      }
100
101     public static List<Account> getAccountsByOwnerAge(List<Account> list, int age){
102         List<Account> result = new ArrayList<>();
103         for (Account account: list){
104             if (account.getOwner().getAge() < age){
105                 result.add(account);
106             }
107         }
108         return result;
109     }
110
111     public static List<Account> filterAccounts(List<Account> list, AccountPredicate predicate){
112         List<Account> result = new ArrayList<>();
113         for (Account account: list){
114             if (predicate.test(account)){
115                 result.add(account);
```

```
116                 }
117             }
118             return result;
119         }
120     }
```