

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

Лабораторная работа №3

Выполнил студент:

Карепин Денис Дмитриевич
группа: М32071

Проверил:

Чижишев Константин Максимович

Санкт-Петербург,
2022 г.

1.1. Текст задания

3 лабораторная

К созданному в прошлой лабораторной сервису добавляется Spring.

Сервис должен предоставлять http интерфейс (REST API) для получения информации о конкретных котиках и владельцах и для получения фильтрованной информации (например, получить всех рыжих котиков)

Внимание: недопустимо отдавать через HTTP интерфейс сущности JPA. Рекомендуется создать отдельные оберточные классы.

Сервисы и dao должны превратиться в Spring Bean'ы с использованием Dependency Injection (Autowired). Dao при этом наследуют JpaRepository и имеет шаблонные Spring Data JPA методы: <https://www.baeldung.com/spring-data-repositoriesrepositories>

При сдаче лабораторной нужно будет показать работоспособность endpoint'ов через http запросы (рекомендуется Postman).

В рамках лабораторной к проекту должен быть добавлен CI/CD, запускающий тесты проекта kotiki-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     {  
7         var cons = new ConsoleInterface();  
8         ConsoleInterface.input();  
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
18     CentralBankException {
19         if (bank == null) {
20             throw new CentralBankException("null bank");
21         }
22         if (user == null) {
23             throw new CentralBankException("null client");
24         }
25         verification = user.getVerification();
26         commissionUsing = bank.getCommissionUsingForCreditAccounts();
27         creditLimit = bank.getCreditLimitForCreditAccounts();
28         balance = amount;
29         belongBank = bank;
30         numberOfAccount = UUID.randomUUID().toString();
31     }
32
33     public void withdrawalMoney(double amount) {
34         if (balance - amount > -creditLimit) {
35             balance -= amount;
36         }
37     }
38
39     public void replenishmentMoney(double amount) {
40         balance += amount;
41     }
42
43     public void transferMoney(IAccount account, double amount) {
44         withdrawalMoney(amount);
45         account.replenishmentMoney(amount);
46     }
47
48     public void actionWithAccount() {
49         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)
17     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         percentage = bank.getPercentageOnBalanceForDepositAccounts().
26             getPairsSumAndPercent().stream().filter(x -> x.getSum
27             () > amount).findFirst().orElse(new Pair(0.0, 0.0)).getPercentage();
28
29         balance = amount;
30         BelongBank = bank;
31         numberOfAccount = UUID.randomUUID().toString();
32     }
33
34     public Bank BelongBank;
35
36     public void withdrawalMoney(double amount) {
37         balance -= amount;
38     }
39
40     public void replenishmentMoney(double amount) {
41         balance += amount;
42     }
43
44     public void transferMoney(IAccount account, double amount) {
45     }
46
47     public void actionWithAccount() {
48         balance += balance * percentage / 100;
49     }
50 }
```

```
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() {  
    return percentageOnBalanceForDepositAccounts;  
}  
  
public void setPercentageOnBalanceForDepositAccounts(  
DepositAccountPercentage value) {  
    events.notify("Change percentageOnBalanceForDepositAccounts");  
    this.percentageOnBalanceForDepositAccounts = value;  
}  
  
public void registerClient(Client client, IAccount account) throws  
CentralBankException {  
    if (client == null) throw new CentralBankException("Incorrect  
client");  
    if (account == null) throw new CentralBankException("Incorrect  
account");  
    if (baseBank.containsKey(client))  
        baseBank.get(client).add(account);  
    else  
        baseBank.put(client, new ArrayList<IAccount>(List.of(  
account)));  
    client.createAccount(this, getInfoAccounts(client));  
}  
  
public IAccount findAccount(String numberId) throws  
CentralBankException {  
    if (numberId == null) throw new CentralBankException("  
Incorrect numberId");  
    return baseBank.values()  
        .stream()  
        .flatMap(Collection::stream)  
        .filter(i -> Objects.equals(i.getIdAccount(), numberId  
)  
        .findFirst()  
        .orElse(null);  
}  
  
public void accruePercentage() {  
    baseBank.values().stream().flatMap(Collection::stream).forEach  
(IAccount::actionWithAccount);  
}  
  
public void addTransaction(Transaction transaction) {  
    transactions.add(transaction);  
}  
  
private ArrayList<IAccount> getInfoAccounts(Client client) {  
    return baseBank.getDefault(client, null);  
}  
}
```

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

```
1 package banksServices;
2
3 public class Transaction {
4     private String withdrawalAccount;
5     private String transferAccount;
6     private double amount;
7
8     public Transaction(String withdrawalAccount, String
transferAccount, double amount) {
9         this.withdrawalAccount = withdrawalAccount;
10        this.transferAccount = transferAccount;
11        this.amount = amount;
12    }
13
14    public String getWithdrawalAccount() {
15        return withdrawalAccount;
16    }
17
18    public void setWithdrawalAccount(String value) {
19        this.withdrawalAccount = value;
20    }
21
22    public String getTransferAccount() {
23        return transferAccount;
24    }
25
26    public void setTransferAccount(String value) {
27        this.transferAccount = value;
28    }
29
30    public double getAmount() {
31        return amount;
32    }
33
34    public void setAmount(double value) {
35        this.amount = value;
36    }
37
38    public String toString() {
39        return this.withdrawalAccount + " to " + this.transferAccount
+ " of " + this.amount;
40    }
41 }
```

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>>
14     clientCollectionAccounts;
15     private UUID id;
16     private String name;
17     private String surname;
18     private String address;
19     private String passport;
20     private boolean isAllInfo;
21
22     public Client(String name, String surname, String address, String
23     passport) throws CentralBankException {
24         this.id = UUID.randomUUID();
25         if (name.isBlank()) {
26             throw new CentralBankException("Incorrect name");
27         }
28         this.name = name;
29         if (surname.isBlank()) {
30             throw new CentralBankException("Incorrect surname");
31         }
32         this.surname = surname;
33         this.address = address;
34         this.passport = passport;
35         this.clientCollectionAccounts = new HashMap<Bank, ArrayList<
36         IAccount>>();
37         isAllInfo = this.address != null && this.passport != null && !
38         this.address.isBlank() && !this.passport.isBlank();
39     }
40
41     public static void update(String other) {
42         System.out.print(other);
43     }
44
45     public static ClientBuilder Builder(String name, String surname)
46     throws CentralBankException {
47         return new ClientBuilder().addName(name).addSurname(surname);
48     }
49 }
```

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

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

```

1 package tools;
2
3 import accountServices.AccountOption;
4 import accountServices.DepositAccountPercentage;
5 import accountServices.IAccount;
6 import banksServices.Bank;
7 import banksServices.CentralBank;
8 import banksServices.Transaction;
9 import clientServices.Client;
10
11 import java.util.Scanner;
12
13 public class ConsoleInterface {
14     public static void input() throws CentralBankException {
15         var centralBank = new CentralBank();
16         Client client = null;
17         IAccount account1 = null;
18         IAccount account2 = null;
19         Bank bank = null;
20         Transaction transaction = null;
21         String b = null;
22         while (b != "Q") {
23             System.out.println("1 - Start Create Bank");
24             System.out.println("2 - Start Create Client");
25             System.out.println("3 - Start Create Account");
26             System.out.println("4 - Start Do Transaction");
27             System.out.println("5 - Start Canceled Transaction");
28             System.out.println("Q - Exit Program");
29             Scanner in = new Scanner(System.in);
30             b = in.nextLine();
31
32             switch (b) {
33                 case "1": {
34                     System.out.println("Set Name\nExample: string");
35                     String name = in.nextLine();
36                     System.out.println("Set limitForNotVerification\nExample: double");
37                     double limitForNotVerification = in.nextDouble();
38                     System.out.println("Set creditLimitForCreditAccounts\nExample: double");
39                     double creditLimitForCreditAccounts = in.nextDouble();
40                     System.out.println("Set commissionUsingForCreditAccounts\nExample: double");
41                     double commissionUsingForCreditAccounts = in.nextDouble();
42                     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

```
1 package tools;
2
3 public class Pair{
4     private double sum;
5     private double percentage;
6     public Pair(double sum, double percentage){
7         this.sum = sum;
8         this.percentage = percentage;
9     }
10    public double getSum(){ return sum; }
11    public double getPercentage(){ return percentage; }
12    public void setSum(double sum){ this.sum = sum; }
13    public void setPercentage(double percentage){ this.percentage =
14    percentage; }
```

Листинг 1.17: Application.java

```
1 package com.ferbator;  
2  
3 import org.springframework.boot.SpringApplication;  
4 import org.springframework.boot.autoconfigure.SpringBootApplication;  
5  
6 @SpringBootApplication  
7 public class Application {  
8     public static void main(String[] args) {  
9         SpringApplication.run(Application.class, args);  
10    }  
11 }
```


Листинг 1.18: CatController.java

```
1 package com.ferbator.controller;
2
3 import com.ferbator.dao.dto.CatDto;
4 import com.ferbator.dao.enums.Colors;
5 import com.ferbator.services.ShelterService;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.http.HttpStatus;
8 import org.springframework.web.bind.annotation.*;
9 import org.springframework.web.server.ResponseStatusException;
10
11 import java.util.List;
12
13 @RestController
14 @RequestMapping("/cat")
15 public class CatController {
16     @Autowired
17     ShelterService service;
18
19     @GetMapping("/find-all-cats")
20     public List<CatDto> findAllCats() {
21         return service.getListAllCats();
22     }
23
24     @GetMapping("/find-all-one-color-cats/{color}")
25     public List<CatDto> findAllOneColorCats(@PathVariable("color")
String color) {
26         try {
27             return service.getListAllOneColorCats(Colors.valueOf(color
));
28         } catch (IllegalArgumentException e) {
29             throw new ResponseStatusException(HttpStatus.BAD_REQUEST);
30         }
31     }
32
33     @PostMapping("/add-cat")
34     public boolean addCat(@RequestBody CatDto catDTO) {
35         return service.addCat(catDTO);
36     }
37
38     @DeleteMapping("/delete-cat/{id}")
39     public boolean delCatById(@PathVariable("id") Long id) {
40         return service.delCat(id);
41     }
42 }
```

Листинг 1.19: OwnerController.java

```
1 package com.ferbator.controller;
2
3 import com.ferbator.dao.dto.OwnerDto;
4 import com.ferbator.services.ShelterService;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.web.bind.annotation.*;
7
8 import java.util.List;
9
10 @RestController
11 @RequestMapping("/owner")
12 public class OwnerController {
13     @Autowired
14     ShelterService service;
15
16     @PostMapping("/add-owner")
17     public boolean addOwner(@RequestBody OwnerDto ownerDTO) {
18         return service.addOwner(ownerDTO);
19     }
20
21     @DeleteMapping("/delete-owner/{id}")
22     public boolean delOwnerById(@PathVariable("id") Long id) {
23         return service.delOwner(id);
24     }
25
26     @GetMapping("/find-all-owner")
27     public List<OwnerDto> findAllOwners() {
28         return service.getListAllOwners();
29     }
30 }
```

Листинг 1.20: CatRepository.java

```
1 package com.ferbator.dao.daoImpl;  
2  
3 import com.ferbator.dao.entities.Cat;  
4 import com.ferbator.dao.enums.Colors;  
5 import org.springframework.data.jpa.repository.JpaRepository;  
6 import org.springframework.stereotype.Repository;  
7  
8 import java.util.List;  
9  
10 @Repository  
11 public interface CatRepository extends JpaRepository<Cat, Long> {  
12     List<Cat> findAllByColor(Colors color);  
13 }
```

Листинг 1.21: FriendshipCatRepository.java

```
1 package com.ferbator.dao.daoImpl;  
2  
3 import com.ferbator.dao.entities.FriendshipCat;  
4 import org.springframework.data.jpa.repository.JpaRepository;  
5 import org.springframework.stereotype.Repository;  
6  
7 @Repository  
8 public interface FriendshipCatRepository extends JpaRepository<  
9     FriendshipCat, Long> {  
10     void deleteAllByFirstCatIdOrSecondCatId(Long firstCatId, Long  
11         secondCatId);  
12 }
```

Листинг 1.22: OwnerRepository.java

```
1 package com.ferbator.dao.daoImpl;  
2  
3 import com.ferbator.dao.entities.Owner;  
4 import org.springframework.data.jpa.repository.JpaRepository;  
5 import org.springframework.stereotype.Repository;  
6  
7 @Repository  
8 public interface OwnerRepository extends JpaRepository<Owner, Long> {  
9 }
```

Листинг 1.23: OwnershipCatRepository.java

```
1 package com.ferbator.dao.daoImpl;  
2  
3 import com.ferbator.dao.entities.OwnershipCat;  
4 import org.springframework.data.jpa.repository.JpaRepository;  
5 import org.springframework.stereotype.Repository;  
6  
7  
8 @Repository  
9 public interface OwnershipCatRepository extends JpaRepository<  
10     OwnershipCat, Long> {  
11     void deleteAllByCatId(Long catId);  
12     void deleteAllByOwnerId(Long ownerId);  
13 }
```

Листинг 1.24: CatDto.java

```
1 package com.ferbator.dao.dto;
2
3 import com.ferbator.dao.entities.Cat;
4 import com.ferbator.dao.enums.Colors;
5
6 import java.sql.Timestamp;
7 import java.util.Objects;
8
9 public class CatDto {
10     private Long id;
11     private Colors color;
12     private String name;
13     private String breed;
14     private Timestamp birthday;
15
16     public CatDto() {
17     }
18
19     public CatDto(Cat cat) {
20         this.id = cat.getId();
21         this.name = cat.getName();
22         this.color = cat.getColor();
23         this.breed = cat.getBreed();
24         this.birthday = cat.getBirthday();
25     }
26
27     public Long getId() {
28         return id;
29     }
30
31     public void setId(Long id) {
32         this.id = id;
33     }
34
35     public Colors getColor() {
36         return color;
37     }
38
39     public void setColor(Colors color) {
40         this.color = color;
41     }
42
43     public String getName() {
44         return name;
45     }
46
47     public void setName(String name) {
48         this.name = name;
49     }
```

```
50
51     public String getBreed() {
52         return breed;
53     }
54
55     public void setBreed(String breed) {
56         this.breed = breed;
57     }
58
59     public Timestamp getBirthday() {
60         return birthday;
61     }
62
63     public void setBirthday(Timestamp birthday) {
64         this.birthday = birthday;
65     }
66
67     @Override
68     public boolean equals(Object o) {
69         if (this == o) return true;
70         if (o == null || getClass() != o.getClass()) return false;
71         CatDto catDTO = (CatDto) o;
72         return Objects.equals(id, catDTO.id)
73             && color == catDTO.color
74             && Objects.equals(name, catDTO.name)
75             && Objects.equals(breed, catDTO.breed)
76             && Objects.equals(birthday, catDTO.birthday);
77     }
78
79     @Override
80     public int hashCode() {
81         return Objects.hash(id, color, name, breed, birthday);
82     }
83 }
```


Листинг 1.25: FriendshipCatDto.java

```
1 package com.ferbator.dao.dto;
2
3 import com.ferbator.dao.entities.FriendshipCat;
4
5 import java.util.Objects;
6
7 public class FriendshipCatDto {
8     private Long id;
9     private Long firstCatId;
10    private Long secondCatId;
11
12    public FriendshipCatDto() {
13    }
14
15    public FriendshipCatDto(FriendshipCat friendshipCat) {
16        this.id = friendshipCat.getId();
17        this.firstCatId = friendshipCat.getFirstCatId();
18        this.secondCatId = friendshipCat.getSecondCatId();
19    }
20
21    public Long getId() {
22        return id;
23    }
24
25    public void setId(Long id) {
26        this.id = id;
27    }
28
29    public Long getFirstCatId() {
30        return firstCatId;
31    }
32
33    public void setFirstCatId(Long firstCatId) {
34        this.firstCatId = firstCatId;
35    }
36
37    public Long getSecondCatId() {
38        return secondCatId;
39    }
40
41    public void setSecondCatId(Long secondCatId) {
42        this.secondCatId = secondCatId;
43    }
44
45    @Override
46    public boolean equals(Object o) {
47        if (this == o) return true;
48        if (o == null || getClass() != o.getClass()) return false;
49        FriendshipCatDto that = (FriendshipCatDto) o;
```

```
50         return Objects.equals(id, that.id)
51             && Objects.equals(firstCatId, that.firstCatId)
52             && Objects.equals(secondCatId, that.secondCatId);
53     }
54
55     @Override
56     public int hashCode() {
57         return Objects.hash(id, firstCatId, secondCatId);
58     }
59 }
```

Листинг 1.26: OwnerDto.java

```
1 package com.ferbator.dao.dto;
2
3 import com.ferbator.dao.entities.Owner;
4
5 import java.sql.Timestamp;
6 import java.util.Objects;
7
8 public class OwnerDto {
9     private Long id;
10    private String name;
11    private Timestamp birthday;
12
13    public OwnerDto(Owner own) {
14        this.id = own.getId();
15        this.name = own.getName();
16        this.birthday = own.getBirthday();
17    }
18
19    public OwnerDto() {
20    }
21
22    public Long getId() {
23        return id;
24    }
25
26    public void setId(Long id) {
27        this.id = id;
28    }
29
30    public String getName() {
31        return name;
32    }
33
34    public void setName(String name) {
35        this.name = name;
36    }
37
38    public Timestamp getBirthday() {
39        return birthday;
40    }
41
42    public void setBirthday(Timestamp birthday) {
43        this.birthday = birthday;
44    }
45
46    @Override
47    public boolean equals(Object o) {
48        if (this == o) return true;
49        if (o == null || getClass() != o.getClass()) return false;
```

```
50         OwnerDto ownerDTO = (OwnerDto) o;  
51         return Objects.equals(id, ownerDTO.id)  
52             && Objects.equals(name, ownerDTO.name)  
53             && Objects.equals(birthday, ownerDTO.birthday);  
54     }  
55  
56     @Override  
57     public int hashCode() {  
58         return Objects.hash(id, name, birthday);  
59     }  
60 }
```

Листинг 1.27: OwnershipCatDto.java

```
1 package com.ferbator.dao.dto;
2
3 import com.ferbator.dao.entities.OwnershipCat;
4
5 import java.util.Objects;
6
7 public class OwnershipCatDto {
8     private Long id;
9     private Long ownerId;
10    private Long catId;
11
12    public OwnershipCatDto() {
13    }
14
15    public OwnershipCatDto(OwnershipCat ownershipCat) {
16        this.ownerId = ownershipCat.getOwnerId();
17        this.catId = ownershipCat.getCatId();
18    }
19
20    public Long getId() {
21        return id;
22    }
23
24    public void setId(Long id) {
25        this.id = id;
26    }
27
28    public Long getOwnerId() {
29        return ownerId;
30    }
31
32    public void setOwnerId(Long ownerId) {
33        this.ownerId = ownerId;
34    }
35
36    public Long getCatId() {
37        return catId;
38    }
39
40    public void setCatId(Long catId) {
41        this.catId = catId;
42    }
43
44    @Override
45    public boolean equals(Object o) {
46        if (this == o) return true;
47        if (o == null || getClass() != o.getClass()) return false;
48        OwnershipCatDto that = (OwnershipCatDto) o;
49        return Objects.equals(id, that.id)
```

```
50         && Objects.equals(ownerId, that.ownerId)
51         && Objects.equals(catId, that.catId);
52     }
53
54     @Override
55     public int hashCode() {
56         return Objects.hash(id, ownerId, catId);
57     }
58 }
```

Листинг 1.28: Cat.java

```
1 package com.ferbator.dao.entities;
2
3 import com.ferbator.dao.enums.Colors;
4 import com.ferbator.dao.dto.CatDto;
5
6 import javax.persistence.*;
7 import java.sql.Timestamp;
8 import java.util.Objects;
9
10 @Entity
11 @Table(name = "cats")
12 public class Cat {
13     @Basic
14     @Column(name = "name", length = -1)
15     private String name;
16     @Basic
17     @Column(name = "birthday")
18     private Timestamp birthday;
19     @Basic
20     @Column(name = "breed", length = -1)
21     private String breed;
22     @GeneratedValue(strategy = GenerationType.IDENTITY)
23     @Id
24     @Column(name = "id", nullable = false)
25     private Long id;
26     @Basic
27     @Column(name = "color", length = -1)
28     @Enumerated(EnumType.STRING)
29     private Colors color;
30
31     public Cat(CatDto cat) {
32         this.name = cat.getName();
33         this.color = cat.getColor();
34         this.breed = cat.getBreed();
35         this.birthday = cat.getBirthday();
36     }
37
38     public Cat() {
39     }
40
41     public String getName() {
42         return name;
43     }
44
45     public void setName(String name) {
46         this.name = name;
47     }
48
49     public Timestamp getBirthday() {
```

```
50         return birthday;
51     }
52
53     public void setBirthday(Timestamp birthday) {
54         this.birthday = birthday;
55     }
56
57     public String getBreed() {
58         return breed;
59     }
60
61     public void setBreed(String breed) {
62         this.breed = breed;
63     }
64
65     public Long getId() {
66         return id;
67     }
68
69     public void setId(Long id) {
70         this.id = id;
71     }
72
73     public Colors getColor() {
74         return color;
75     }
76
77     public void setColor(Colors color) {
78         this.color = color;
79     }
80
81     @Override
82     public boolean equals(Object o) {
83         if (this == o) return true;
84         if (o == null || getClass() != o.getClass()) return false;
85         Cat cat = (Cat) o;
86         return Objects.equals(id, cat.id)
87             && Objects.equals(name, cat.name)
88             && Objects.equals(birthday, cat.birthday)
89             && Objects.equals(breed, cat.breed)
90             && Objects.equals(color, cat.color);
91     }
92
93     @Override
94     public int hashCode() {
95         return Objects.hash(name, birthday, breed, id, color);
96     }
97 }
```


Листинг 1.29: FriendshipCat.java

```
1 package com.ferbator.dao.entities;
2
3 import javax.persistence.*;
4 import java.util.Objects;
5
6 @Entity
7 @Table(name = "friendshipcats")
8 public class FriendshipCat {
9     @GeneratedValue(strategy = GenerationType.IDENTITY)
10    @Id
11    @Column(name = "id", nullable = false)
12    private Long id;
13    @Basic
14    @Column(name = "first_cat_id", nullable = false)
15    private Long firstCatId;
16    @Basic
17    @Column(name = "second_cat_id", nullable = false)
18    private Long secondCatId;
19
20    public FriendshipCat(FriendshipCat friendshipCatDTO) {
21        this.firstCatId = friendshipCatDTO.getFirstCatId();
22        this.secondCatId = friendshipCatDTO.getSecondCatId();
23    }
24
25    public FriendshipCat() {
26    }
27
28    public Long getId() {
29        return id;
30    }
31
32    public void setId(Long id) {
33        this.id = id;
34    }
35
36    public Long getFirstCatId() {
37        return firstCatId;
38    }
39
40    public void setFirstCatId(Long firstCatId) {
41        this.firstCatId = firstCatId;
42    }
43
44    public Long getSecondCatId() {
45        return secondCatId;
46    }
47
48    public void setSecondCatId(Long secondCatId) {
49        this.secondCatId = secondCatId;
50    }
51 }
```

```
50     }
51
52     @Override
53     public boolean equals(Object o) {
54         if (this == o) return true;
55         if (o == null || getClass() != o.getClass()) return false;
56         FriendshipCat that = (FriendshipCat) o;
57         return Objects.equals(id, that.id)
58             && Objects.equals(firstCatId, that.firstCatId)
59             && Objects.equals(secondCatId, that.secondCatId);
60     }
61
62     @Override
63     public int hashCode() {
64         return Objects.hash(id, firstCatId, secondCatId);
65     }
66 }
```

Листинг 1.30: Owner.java

```
1 package com.ferbator.dao.entities;
2
3 import com.ferbator.dao.dto.OwnerDto;
4
5 import javax.persistence.*;
6 import java.sql.Timestamp;
7 import java.util.Objects;
8
9 @Entity
10 @Table(name = "owners")
11 public class Owner {
12     @GeneratedValue(strategy = GenerationType.IDENTITY)
13     @Id
14     @Column(name = "id", nullable = false)
15     private Long id;
16     @Basic
17     @Column(name = "name", length = -1)
18     private String name;
19     @Basic
20     @Column(name = "birthday")
21     private Timestamp birthday;
22
23     public Owner(OwnerDto own) {
24         this.name = own.getName();
25         this.birthday = own.getBirthday();
26     }
27
28     public Owner() {
29     }
30
31     public Long getId() {
32         return id;
33     }
34
35     public void setId(Long id) {
36         this.id = id;
37     }
38
39     public String getName() {
40         return name;
41     }
42
43     public void setName(String name) {
44         this.name = name;
45     }
46
47     public Timestamp getBirthday() {
48         return birthday;
49     }
```

```
50
51 public void setBirthday(Timestamp birthday) {
52     this.birthday = birthday;
53 }
54
55 @Override
56 public boolean equals(Object o) {
57     if (this == o) return true;
58     if (o == null || getClass() != o.getClass()) return false;
59     Owner owner = (Owner) o;
60     return Objects.equals(id, owner.id)
61         && Objects.equals(name, owner.name)
62         && Objects.equals(birthday, owner.birthday);
63 }
64
65 @Override
66 public int hashCode() {
67     return Objects.hash(id, name, birthday);
68 }
69 }
```

Листинг 1.31: OwnershipCat.java

```
1 package com.ferbator.dao.entities;
2
3 import com.ferbator.dao.dto.OwnershipCatDto;
4
5 import javax.persistence.*;
6 import java.util.Objects;
7
8 @Entity
9 @Table(name = "ownershipcats")
10 public class OwnershipCat {
11     @GeneratedValue(strategy = GenerationType.IDENTITY)
12     @Id
13     @Column(name = "id", nullable = false)
14     private Long id;
15     @Basic
16     @Column(name = "owner_id", nullable = false)
17     private Long ownerId;
18     @Basic
19     @Column(name = "cat_id", nullable = false)
20     private Long catId;
21
22     public OwnershipCat(OwnershipCatDto ownershipCat) {
23         this.ownerId = ownershipCat.getOwnerId();
24         this.catId = ownershipCat.getCatId();
25     }
26
27     public OwnershipCat() {
28     }
29
30     public Long getId() {
31         return id;
32     }
33
34     public void setId(Long id) {
35         this.id = id;
36     }
37
38     public Long getOwnerId() {
39         return ownerId;
40     }
41
42     public void setOwnerId(Long ownerId) {
43         this.ownerId = ownerId;
44     }
45
46     public Long getCatId() {
47         return catId;
48     }
49 }
```

```
50     public void setCatId(Long catId) {
51         this.catId = catId;
52     }
53
54     @Override
55     public boolean equals(Object o) {
56         if (this == o) return true;
57         if (o == null || getClass() != o.getClass()) return false;
58         OwnershipCat that = (OwnershipCat) o;
59         return Objects.equals(id, that.id)
60             && Objects.equals(ownerId, that.ownerId)
61             && Objects.equals(catId, that.catId);
62     }
63
64     @Override
65     public int hashCode() {
66         return Objects.hash(id, ownerId, catId);
67     }
68 }
```

Листинг 1.32: Colors.java

```
1 package com.ferbator.dao.enums;  
2  
3 public enum Colors {  
4     Black ,  
5     Red ,  
6     Orange ,  
7     Blue  
8 }
```

Листинг 1.33: DAOException.java

```
1 package com.ferbator.dao.tools;  
2  
3 public class DAOException extends Exception{  
4     public DAOException() {  
5         super();  
6     }  
7  
8     public DAOException(String message) {  
9         super(message);  
10    }  
11  
12    public DAOException(String message, Throwable cause) {  
13        super(message, cause);  
14    }  
15 }
```


Листинг 1.34: ShelterService.java

```
1 package com.ferbator.services;
2
3 import com.ferbator.dao.daoImpl.CatRepository;
4 import com.ferbator.dao.daoImpl.FriendshipCatRepository;
5 import com.ferbator.dao.daoImpl.OwnerRepository;
6 import com.ferbator.dao.daoImpl.OwnershipCatRepository;
7 import com.ferbator.dao.dto.OwnerDto;
8 import com.ferbator.dao.entities.Cat;
9 import com.ferbator.dao.dto.CatDto;
10 import com.ferbator.dao.entities.Owner;
11 import com.ferbator.dao.enums.Colors;
12 import org.springframework.beans.factory.annotation.Autowired;
13 import org.springframework.stereotype.Service;
14
15 import javax.transaction.Transactional;
16 import java.util.List;
17
18 @Service
19 public class ShelterService {
20     CatRepository catRepository;
21     OwnerRepository ownerRepository;
22     FriendshipCatRepository friendshipCatRepository;
23     OwnershipCatRepository ownershipCatRepository;
24
25     @Autowired
26     public ShelterService(CatRepository catRepository, OwnerRepository
27         ownerRepository, FriendshipCatRepository friendshipCatRepository,
28         OwnershipCatRepository ownershipCatRepository) {
29         this.catRepository = catRepository;
30         this.ownerRepository = ownerRepository;
31         this.friendshipCatRepository = friendshipCatRepository;
32         this.ownershipCatRepository = ownershipCatRepository;
33     }
34
35     public List<CatDto> getListAllCats() {
36         return catRepository.findAll().stream().map(CatDto::new).
37             toList();
38     }
39
40     public List<OwnerDto> getListAllOwners() {
41         return ownerRepository.findAll().stream().map(OwnerDto::new).
42             toList();
43     }
44
45     public List<CatDto> getListAllOneColorCats(Colors colors) {
46         return catRepository.findAllByColor(colors).stream().map(
47             CatDto::new).toList();
48     }
49 }
```

```
45 public boolean addCat(CatDto cat) {
46     catRepository.save(new Cat(cat));
47     return true;
48 }
49
50 public boolean delCat(Long id) {
51     ownershipCatRepository.deleteAllByCatId(id);
52     friendshipCatRepository.deleteAllByFirstCatIdOrSecondCatId(id,
53     id);
54     catRepository.deleteById(id);
55     return true;
56 }
57
58 public boolean addOwner(OwnerDto owner) {
59     ownerRepository.save(new Owner(owner));
60     return true;
61 }
62
63 @Transactional
64 public boolean delOwner(Long id) {
65     ownershipCatRepository.deleteAllByOwnerId(id);
66     ownerRepository.deleteById(id);
67     return true;
68 }
```

Листинг 1.35: ShelterServiceException.java

```
1 package com.ferbator.services.tools;  
2  
3 public class ShelterServiceException extends Exception {  
4     public ShelterServiceException() {  
5         super();  
6     }  
7  
8     public ShelterServiceException(String message) {  
9         super(message);  
10    }  
11  
12    public ShelterServiceException(String message, Throwable cause) {  
13        super(message, cause);  
14    }  
15 }
```

Листинг 1.36: CatControllerTest.java

```
1 package com.ferbator.controller;
2
3 import com.ferbator.services.ShelterService;
4 import org.junit.jupiter.api.Test;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.boot.test.autoconfigure.web.servlet.
    WebMvcTest;
7 import org.springframework.boot.test.mock.mockito.MockBean;
8 import org.springframework.test.web.servlet.MockMvc;
9
10 import static org.springframework.test.web.servlet.request.
    MockMvcRequestBuilders.get;
11 import static org.springframework.test.web.servlet.result.
    MockMvcResultMatchers.status;
12
13 @WebMvcTest(CatController.class)
14 public class CatControllerTest {
15     @MockBean
16     ShelterService service;
17
18     @Autowired
19     private MockMvc mockMvc;
20
21     @Test
22     void getCatsByColor_ThrowException() throws Exception {
23         String url = "http://localhost:8080/cat/findAllOneColorCats/
    Grey";
24         mockMvc.perform(get(url)).andExpect(status().is4xxClientError
    ());
25     }
26 }
```

Листинг 1.37: ShelterServiceTest.java

```
1 package com.ferbator.services;  
2  
3 class ShelterServiceTest {  
4 }
```