Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

Лабороторная работа №1

Выполнил студент:

Карепин Денис Дмитриевич
группа: M32071

Проверил:

Чикишев Константин Максимович

Санкт-Петербург,
2022 г.

1.1. Текст задания

1 лабораторная
Есть несколько Банков, которые предоставляют финансовые услуги по операциям с деньгами.
В банке есть Счета и Клиенты. У клиента есть имя, фамилия, адрес и номер паспорта (имя и фамилия обязательны, остальное – опционально).
Счета и проценты
Счета бывают трёх видов: Дебетовый счет, Депозит и Кредитный счет. Каждый счет принадлежит какому-то клиенту.
Дебетовый счет – обычный счет с фиксированным процентом на остаток. Деньги можно снимать в любой момент, в минус уходить нельзя. Комиссий нет.
Депозитный счет – счет, с которого нельзя снимать и переводить деньги до тех пор, пока не закончится его срок (пополнять можно). Процент на остаток зависит от изначальной суммы, например, если открываем депозит до 50 000 р. - 3
Кредитный счет – имеет кредитный лимит, в рамках которого можно уходить в минус (в плюс тоже можно). Процента на остаток нет. Есть фиксированная комиссия за использование, если клиент в минусе.
Комиссии
Периодически банки проводят операции по выплате процентов и вычету комиссии. Это значит, что нужен механизм проматывания времени, чтобы посмотреть, что будет через день/месяц/год и т.п.
Процент на остаток начисляется ежедневно от текущей суммы в этот день, но выплачивается раз в месяц (и для дебетовой карты и для депозита). Например, 3.65
Разные банки предлагают разные условия. В каждом банке известны величины процентов и комиссий.
Центральный банк
Регистрацией всех банков, а также взаимодействием между банками занимается центральный банк. Он должен управлять банками (предоставлять возможность создать банк) и предоставлять необходимый функционал, чтобы банки могли взаимодействовать с другими банками (например, можно реализовать переводы между банками через него). Он также занимается уведомлением других банков о том, что нужно начислять остаток или комиссию - для этого механизма не требуется создавать таймеры и завязываться на реальное время.
Операции и транзакции
Каждый счет должен предоставлять механизм снятия, пополнения и перевода денег (то есть счетам нужны некоторые идентификаторы).
Еще обязательный механизм, который должны иметь банки - отмена транзакций. Если вдруг выяснится, что транзакция была совершена злоумышленником, то такая транзакция должна быть отменена. Отмена транзакции подразумевает возвращение банком суммы обратно. Транзакция не может быть повторно

отменена.
Создание клиента и счета
Клиент должен создаваться по шагам. Сначала он указывает имя и фамилию (обязательно), затем адрес (можно пропустить и не указывать), затем паспортные данные (можно пропустить и не указывать).
Если при создании счета у клиента не указаны адрес или номер паспорта, мы объявляем такой счет (любого типа) сомнительным, и запрещаем операции снятия и перевода выше определенной суммы (у каждого банка своё значение). Если в дальнейшем клиент указывает всю необходимую информацию о себе - счет перестает быть сомнительным и может использоваться без ограничений.
Обновление условий счетов
Для банков требуется реализовать методы изменений процентов и лимитов не перевод. Также требуется реализовать возможность пользователям подписываться на информацию о таких изменениях - банк должен предоставлять возможность клиенту подписаться на уведомления. Стоит продумать расширяемую систему, в которой могут появится разные способы получения нотификаций клиентом (да, да, это референс на тот самый сайт). Например, когда происходит изменение лимита для кредитных карт – все пользователи, которые подписались и имеют кредитные карты, должны получить уведомление.
Консольный интерфейс работы
Для взаимодействия с банком требуется реализовать консольный интерфейс, который будет взаимодействовать с логикой приложения, отправлять и получать данные, отображать нужную информацию и предоставлять интерфейс для ввода информации пользователем.
Дополнения
На усмотрение студента можно ввести свои дополнительные идентификаторы для пользователей, банков etc.
На усмотрение студента можно пользователю добавить номер телефона или другие характеристики, если есть понимание зачем это нужно.
QnA
Q: Нужно ли предоставлять механизм отписки от информации об изменениях в условии счетов
A: Не обговорено, значит на ваше усмотрение (это вообще не критичный момент судя по условию лабы)
Q: Транзакциями считаются все действия со счётом, или только переводы между счетами. Если 1, то как-то странно поддерживать отмену операции снятия, а то после отмены деньги удвоятся: они будут и у злоумышленника на руках и на счету. Или просто на это забить
A: Все операции со счетами - транзакции.
Q: Фиксированная комиссия за использование кредитного счёта, когда тот в минусе измеряется в

A: Фиксированная комиссия означает, что это фиксированная сумма, а не процент. Да, при отмене транзакции стоит учитывать то, что могла быть также комиссия.
Q: Если транзакция подразумевает возвращение суммы обратно - но при этом эта же сумма была переведена на несколько счетов (пример перевод денег со счета 1 на счёт 2, со счёта 2 на счёт 3) Что происходит если клиент 1 отменяет транзакцию?
Подразумевается ли что деньги по цепочке снимаются со счёта 3? (на счету 2 их уже физически нет) Либо у нас банк мошеннический и деньги "отмываются"и возмещаются клиенту 1 с уводом счёта 2 в минус
A: Банк не мошеннический, просто упрощённая система. Транзакции не связываются между собой. Так что да, можно считать, что может уйти в минус.
Иными словами: переписать лабораторную 4 из курса по ООП на Java

1.2. Решение

Листинг 1.1: mainBank.java

```
1  import tools.CentralBankException;
2  import tools.ConsoleInterface;
3
4  public class mainBank {
5      public static void main(String[] args) throws CentralBankException
       {
6          var cons = new ConsoleInterface();
7          ConsoleInterface.input();
8
9      }
10 }
```

Листинг 1.2: AccountOption.java

```
1 package accountServices;
2
3 public enum AccountOption {
4     Deposit,
5     Debit,
6     Credit,
7 }
```

Листинг 1.3: CreditAccount.java

```
1  package accountServices;
2
3  import banksServices.Bank;
4  import clientServices.Client;
5  import tools.CentralBankException;
6
7  import java.util.UUID;
8
9  public class CreditAccount implements IAccount {
10     private double balance;
11     private double commissionUsing;
12     private double creditLimit;
13     private String numberOfAccount;
14     private boolean verification;
15     public Bank belongBank;
16
17     public CreditAccount(Client user, Bank bank, double amount) throws
       CentralBankException {
18         if (bank == null) {
19             throw new CentralBankException("null bank");
20         }
21         if (user == null) {
22             throw new CentralBankException("null client");
23         }
24         verification = user.getVerification();
25         commissionUsing = bank.getCommissionUsingForCreditAccounts();
26         creditLimit = bank.getCreditLimitForCreditAccounts();
27         balance = amount;
28         belongBank = bank;
29         numberOfAccount = UUID.randomUUID().toString();
30     }
31
32     public void withdrawalMoney(double amount) {
33         if (balance - amount > -creditLimit) {
34             balance -= amount;
35         }
36     }
37
38     public void replenishmentMoney(double amount) {
39         balance += amount;
40     }
41
42     public void transferMoney(IAccount account, double amount) {
43         withdrawalMoney(amount);
44         account.replenishmentMoney(amount);
45     }
46
47     public void actionWithAccount() {
48         if ((balance < 0) && (balance - commissionUsing >= -
```

```
    creditLimit)) {
49              balance -= commissionUsing;
50          }
51      }
52
53      public String getIdAccount() {
54          return numberOfAccount;
55      }
56
57      public boolean checkVerification() {
58          return verification;
59      }
60
61      public Bank getBelongBank() {
62          return belongBank;
63      }
64  }
```

Листинг 1.4: DebitAccount.java

```
1  package accountServices;
2
3  import banksServices.Bank;
4  import clientServices.Client;
5  import tools.CentralBankException;
6
7  import java.util.UUID;
8
9  public class DebitAccount implements IAccount {
10     private double balance;
11     private double percentageOnBalance;
12     private String numberOfAccount;
13     private boolean verification;
14
15     public DebitAccount(Client user, Bank bank, double amount) throws
       CentralBankException {
16         if (bank == null) {
17             throw new CentralBankException("null bank");
18         }
19         if (user == null) {
20             throw new CentralBankException("null client");
21         }
22         verification = user.getVerification();
23         percentageOnBalance = bank.
       getPercentageOnBalanceForDebitAccounts();
24         BelongBank = bank;
25         balance = amount;
26         numberOfAccount = UUID.randomUUID().toString();
27     }
28
29     public Bank BelongBank;
30
31     public void withdrawalMoney(double amount) {
32         balance -= amount;
33     }
34
35     public void replenishmentMoney(double amount) {
36         balance += amount;
37     }
38
39     public void transferMoney(IAccount account, double amount) {
40         withdrawalMoney(amount);
41         account.replenishmentMoney(amount);
42     }
43
44     public void actionWithAccount() {
45         balance += balance * percentageOnBalance / 100;
46     }
47
```

```
48      public String getIdAccount() {
49          return numberOfAccount;
50      }
51
52      public boolean checkVerification() {
53          return verification;
54      }
55
56      public Bank getBelongBank() {
57          return BelongBank;
58      }
59  }
```

Листинг 1.5: DepositAccount.java

```
1  package accountServices;
2
3  import banksServices.Bank;
4  import clientServices.Client;
5  import tools.CentralBankException;
6  import tools.Pair;
7
8  import java.util.UUID;
9
10 public class DepositAccount implements IAccount {
11     private double balance;
12     private double percentage;
13     private String numberOfAccount;
14     private boolean verification;
15
16     public DepositAccount(Client user, Bank bank, double amount)
       throws CentralBankException {
17         if (bank == null) {
18             throw new CentralBankException("null bank");
19         }
20         if (user == null) {
21             throw new CentralBankException("null client");
22         }
23         verification = user.getVerification();
24         percentage = bank.getPercentageOnBalanceForDepositAccounts().
25                 getPairsSumAndPercent().stream().filter(x -> x.getSum
       () > amount).findFirst().orElse(new Pair(0.0, 0.0)).getPercentage()
       ;
26         balance = amount;
27         BelongBank = bank;
28         numberOfAccount = UUID.randomUUID().toString();
29     }
30
31     public Bank BelongBank;
32
33     public void withdrawalMoney(double amount) {
34         balance -= amount;
35     }
36
37     public void replenishmentMoney(double amount) {
38         balance += amount;
39     }
40
41     public void transferMoney(IAccount account, double amount) {
42     }
43
44     public void actionWithAccount() {
45         balance += balance * percentage / 100;
46     }
```

```
47
48      public String getIdAccount() {
49          return numberOfAccount;
50      }
51
52      public boolean checkVerification() {
53          return verification;
54      }
55
56      public Bank getBelongBank() {
57          return BelongBank;
58      }
59  }
```

Листинг 1.6: DepositAccountPercentage.java

```
1  package accountServices;
2
3  import tools.Pair;
4
5  import java.util.ArrayList;
6
7  public class DepositAccountPercentage {
8      private ArrayList<Pair> pairsSumAndPercent;
9
10     public DepositAccountPercentage() {
11         pairsSumAndPercent = new ArrayList<Pair>();
12     }
13
14     public void addParametersForDepositAccountBank(double sum, double
       percentage) {
15         pairsSumAndPercent.add(new Pair(sum, percentage));
16     }
17
18     public ArrayList<Pair> getPairsSumAndPercent() {
19         return pairsSumAndPercent;
20     }
21
22     public void setPairsSumAndPercent(ArrayList<Pair> value) {
23         this.pairsSumAndPercent = value;
24     }
25
26 }
```

Листинг 1.7: IAccount.java

```
1  package accountServices;
2
3  import banksServices.Bank;
4
5  public interface IAccount {
6      void withdrawalMoney(double amount);
7
8      void replenishmentMoney(double amount);
9
10     void transferMoney(IAccount account, double amount);
11
12     void actionWithAccount();
13
14     String getIdAccount();
15
16     boolean checkVerification();
17
18     Bank getBelongBank();
19 }
```

Листинг 1.8: Bank.java

```
1  package banksServices;
2
3  import accountServices.DepositAccountPercentage;
4  import accountServices.IAccount;
5  import clientServices.Client;
6  import tools.CentralBankException;
7  import tools.EventRegistrar;
8
9  import java.util.*;
10
11 public class Bank {
12     public EventRegistrar events;
13     private HashMap<Client, ArrayList<IAccount>> baseBank;
14     private ArrayList<Transaction> transactions;
15     private double limitForNotVerification;
16     private double creditLimitForCreditAccounts;
17     private double commissionUsingForCreditAccounts;
18     private double percentageOnBalanceForDebitAccounts;
19     private String name;
20     private DepositAccountPercentage
       percentageOnBalanceForDepositAccounts;
21
22     public Bank(String name, double limitForNotVerification, double
       creditLimitForCreditAccounts, double
       commissionUsingForCreditAccounts,
23                 DepositAccountPercentage
       percentageOnBalanceForDepositAccounts, double
       percentageOnBalanceForDebitAccounts) throws CentralBankException {
24         if (name == null) throw new CentralBankException("Incorrect
       name");
25         this.name = name;
26         this.limitForNotVerification = limitForNotVerification;
27         this.creditLimitForCreditAccounts =
       creditLimitForCreditAccounts;
28         this.commissionUsingForCreditAccounts =
       commissionUsingForCreditAccounts;
29         this.percentageOnBalanceForDepositAccounts =
       percentageOnBalanceForDepositAccounts;
30         this.percentageOnBalanceForDebitAccounts =
       percentageOnBalanceForDebitAccounts;
31         this.baseBank = new HashMap<>();
32         this.transactions = new ArrayList<>();
33         this.events = new EventRegistrar("Change name",
34                 "Change creditLimitForCreditAccounts",
35                 "Change commissionUsingForCreditAccounts",
36                 "Change limitForNotVerification",
37                 "Change percentageOnBalanceForDebitAccounts",
38                 "Change percentageOnBalanceForDepositAccounts");
39     }
```

```
40
41      // public delegate void ChangeFieldInBanks(double other);
42      // public event ChangeFieldInBanks ChangeFieldInBank;
43
44      public String getName() {
45          return name;
46      }
47
48      public void setName(String name) {
49          events.notify("Change name");
50          this.name = name;
51      }
52
53      public double getCreditLimitForCreditAccounts() {
54          return creditLimitForCreditAccounts;
55      }
56
57      public void setCreditLimitForCreditAccounts(double value) {
58          events.notify("Change creditLimitForCreditAccounts");
59          this.creditLimitForCreditAccounts = value;
60      }
61
62      public double getCommissionUsingForCreditAccounts() {
63          return commissionUsingForCreditAccounts;
64      }
65
66      public void setCommissionUsingForCreditAccounts(double value) {
67          events.notify("Change commissionUsingForCreditAccounts");
68          this.commissionUsingForCreditAccounts = value;
69      }
70
71      public double getLimitForNotVerification() {
72          return limitForNotVerification;
73      }
74
75      public void setLimitForNotVerification(double value) {
76          events.notify("Change limitForNotVerification");
77          this.limitForNotVerification = value;
78      }
79
80      public double getPercentageOnBalanceForDebitAccounts() {
81          return percentageOnBalanceForDebitAccounts;
82      }
83
84      public void setPercentageOnBalanceForDebitAccounts(double value) {
85          events.notify("Change percentageOnBalanceForDebitAccounts");
86          this.percentageOnBalanceForDebitAccounts = value;
87      }
88
89      public DepositAccountPercentage
```

```
    getPercentageOnBalanceForDepositAccounts() {
90          return percentageOnBalanceForDepositAccounts;
91      }
92
93      public void setPercentageOnBalanceForDepositAccounts(
    DepositAccountPercentage value) {
94          events.notify("Change percentageOnBalanceForDepositAccounts");
95          this.percentageOnBalanceForDepositAccounts = value;
96      }
97
98      public void registerClient(Client client, IAccount account) throws
     CentralBankException {
99          if (client == null) throw new CentralBankException("Incorrect
    client");
100         if (account == null) throw new CentralBankException("Incorrect
     account");
101         if (baseBank.containsKey(client))
102             baseBank.get(client).add(account);
103         else
104             baseBank.put(client, new ArrayList<IAccount>(List.of(
    account)));
105         client.createAccount(this, getInfoAccounts(client));
106     }
107
108     public IAccount findAccount(String numberId) throws
    CentralBankException {
109         if (numberId == null) throw new CentralBankException("
    Incorrect numberId");
110         return baseBank.values()
111                 .stream()
112                 .flatMap(Collection::stream)
113                 .filter(i -> Objects.equals(i.getIdAccount(), numberId
    ))
114                 .findFirst()
115                 .orElse(null);
116     }
117
118     public void accruePercentage() {
119         baseBank.values().stream().flatMap(Collection::stream).forEach
    (IAccount::actionWithAccount);
120     }
121
122     public void addTransaction(Transaction transaction) {
123         transactions.add(transaction);
124     }
125
126     private ArrayList<IAccount> getInfoAccounts(Client client) {
127         return baseBank.getOrDefault(client, null);
128     }
129 }
```

Листинг 1.9: CentralBank.java

```
1  package banksServices;
2
3  import accountServices.*;
4  import clientServices.Client;
5  import tools.CentralBankException;
6
7  import java.util.ArrayList;
8  import java.util.List;
9  import java.util.Objects;
10
11 public class CentralBank {
12     private List<Bank> banks;
13     private List<Transaction> transactions;
14
15     public CentralBank() {
16         banks = new ArrayList<Bank>();
17         transactions = new ArrayList<Transaction>();
18     }
19
20     public Bank addBankToBase(String name, double
       limitForNotVerification, double creditLimitForCreditAccounts,
       double commissionUsingForCreditAccounts, DepositAccountPercentage
       percentageOnBalanceForDepositAccounts, double
       percentageOnBalanceForDebitAccounts) throws CentralBankException {
21         banks.add(new Bank(name, limitForNotVerification,
       creditLimitForCreditAccounts, commissionUsingForCreditAccounts,
       percentageOnBalanceForDepositAccounts,
       percentageOnBalanceForDebitAccounts));
22         return banks.get(banks.size() - 1);
23     }
24
25     public IAccount regAccountClientInBank(Bank bank, Client client,
       AccountOption option, double amount) throws CentralBankException {
26         if (bank == null) {
27             throw new CentralBankException("null bank");
28         }
29         if (client == null) {
30             throw new CentralBankException("null client");
31         }
32         if (!banks.contains(bank)) {
33             throw new CentralBankException("Bank dont registered");
34         }
35         if (amount < 0) {
36             throw new CentralBankException("Negative balance");
37         }
38         IAccount account;
39         switch (option) {
40             case Credit -> {
41                 account = new CreditAccount(client, bank, amount);
```

```
42                  bank.registerClient(client, account);
43                  return account;
44              }
45              case Deposit -> {
46                  account = new DepositAccount(client, bank, amount);
47                  bank.registerClient(client, account);
48                  return account;
49              }
50              case Debit -> {
51                  account = new DebitAccount(client, bank, amount);
52                  bank.registerClient(client, account);
53                  return account;
54              }
55              default -> throw new CentralBankException("{option} -
    Incorrect options");
56          }
57      }
58
59      public Transaction withdrawalMoney(IAccount account, double amount
    ) throws CentralBankException {
60          if (!account.checkVerification()
61                  &&
62                  account.getBelongBank().getLimitForNotVerification() <
     amount) {
63              throw new CentralBankException("Attempt to withdraw money
    from an unverified account");
64          }
65          var tmpTransaction = new Transaction(account.getIdAccount(),
    null, amount);
66          transactions.add(tmpTransaction);
67          account.getBelongBank().addTransaction(tmpTransaction);
68          account.withdrawalMoney(amount);
69          return transactions.get(transactions.size() - 1);
70      }
71
72      public Transaction replenishmentMoney(IAccount account, double
    amount) {
73          var tmpTransaction = new Transaction(null, account.
    getIdAccount(), amount);
74          transactions.add(tmpTransaction);
75          account.getBelongBank().addTransaction(tmpTransaction);
76          account.replenishmentMoney(amount);
77          return transactions.get(transactions.size() - 1);
78      }
79
80      public Transaction transferMoney(IAccount account1, IAccount
    account2, double amount) throws CentralBankException {
81          if (!account1.checkVerification()
82                  &&
83                  account1.getBelongBank().getLimitForNotVerification()
```

```
< amount) {
84              throw new CentralBankException("Attempt to withdraw money
    from an unverified account");
85          }
86          var tmpTransaction = new Transaction(account1.getIdAccount(),
    account2.getIdAccount(), amount);
87          transactions.add(tmpTransaction);
88          account1.getBelongBank().addTransaction(tmpTransaction);
89          account2.getBelongBank().addTransaction(tmpTransaction);
90          account1.transferMoney(account2, amount);
91          return transactions.get(transactions.size() - 1);
92      }
93
94      public void cancelTransaction(Transaction transaction) throws
    CentralBankException {
95          if (transaction == null) {
96              throw new CentralBankException("Incorrect transaction");
97          }
98          IAccount tmpTransferAccount = null;
99          IAccount tmpWithdrawalAccount = null;
100         if (transaction.getTransferAccount() != null && transaction.
    getWithdrawalAccount() != null) {
101             for (Bank bank : banks) {
102                 tmpTransferAccount = bank.findAccount(transaction.
    getTransferAccount());
103                 tmpWithdrawalAccount = bank.findAccount(transaction.
    getWithdrawalAccount());
104                 if (tmpTransferAccount != null && tmpWithdrawalAccount
     != null) break;
105             }
106
107             Objects.requireNonNull(tmpTransferAccount).
    replenishmentMoney(transaction.getAmount());
108             Objects.requireNonNull(tmpWithdrawalAccount).
    withdrawalMoney(transaction.getAmount());
109         } else if (transaction.getWithdrawalAccount() == null &&
    transaction.getTransferAccount() != null) {
110             for (Bank bank : banks) {
111                 tmpTransferAccount = bank.findAccount(transaction.
    getTransferAccount());
112                 if (tmpTransferAccount != null) break;
113             }
114
115             Objects.requireNonNull(tmpTransferAccount).withdrawalMoney
    (transaction.getAmount());
116         } else if (transaction.getTransferAccount() == null &&
    transaction.getWithdrawalAccount() != null) {
117             for (Bank bank : banks) {
118                 tmpWithdrawalAccount = bank.findAccount(transaction.
    getWithdrawalAccount());
```

```
119                 if (tmpWithdrawalAccount != null) break;
120             }
121
122             Objects.requireNonNull(tmpWithdrawalAccount).
    replenishmentMoney(transaction.getAmount());
123         }
124
125         transactions.remove(transaction);
126     }
127
128     public void manageTime(int countOfDay) {
129         for (int i = 0; i < countOfDay % 30; i++) {
130             for (Bank bank : banks) {
131                 bank.accruePercentage();
132             }
133         }
134     }
135
136     public boolean findBank(Bank bank) {
137         return banks.contains(bank);
138     }
139 }
```

Листинг 1.10: Transaction.java

```
package banksServices;

public class Transaction {
    private String withdrawalAccount;
    private String transferAccount;
    private double amount;

    public Transaction(String withdrawalAccount, String
   transferAccount, double amount) {
        this.withdrawalAccount = withdrawalAccount;
        this.transferAccount = transferAccount;
        this.amount = amount;
    }

    public String getWithdrawalAccount() {
        return withdrawalAccount;
    }

    public void setWithdrawalAccount(String value) {
        this.withdrawalAccount = value;
    }

    public String getTransferAccount() {
        return transferAccount;
    }

    public void setTransferAccount(String value) {
        this.transferAccount = value;
    }

    public double getAmount() {
        return amount;
    }

    public void setAmount(double value) {
        this.amount = value;
    }

    public String toString() {
        return this.withdrawalAccount + " to " + this.transferAccount
   + "of" + this.amount;
    }
}
```

Листинг 1.11: Client.java

```
1  package clientServices;
2
3  import accountServices.IAccount;
4  import banksServices.Bank;
5  import tools.CentralBankException;
6
7  import java.util.ArrayList;
8  import java.util.HashMap;
9  import java.util.UUID;
10
11
12 public class Client {
13     private HashMap<Bank, ArrayList<IAccount>>
       clientCollectionAccounts;
14     private UUID id;
15     private String name;
16     private String surname;
17     private String address;
18     private String passport;
19     private boolean isAllInfo;
20
21     public Client(String name, String surname, String address, String
       passport) throws CentralBankException {
22         this.id = UUID.randomUUID();
23         if (name.isBlank()) {
24             throw new CentralBankException("Incorrect name");
25         }
26         this.name = name;
27         if (surname.isBlank()) {
28             throw new CentralBankException("Incorrect surname");
29         }
30         this.surname = surname;
31         this.address = address;
32         this.passport = passport;
33         this.clientCollectionAccounts = new HashMap<Bank, ArrayList<
       IAccount>>();
34         isAllInfo = this.address != null && this.passport != null && !
       this.address.isBlank() && !this.passport.isBlank();
35     }
36
37     public static void update(String other) {
38         System.out.print(other);
39     }
40
41     public static ClientBuilder Builder(String name, String surname)
       throws CentralBankException {
42         return new ClientBuilder().addName(name).addSurname(surname);
43     }
44
```

```
45      public boolean getVerification() {
46          return isAllInfo;
47      }
48
49
50      public void createAccount(Bank bank, ArrayList<IAccount> accounts)
        {
51          if (!clientCollectionAccounts.containsKey(bank))
52              clientCollectionAccounts.put(bank, accounts);
53          else {
54              clientCollectionAccounts.get(bank).addAll(accounts);
55          }
56      }
57  }
```

Листинг 1.12: ClientBuilder.java

```
1  package clientServices;
2
3  import tools.CentralBankException;
4
5  public class ClientBuilder {
6      private String name;
7      private String surname;
8      private String address;
9      private String passport;
10
11     public ClientBuilder addName(String name) throws
       CentralBankException {
12         if ((name == null) || (name.isBlank())) {
13             throw new CentralBankException("Incorrect name");
14         }
15         this.name = name;
16         return this;
17     }
18
19     public ClientBuilder addSurname(String surname) throws
       CentralBankException {
20         if ((surname == null) || (surname.isBlank())) {
21             throw new CentralBankException("Incorrect surname");
22         }
23         this.surname = surname;
24         return this;
25     }
26
27     public ClientBuilder addAddress(String address) throws
       CentralBankException {
28         if ((address == null) || (address.isBlank())) {
29             throw new CentralBankException("Incorrect address");
30         }
31         this.address = address;
32
33         return this;
34     }
35
36     public ClientBuilder addPassport(String passport) throws
       CentralBankException {
37         if ((passport == null) || (passport.isBlank())) {
38             throw new CentralBankException("Incorrect passport");
39         }
40         this.passport = passport;
41         return this;
42     }
43
44     public Client getClient() throws CentralBankException {
45         return new Client(name, surname, address, passport);
```

```
46          }
47  }
```

Листинг 1.13: CentralBankException.java

```
1 package tools;
2
3 public class CentralBankException extends Throwable{
4         public CentralBankException(String message) {
5             super(message);
6     }
7 }
```

Листинг 1.14: ConsoleInterface.java

```
package tools;

import accountServices.AccountOption;
import accountServices.DepositAccountPercentage;
import accountServices.IAccount;
import banksServices.Bank;
import banksServices.CentralBank;
import banksServices.Transaction;
import clientServices.Client;

import java.util.Scanner;

public class ConsoleInterface {
    public static void input() throws CentralBankException {
        var centralBank = new CentralBank();
        Client client = null;
        IAccount account1 = null;
        IAccount account2 = null;
        Bank bank = null;
        Transaction transaction = null;
        String b = null;
        while (b != "Q") {
            System.out.println("1 - Start Create Bank");
            System.out.println("2 - Start Create Client");
            System.out.println("3 - Start Create Account");
            System.out.println("4 - Start Do Transaction");
            System.out.println("5 - Start Canceled Transaction");
            System.out.println("Q - Exit Program");
            Scanner in = new Scanner(System.in);
            b = in.nextLine();

            switch (b) {
                case "1": {
                    System.out.println("Set Name\nExample: string");
                    String name = in.nextLine();
                    System.out.println("Set limitForNotVerification\
nExample: double");
                    double limitForNotVerification = in.nextDouble();
                    System.out.println("Set
creditLimitForCreditAccounts\nExample: double");
                    double creditLimitForCreditAccounts = in.
nextDouble();
                    System.out.println("Set
commissionUsingForCreditAccounts\nExample: double");
                    double commissionUsingForCreditAccounts = in.
nextDouble();
                    System.out.println("Set
percentageOnBalanceForDepositAccounts\nExample: 50000 3\nExample:
100000 5\nExample:1000000 6");
```

```
43                  DepositAccountPercentage
    percentageOnBalanceForDepositAccounts = new
    DepositAccountPercentage();
44                  for (int i = 0; i < 3; i++) {
45                      Scanner in2 = new Scanner(System.in);
46                      String str = in2.nextLine();
47                      double first = Double.parseDouble(str.split("
    ")[0]);
48                      double second = Double.parseDouble(str.split("
     ")[1]);
49                      percentageOnBalanceForDepositAccounts.
    addParametersForDepositAccountBank(first, second);
50                  }
51
52                  System.out.println("Set
    percentageOnBalanceForDebitAccounts\nExample: double");
53                  double percentageOnBalanceForDebitAccounts = in.
    nextDouble();
54                  bank = centralBank.addBankToBase(
55                          name,
56                          limitForNotVerification,
57                          creditLimitForCreditAccounts,
58                          commissionUsingForCreditAccounts,
59                          percentageOnBalanceForDepositAccounts,
60                          percentageOnBalanceForDebitAccounts);
61                  System.out.println("Done");
62                  break;
63              }
64
65              case "2":
66                  System.out.println("1 - Start Create Client");
67                  System.out.println("2 - Start Create Verification
    Client");
68                  b = in.nextLine();
69                  switch (b) {
70                      case "1": {
71                          System.out.println("Set name\nExample:
    string");
72                          String name = in.nextLine();
73                          System.out.println("Set surname\nExample:
    string");
74                          String surname = in.nextLine();
75                          client = Client.Builder(name, surname).
    getClient();
76                          System.out.println("Done");
77                          break;
78                      }
79
80                      case "2": {
81                          System.out.println("Set name\nExample:
```

```
    string");
82                              String name = in.nextLine();
83                              System.out.println("Set surname\nExample:
    string");
84                              String surname = in.nextLine();
85                              System.out.println("Set address\nExample:
    string");
86                              String address = in.nextLine();
87                              System.out.println("Set passport\nExample:
     string");
88                              String passport = in.nextLine();
89                              client = Client.Builder(name, surname).
    addAddress(address).addPassport(passport).getClient();
90                              System.out.println("Done");
91                              break;
92                          }
93                      }
94
95                      break;
96                  case "3":
97                      System.out.println("1 - Start Create Debit");
98                      System.out.println("2 - Start Create Credit");
99                      System.out.println("3 - Start Create Deposit");
100                     b = in.nextLine();
101                     switch (b) {
102                         case "1":
103                             account1 = centralBank.
    regAccountClientInBank(bank, client, AccountOption.Debit, 10000);
104                             account2 = centralBank.
    regAccountClientInBank(bank, client, AccountOption.Debit, 10000);
105                             System.out.println("Done");
106                             break;
107                         case "2":
108                             account1 = centralBank.
    regAccountClientInBank(bank, client, AccountOption.Credit, 10000);
109                             account2 = centralBank.
    regAccountClientInBank(bank, client, AccountOption.Credit, 10000);
110                             System.out.println("Done");
111                             break;
112                         case "3":
113                             account1 = centralBank.
    regAccountClientInBank(bank, client, AccountOption.Deposit, 10000);
114                             account2 = centralBank.
    regAccountClientInBank(bank, client, AccountOption.Deposit, 10000);
115                             System.out.println("Done");
116                             break;
117                     }
118
119                     break;
120                 case "4":
```

```
121                     System.out.println("Set Amount Transaction\
    nExample: double");
122                     double amount = in.nextDouble();
123                     System.out.println("1 - Do Replenishment Money
    Transaction");
124                     System.out.println("2 - Do Withdrawal Money
    Transaction");
125                     System.out.println("3 - Do Transfer Money
    Transaction");
126                     b = in.nextLine();
127                     switch (b) {
128                         case "1":
129                             transaction = centralBank.
    replenishmentMoney(account1, amount);
130                             System.out.println("Done");
131                             break;
132                         case "2":
133                             transaction = centralBank.withdrawalMoney(
    account1, amount);
134                             System.out.println("Done");
135                             break;
136                         case "3":
137                             transaction = centralBank.transferMoney(
    account1, account2, amount);
138                             System.out.println("Done");
139                             break;
140                     }
141
142                     break;
143                 case "5":
144                     centralBank.cancelTransaction(transaction);
145                     System.out.println("Done");
146                     break;
147             }
148         }
149     }
150 }
```

Листинг 1.15: EventRegistrar.java

```
1  package tools;
2
3  import clientServices.Client;
4
5  import java.util.ArrayList;
6  import java.util.HashMap;
7
8  public class EventRegistrar {
9      HashMap<String, ArrayList<Client>> listeners = new HashMap<>();
10
11     public EventRegistrar(String... operations) {
12         for (String operation : operations) {
13             this.listeners.put(operation, new ArrayList<>());
14         }
15     }
16
17     public void subscribe(String eventType, Client listener) {
18         ArrayList<Client> users = listeners.get(eventType);
19         users.add(listener);
20     }
21
22     public void subscribeAll(Client listener) {
23         for (ArrayList<Client> clients : listeners.values())
24             clients.add(listener);
25     }
26
27     public void unsubscribe(String eventType, Client listener) {
28         ArrayList<Client> users = listeners.get(eventType);
29         users.remove(listener);
30     }
31
32     public void notify(String eventType) {
33         ArrayList<Client> users = listeners.get(eventType);
34         for (Client listener : users) {
35             Client.update(eventType);
36         }
37     }
38 }
```

Листинг 1.16: Pair.java

```
package tools;

public class Pair{
    private double sum;
    private double percentage;
    public Pair(double sum, double percentage){
        this.sum = sum;
        this.percentage = percentage;
    }
    public double getSum(){ return sum; }
    public double getPercentage(){ return percentage; }
    public void setSum(double sum){ this.sum = sum; }
    public void setPercentage(double percentage){ this.percentage =
   percentage; }
}
```