|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/05 Современные интеллектуальные**

**программно-аппаратные комплексы.**

**Отчет**

**по лабораторной работе № 2**

**Название:** Создание прототипа ИС на базе MongoDB

**Дисциплина:** Распределённые базы данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-11М |  |  | И.С. Марчук |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | М.М. Фомин |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**Цель работы:** научиться создавать БД в MongoDB.

**Вариант: 8**

Предметная область для практических заданий: **Сотовая связь**

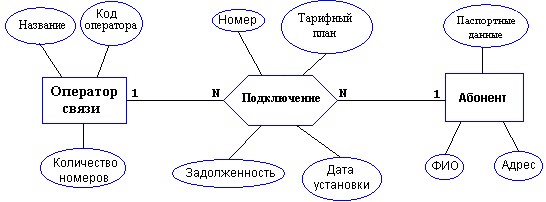


Рисунок 1 – Предметная область «Сотовая связь»

**Задание:** разработать фрагмент базы данных системы, предметную область которой кратко описана в варианте. Конкретную предметную область можно получить, взяв вариант, совпадающий с вашим порядковым номером в списке группы. Надо сказать, что приведенные схемы весьма общи, а иногда и не очень корректны, поэтому Вы можете изменить их по вашему усмотрению, оставаясь, конечно, в рамках предметной области. Для вашей предметной области необходимо **выделить минимум 2 бизнес-процесса**, которые поддаются автоматизации (это не значит, что вы должны их автоматизировать). **По** этим **бизнес-процессам** надо **выделить сущности предметной области**, которые создают костяк информационной модели предметной области. Далее нужно **прорисовать примерную схему, показывающую связи сущностей**. Не надо прорабатывать подробную схему таблиц как при работе с реляционной БД, да и можно ли говорить о «схеме» в NoSQL СУБД… Вам предлагается **создать интерфейсы к системе учета данных вашей предметной области с минимальным функционалом**:

• ввод и сохранение данных о базовых сущностях (можно просто скрипт создающий базу данных и заполняющую ее тестовыми значениями);

• поиск данных о базовых сущностях в соответствии с построенными бизнес-процессами (необходимо разработать интерфейс для поиска сущностей и скрипты выполняющие этот поиск);

• вывод и отображение данных хранящихся в БД.

Данные должны храниться в MongoDB, а инструментарий по построению интерфейса выбирается на ваше усмотрение.

Отчёт по лабораторной работе должен включать:

• схемы бизнес процессов;

• схему взаимодействия сущностей;

• тексты скриптов для создания и заполнения коллекций;

• тексты скриптов для поиска и выдачи данных по базовым сущностям;

• скриншоты интерфейсов и заполненных коллекций.

**Ход работы:**

У меня уже была установлена среда IntellijIdea, поэтому я решил разработать прототип ИС в ней на языке java. Структура созданного проекта представлена на рисунке 2.

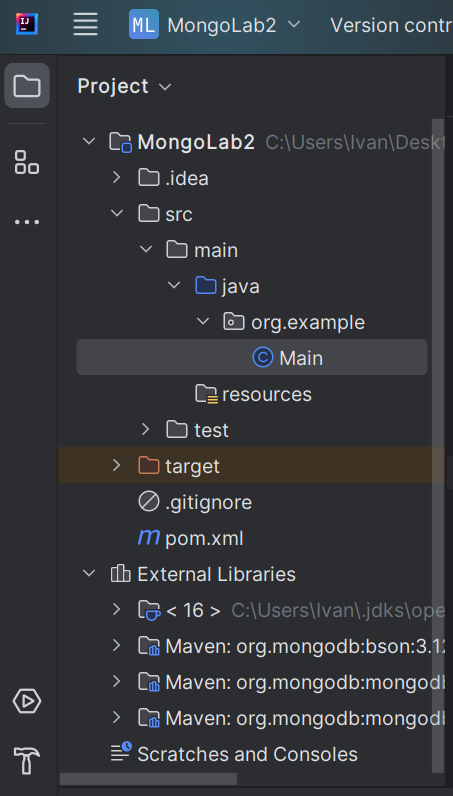


Рисунок 2 – Структура созданного проекта

Я создал проект со сборщиком maven и добавил в него библиотеку для работы с mongodb. Скриншот настройки Maven представлен на рисунке 3.

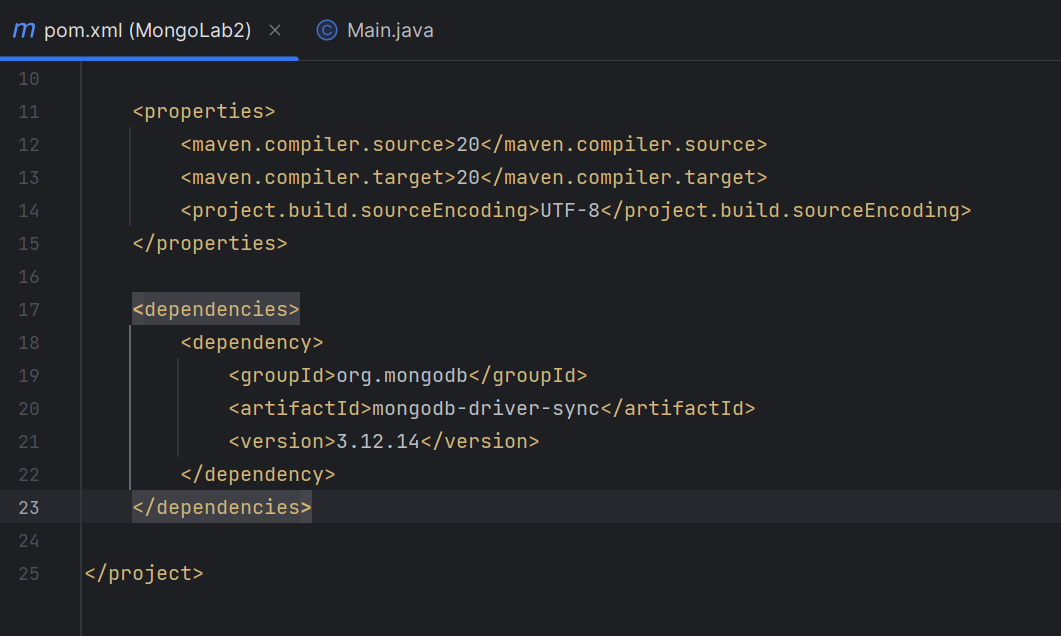


Рисунок 3 – Зависимость библиотеки MongoDB в файле Maven

Для дальнейшей работы необходимо было запустить на компьютере сервер с MongoDB. Я сделал это через консоль командой:

"c:\Program Files\MongoDB\Server\7.0\bin\mongod.exe" --dbpath="c:\mongodata\db"

Доступ к серверу происходил по адресу «mongodb://localhost:27017»

Затем я написал код по добавлению тестовой коллекции с записью в БД (рисунок 4).

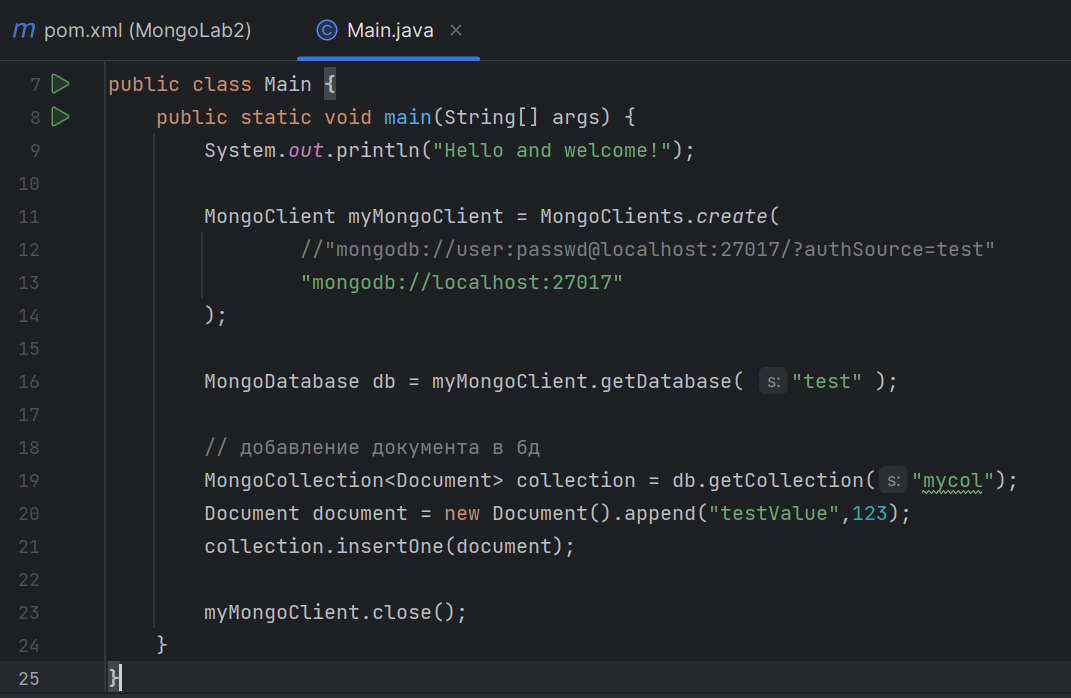


Рисунок 4 – Пример программы добавляющей документ в коллекцию

Для удобства пока я работал я активно пользовался программой MongoDB compass. В ней я и увидел созданные коллекцию «test» и документ «mycol» (Рисунок 5).

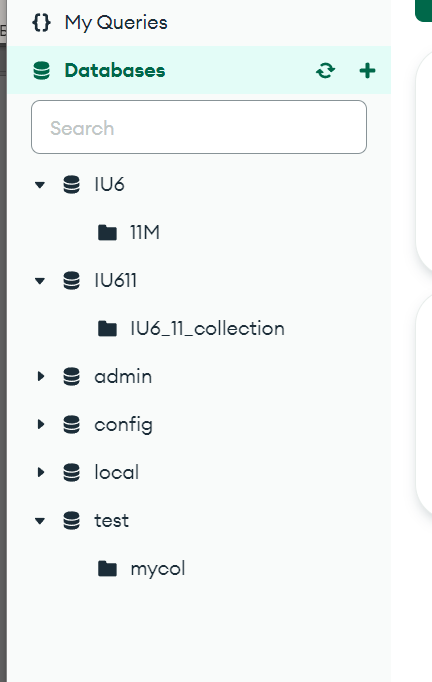


Рисунок 5 – Содержимое локальной базы данных

Дальнейшая работа по созданию ИС требовала проектирования. Я построил схемы бизнес-процессов, происходящих в предметной области по заданию. Схемы бизнес-процессов (IDEF0) представлены на рисунках 6-8.



Рисунок 6 – Схема бизнес-процесса «Регистрация абонента»



Рисунок 7 – Схема бизнес-процесса «Просмотр тарифных планов различных операторов»



Рисунок 8 – Схема бизнес-процесса «Покупка абонентом тарифа» (требованиями могут быть, например отсутствие задолженности)

Для создания не реляционной базы данных я спроектировал схему взаимодействия сущностей (UML диаграмма классов) (рисунок 9):



Рисунок 9 – схема взаимодействия сущностей

После проектирования я приступил к разработке приложения, реализующего ИС.

Были созданы 3 класса (рисунок 10).

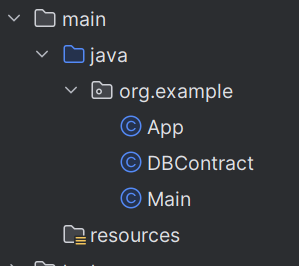


Рисунок 10 – Структура программы ИС

DBContract – класс содержащий в себе текстовые константы с названиями полей в документах и коллекций и текущую версию базы данных.

Main – класс запуска содержащий в себе статический метод запуска и обновления структуры базы данных (обновление нужно было для более удобного полного стирания а затем заполнения базы данных начальными значениями)

App – класс содержащий в себе интерфейс и основные методы работы с сущностями БД.

Теперь приведу тексты скриптов для создания и заполнения коллекций.

Механизм обновления БД (рисунок 11). Для контроля текущей версии базы данных была добавлена коллекция с настройками, содержащая номер текущей версии. При запуске программа проверяет, если документ настроек еще не создан, то база данных заполняется начальными данными (заполняет коллекцию операторов) и создает этот докуммент.

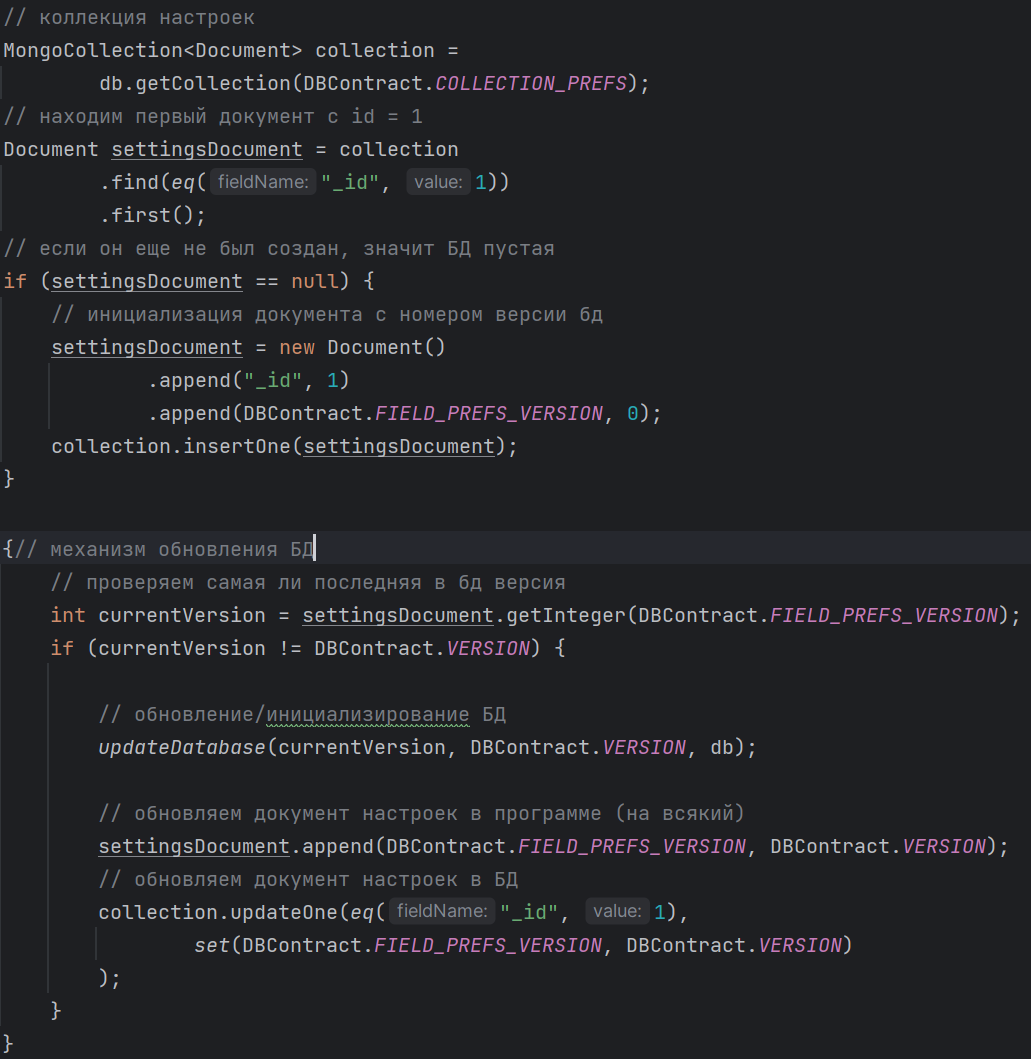


Рисунок 11 – Программа инициализации и обновления БД

Главное меню, представляет из себя циклический выбор одного из трех пунктов. (Рисунки 12-13).

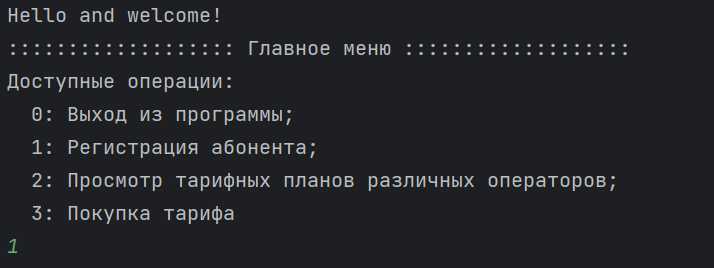


Рисунок 12 – Вид интерфейса главного меню

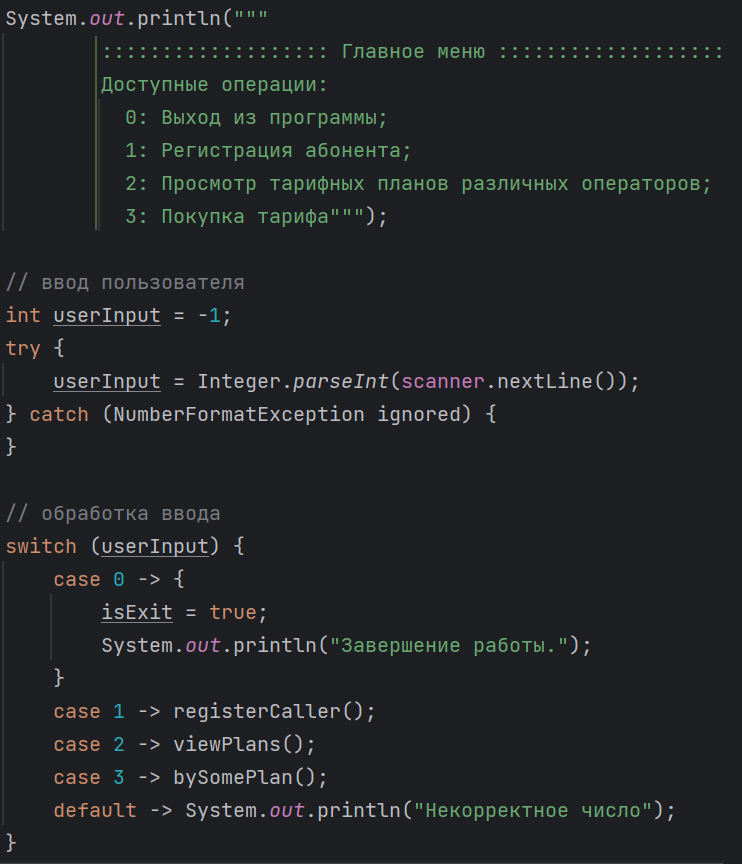


Рисунок 13 – Код выбора одного из пунктов меню

Меню регистрации абонента (рисунок 14) позволяет добавить в систему нового пользователя.

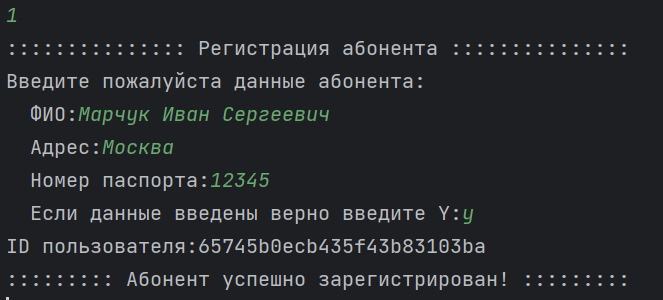


Рисунок 14 – Интерфейс регистрации абонента

Поле Connection содержащееся в документе коллекции Caller изначально пустое (что видно из рисунка 15), его можно заполнить в меню «покупка тарифа».

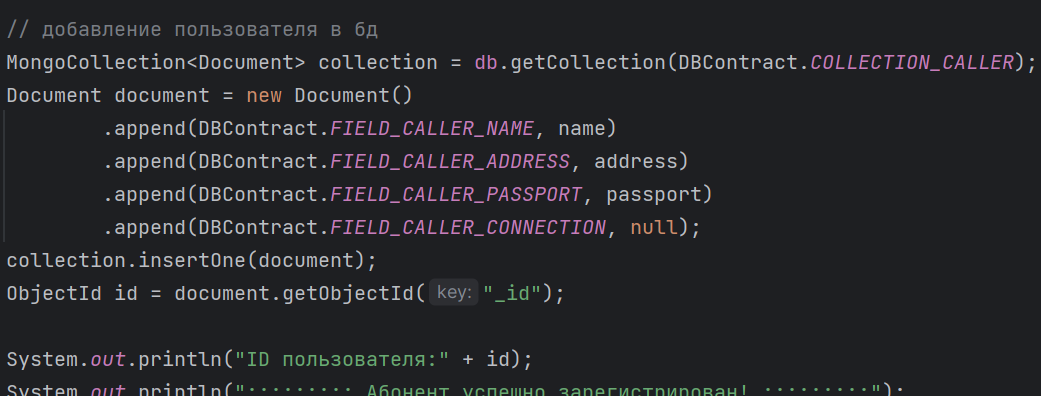


Рисунок 15 – Программа добавления документа в коллекцию Caller

В compass можно посмотреть, что документ Caller действительно добавился (рисунок 16).

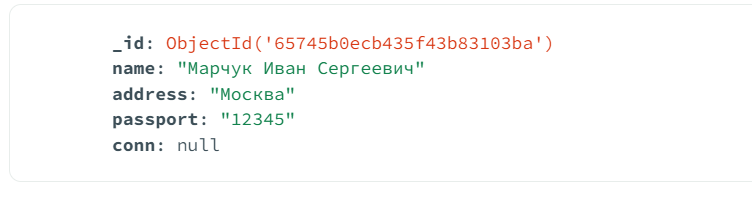


Рисунок 16 – Документ в коллекции Caller в программе Compass

В меню «просмотр тарифных планов» (рисунок 17) можно посмотреть общий список операторов и сведения о каждом из них (код на рисунке 18), а при выборе определенного оператора выведутся его тарифы (код на рисунке 19).

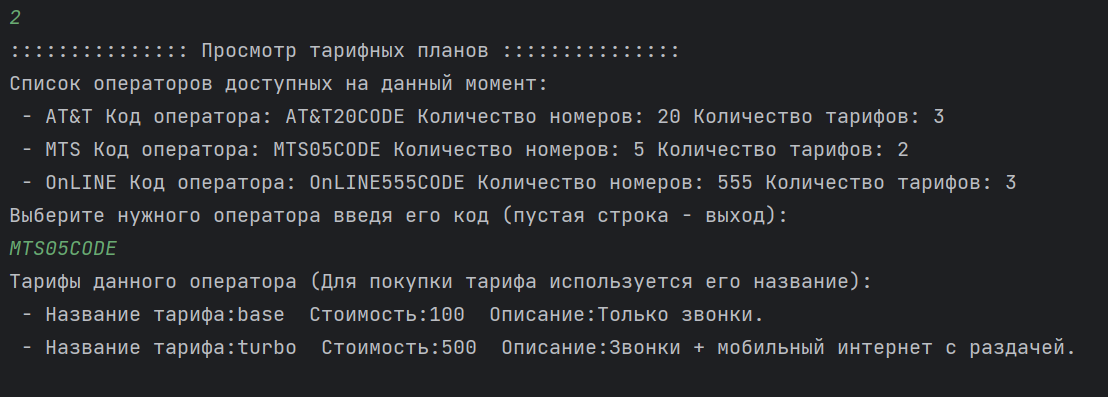


Рисунок 17 – Интерфейс просмотра операторов и тарифных планов

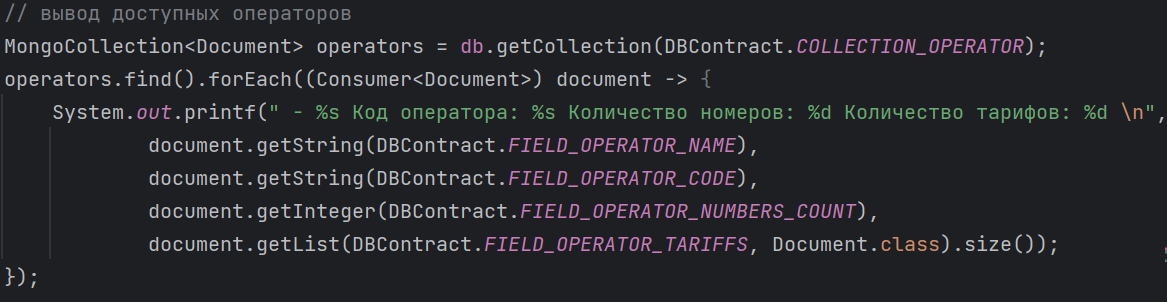


Рисунок 18 – Программа вывода списка операторов

Получение списка тарифов происходит через коллекцию Java List, то есть весь массив документов «Tariff» из документа «Operator» выгружается в оперативную память. К сожалению по другому со вложенными массивами документов в MongoDB работать нельзя, а по тому вложенные коллекции не могут быть большими, чтобы не переполнялась оперативная память клиента.

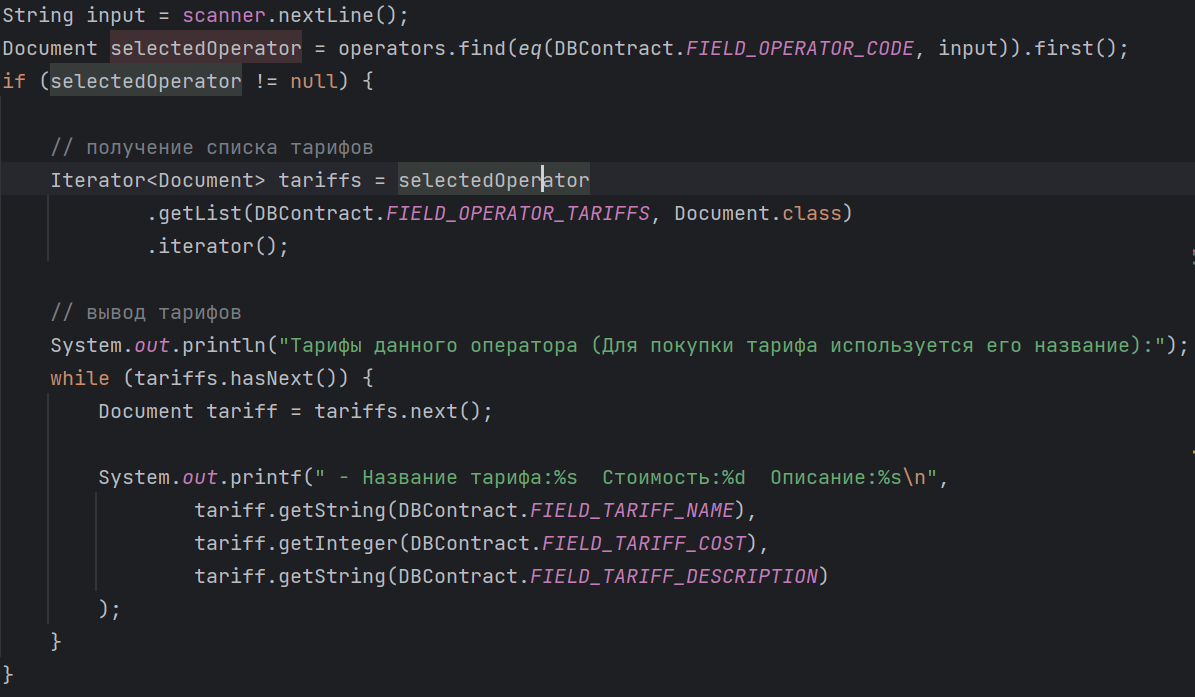


Рисунок 19 – Программа выбора оператора для отображения и вывода списка тарифов из этого оператора

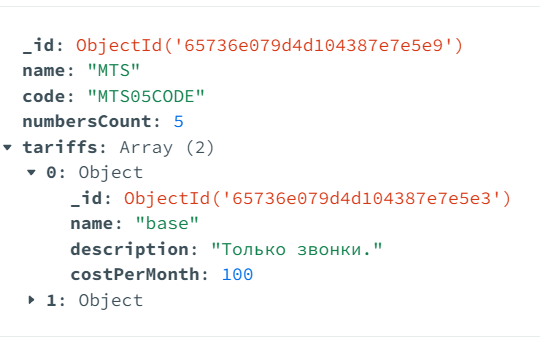


Рисунок 20 – Структура документа в коллекции Operator

Меню «покупка тарифа» позволяет найти добавленного «Caller» и посмотреть его документ с данными, где будет, в том числе, выведена информация о текущем подключении (рисунок 21).

Программа также позволяет выбрать оператора и тариф, чтобы добавить Абоненту (Caller), документ Connection (код приведен на рисунке 22).

Таким образом происходит подключение тарифа пользователю. Результат добавления представлен на рисунке 23.

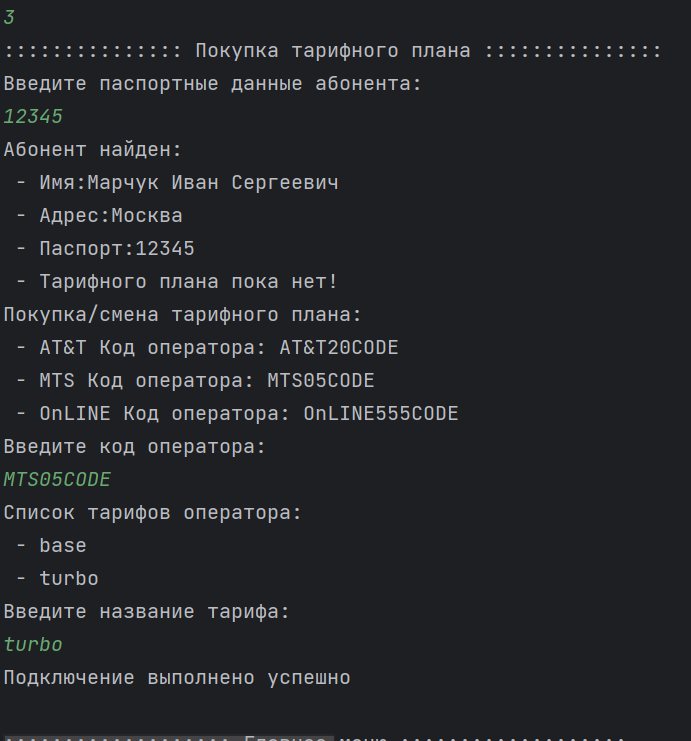


Рисунок 21 – Интерфейс меню подключения тарифного плана и просмотра данных пользователя

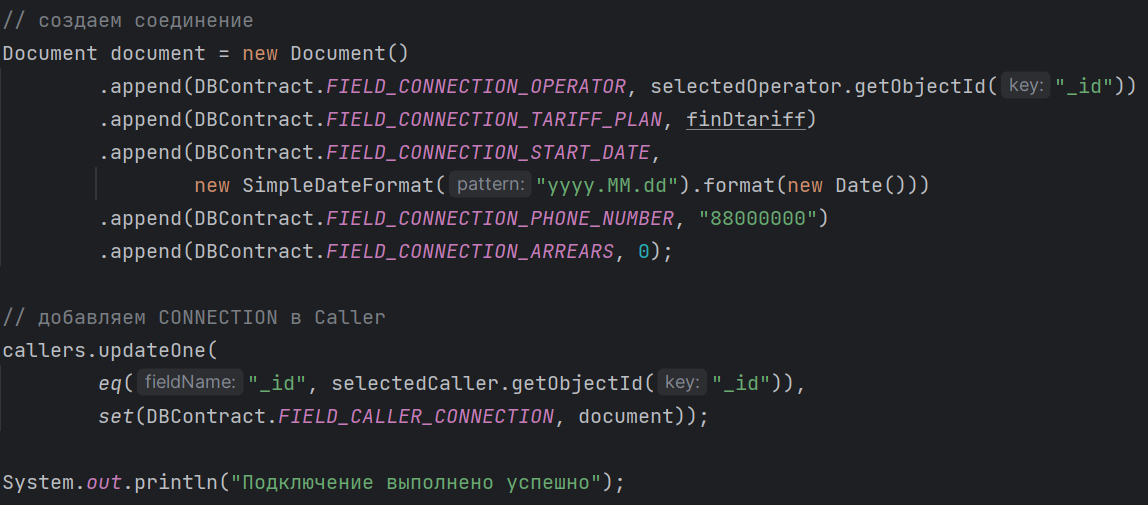


Рисунок 22 – Обновление поля connection (содержащим документ) в документе Caller

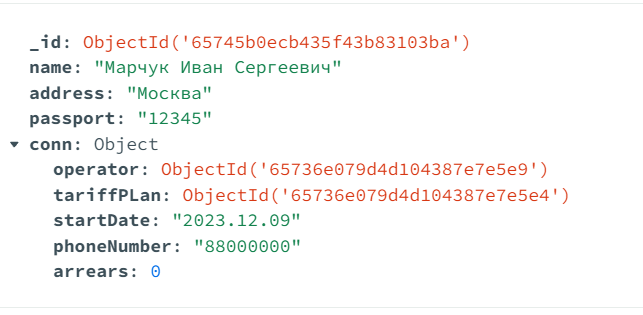


Рисунок 23 – Структура документа Caller после добавления в неё документа Connection

Полный код ИС представлен в приложении А.

**Вывод:**

Я создал информационную систему на основе СУБД Mongo DB. Получил опыт программирования на языке Java и работы с библиотекой java driver Mongo DB

**Приложение А**

**Полный код информационной системы**

Листов 10

Листинг класса DBContract:

package org.example;  
  
public class DBContract {  
  
 // константы с названиями полей и коллекций  
 public static final String *DB\_NAME* = "Сommunication3K";  
  
 public static final String *COLLECTION\_PREFS* = "Preferences";  
 /\*  
 \* P \_Id : Integer  
 \* version : String  
 \* \*/  
 public static final String *FIELD\_PREFS\_VERSION* = "version";  
 // Текущая версия БД  
 public static final int *VERSION* = 2;  
  
  
 public static final String *COLLECTION\_CALLER* = "Caller";  
 /\*  
 \* P \_id : ID  
 \* name : String  
 \* address : String  
 \* passport : String  
 \* conn : Connection  
 \* \*/  
 public static final String *FIELD\_CALLER\_NAME* = "name";  
 public static final String *FIELD\_CALLER\_ADDRESS* = "address";  
 public static final String *FIELD\_CALLER\_PASSPORT* = "passport";  
 public static final String *FIELD\_CALLER\_CONNECTION* = "conn";  
  
 public static final String *COLLECTION\_CONNECTION* = "Connection";  
 /\*  
 \* P \_id : ID  
 \* F operator : ID  
 \* F tariffPLan : ID  
 \* startDate : Date  
 \* phoneNumber : String  
 \* arrears : Integer  
 \* \*/  
 public static final String *FIELD\_CONNECTION\_OPERATOR* = "operator";  
 public static final String *FIELD\_CONNECTION\_TARIFF\_PLAN* = "tariffPLan";  
 public static final String *FIELD\_CONNECTION\_START\_DATE* = "startDate";  
 public static final String *FIELD\_CONNECTION\_PHONE\_NUMBER* = "phoneNumber";  
 public static final String *FIELD\_CONNECTION\_ARREARS* = "arrears";  
  
  
 public static final String *COLLECTION\_OPERATOR* = "Operator";  
 /\*  
 \* P \_Id : ID  
 \* name : String  
 \* code : String  
 \* numbersCount : Integer  
 \* tariffs : TariffPlan[]  
 \* \*/  
 public static final String *FIELD\_OPERATOR\_NAME* = "name";  
 public static final String *FIELD\_OPERATOR\_CODE* = "code";  
 public static final String *FIELD\_OPERATOR\_NUMBERS\_COUNT* = "numbersCount";  
 public static final String *FIELD\_OPERATOR\_TARIFFS* = "tariffs";  
  
  
 public static final String *COLLECTION\_TARIFF* = "TariffPlan";  
 /\*  
 \* P \_id : ID  
 \* name : String  
 \* description : String  
 \* costPerMonth : Integer  
 \* \*/  
 public static final String *FIELD\_TARIFF\_NAME* = "name";  
 public static final String *FIELD\_TARIFF\_DESCRIPTION* = "description";  
 public static final String *FIELD\_TARIFF\_COST* = "costPerMonth";  
  
}

Листинг класса Main:

package org.example;

import com.mongodb.client.MongoClient;

import com.mongodb.client.MongoClients;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.MongoDatabase;

import org.bson.Document;

import org.bson.types.ObjectId;

import java.util.ArrayList;

import java.util.List;

import static com.mongodb.client.model.Filters.eq;

import static com.mongodb.client.model.Updates.set;

public class Main {

public static void main(String[] args) {

// подключаемся к СУБД

System.out.println("Инициализация базы.");

try (MongoClient myMongoClient = MongoClients.create(

"mongodb://localhost:27017"//"mongodb://user:passwd@localhost:27017/?authSource=test"

)) {

// ждем когда MONGODB подключится

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

e.printStackTrace();

}

MongoDatabase db = myMongoClient.getDatabase(DBContract.DB\_NAME);

{// проверка бд на инициализированность

// коллекция настроек

MongoCollection<Document> collection =

db.getCollection(DBContract.COLLECTION\_PREFS);

// находим первый документ c id = 1

Document settingsDocument = collection

.find(eq("\_id", 1))

.first();

// если он еще не был создан, значит БД пустая

if (settingsDocument == null) {

// инициализация документа с номером версии бд

settingsDocument = new Document()

.append("\_id", 1)

.append(DBContract.FIELD\_PREFS\_VERSION, 0);

collection.insertOne(settingsDocument);

}

{// механизм обновления БД

// проверяем самая ли последняя в бд версия

int currentVersion = settingsDocument.getInteger(DBContract.FIELD\_PREFS\_VERSION);

if (currentVersion != DBContract.VERSION) {

// обновление/инициализирование БД

updateDatabase(currentVersion, DBContract.VERSION, db);

// обновляем документ настроек в программе (на всякий)

settingsDocument.append(DBContract.FIELD\_PREFS\_VERSION, DBContract.VERSION);

// обновляем документ настроек в БД

collection.updateOne(eq("\_id", 1),

set(DBContract.FIELD\_PREFS\_VERSION, DBContract.VERSION)

);

}

}

}

System.out.println("Hello and welcome!");

// инициализация интерфейса пользователя

App app = new App(db);

// запуск главного меню

app.startManMenu();

}

}

private static void updateDatabase(int oldVersion, int newVersion, MongoDatabase db) {

System.out.println("Update database from:" + oldVersion + "v to:" + newVersion + "v");

// пустая БД

if (oldVersion == 0) {

// заполняем коллекции операторов и тарифов

MongoCollection<Document> operators =

db.getCollection(DBContract.COLLECTION\_OPERATOR);

// чистка всего

operators.drop();

// создание списка операторов

List<Document> operatorsDocs = new ArrayList<>();

{

// список тарифов конкретного оператора

List<Document> tariffDocsATNT = new ArrayList<>();

tariffDocsATNT.add(new Document()

.append("\_id", new ObjectId())

.append(DBContract.FIELD\_TARIFF\_NAME, "smallTariff")

.append(DBContract.FIELD\_TARIFF\_DESCRIPTION, "Минимум функций, только необходимое.")

.append(DBContract.FIELD\_TARIFF\_COST, 100));

tariffDocsATNT.add(new Document()

.append("\_id", new ObjectId())

.append(DBContract.FIELD\_TARIFF\_NAME, "mediumTariff")

.append(DBContract.FIELD\_TARIFF\_DESCRIPTION, "Как хлебушек, но с маслом.")

.append(DBContract.FIELD\_TARIFF\_COST, 200));

tariffDocsATNT.add(new Document()

.append("\_id", new ObjectId())

.append(DBContract.FIELD\_TARIFF\_NAME, "SuperTariff")

.append(DBContract.FIELD\_TARIFF\_DESCRIPTION, "Любой каприз за ваши деньги.")

.append(DBContract.FIELD\_TARIFF\_COST, 500));

// создание оператора

operatorsDocs.add(new Document()

.append(DBContract.FIELD\_OPERATOR\_NAME, "AT&T")

.append(DBContract.FIELD\_OPERATOR\_CODE, "AT&T20CODE")

.append(DBContract.FIELD\_OPERATOR\_NUMBERS\_COUNT, 20)

.append(DBContract.FIELD\_OPERATOR\_TARIFFS, tariffDocsATNT)

);

}

{

// список тарифов конкретного оператора

List<Document> tariffDocsMTS = new ArrayList<>();

tariffDocsMTS.add(new Document()

.append("\_id", new ObjectId())

.append(DBContract.FIELD\_TARIFF\_NAME, "base")

.append(DBContract.FIELD\_TARIFF\_DESCRIPTION, "Только звонки.")

.append(DBContract.FIELD\_TARIFF\_COST, 100));

tariffDocsMTS.add(new Document()

.append("\_id", new ObjectId())

.append(DBContract.FIELD\_TARIFF\_NAME, "turbo")

.append(DBContract.FIELD\_TARIFF\_DESCRIPTION, "Звонки + мобильный интернет с раздачей.")

.append(DBContract.FIELD\_TARIFF\_COST, 500));

// создание оператора

operatorsDocs.add(new Document()

.append(DBContract.FIELD\_OPERATOR\_NAME, "MTS")

.append(DBContract.FIELD\_OPERATOR\_CODE, "MTS05CODE")

.append(DBContract.FIELD\_OPERATOR\_NUMBERS\_COUNT, 5)

.append(DBContract.FIELD\_OPERATOR\_TARIFFS, tariffDocsMTS)

);

}

{

// список тарифов конкретного оператора

List<Document> tariffDocsOnLine = new ArrayList<>();

tariffDocsOnLine.add(new Document()

.append("\_id", new ObjectId())

.append(DBContract.FIELD\_TARIFF\_NAME, "asket")

.append(DBContract.FIELD\_TARIFF\_DESCRIPTION, "Минимум функций, только необходимое.")

.append(DBContract.FIELD\_TARIFF\_COST, 250));

tariffDocsOnLine.add(new Document()

.append("\_id", new ObjectId())

.append(DBContract.FIELD\_TARIFF\_NAME, "middle man")

.append(DBContract.FIELD\_TARIFF\_DESCRIPTION, "Средний такой тариф")

.append(DBContract.FIELD\_TARIFF\_COST, 300));

tariffDocsOnLine.add(new Document()

.append("\_id", new ObjectId())

.append(DBContract.FIELD\_TARIFF\_NAME, "gigachad")

.append(DBContract.FIELD\_TARIFF\_DESCRIPTION, "Вы не найдете тарифа лучше!")

.append(DBContract.FIELD\_TARIFF\_COST, 400));

// создание оператора

operatorsDocs.add(new Document()

.append(DBContract.FIELD\_OPERATOR\_NAME, "OnLINE")

.append(DBContract.FIELD\_OPERATOR\_CODE, "OnLINE555CODE")

.append(DBContract.FIELD\_OPERATOR\_NUMBERS\_COUNT, 555)

.append(DBContract.FIELD\_OPERATOR\_TARIFFS, tariffDocsOnLine)

);

}

operators.insertMany(operatorsDocs);

}

}

}

Листинг класса App:

package org.example;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.MongoDatabase;

import org.bson.Document;

import org.bson.types.ObjectId;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.Iterator;

import java.util.List;

import java.util.Scanner;

import java.util.function.Consumer;

import static com.mongodb.client.model.Filters.eq;

import static com.mongodb.client.model.Updates.set;

public class App {

private final MongoDatabase db;

private final Scanner scanner;

// инициализация интерфейса

public App(MongoDatabase db) {

this.db = db;

scanner = new Scanner(System.in);

}

// запуск работы интерфейса

void startManMenu() {

// главный интерфейсный цикл

boolean isExit = false;

while (!isExit) {

System.out.println("""

::::::::::::::::::: Главное меню :::::::::::::::::::

Доступные операции:

0: Выход из программы;

1: Регистрация абонента;

2: Просмотр тарифных планов различных операторов;

3: Покупка тарифа""");

// ввод пользователя

int userInput = -1;

try {

userInput = Integer.parseInt(scanner.nextLine());

} catch (NumberFormatException ignored) {

}

// обработка ввода

switch (userInput) {

case 0 -> {

isExit = true;

System.out.println("Завершение работы.");

}

case 1 -> registerCaller();

case 2 -> viewPlans();

case 3 -> bySomePlan();

default -> System.out.println("Некорректное число");

}

scanner.nextLine();

}

}

// Регистрация абонента

private void registerCaller() {

System.out.println("::::::::::::::: Регистрация абонента :::::::::::::::");

System.out.println("Введите пожалуйста данные абонента:");

System.out.print(" ФИО:");

String name = scanner.nextLine();

System.out.print(" Адрес:");

String address = scanner.nextLine();

System.out.print(" Номер паспорта:");

String passport = scanner.nextLine();

System.out.print(" Если данные введены верно введите Y:");

String confirm = scanner.nextLine();

if (confirm.charAt(0) == 'Y' || confirm.charAt(0) == 'y' ||

confirm.charAt(0) == 'Н' || confirm.charAt(0) == 'н') {

// добавление пользователя в бд

MongoCollection<Document> collection = db.getCollection(DBContract.COLLECTION\_CALLER);

Document document = new Document()

.append(DBContract.FIELD\_CALLER\_NAME, name)

.append(DBContract.FIELD\_CALLER\_ADDRESS, address)

.append(DBContract.FIELD\_CALLER\_PASSPORT, passport)

.append(DBContract.FIELD\_CALLER\_CONNECTION, null);

collection.insertOne(document);

ObjectId id = document.getObjectId("\_id");

System.out.println("ID пользователя:" + id);

System.out.println("::::::::: Абонент успешно зарегистрирован! :::::::::");

} else {

System.out.println(":::::::::::::::::: Отмена операции ::::::::::::::::::");

}

}

// Просмотр тарифных планов различных операторов

private void viewPlans() {

System.out.println("::::::::::::::: Просмотр тарифных планов :::::::::::::::");

System.out.println("Список операторов доступных на данный момент:");

// вывод доступных операторов

MongoCollection<Document> operators = db.getCollection(DBContract.COLLECTION\_OPERATOR);

operators.find().forEach((Consumer<Document>) document -> {

System.out.printf(" - %s Код оператора: %s Количество номеров: %d Количество тарифов: %d \n",

document.getString(DBContract.FIELD\_OPERATOR\_NAME),

document.getString(DBContract.FIELD\_OPERATOR\_CODE),

document.getInteger(DBContract.FIELD\_OPERATOR\_NUMBERS\_COUNT),

document.getList(DBContract.FIELD\_OPERATOR\_TARIFFS, Document.class).size());

});

// выбор оператора

System.out.println("Выберите нужного оператора введя его код (пустая строка - выход):");

String input = scanner.nextLine();

Document selectedOperator = operators.find(eq(DBContract.FIELD\_OPERATOR\_CODE, input)).first();

if (selectedOperator != null) {

// получение списка тарифов

Iterator<Document> tariffs = selectedOperator

.getList(DBContract.FIELD\_OPERATOR\_TARIFFS, Document.class)

.iterator();

// вывод тарифов

System.out.println("Тарифы данного оператора (Для покупки тарифа используется его название):");

while (tariffs.hasNext()) {

Document tariff = tariffs.next();

System.out.printf(" - Название тарифа:%s Стоимость:%d Описание:%s\n",

tariff.getString(DBContract.FIELD\_TARIFF\_NAME),

tariff.getInteger(DBContract.FIELD\_TARIFF\_COST),

tariff.getString(DBContract.FIELD\_TARIFF\_DESCRIPTION)

);

}

}

}

// Покупка абонентом тарифа

private void bySomePlan() {

System.out.println("::::::::::::::: Покупка тарифного плана :::::::::::::::");

MongoCollection<Document> callers = db.getCollection(DBContract.COLLECTION\_CALLER);

// Здесь можно вывести список пользователей... потом

System.out.println("Введите паспортные данные абонента:");

Document selectedCaller = callers.find(eq(DBContract.FIELD\_CALLER\_PASSPORT, scanner.nextLine())).first();

if (selectedCaller != null) {

System.out.println("Абонент найден:");

System.out.println(" - Имя:" + selectedCaller.getString(DBContract.FIELD\_CALLER\_NAME));

System.out.println(" - Адрес:" + selectedCaller.getString(DBContract.FIELD\_CALLER\_ADDRESS));

System.out.println(" - Паспорт:" + selectedCaller.getString(DBContract.FIELD\_CALLER\_PASSPORT));

// текущее соединение

Document connection = (Document) selectedCaller.get(DBContract.FIELD\_CALLER\_CONNECTION);

if (connection == null) {

System.out.println(" - Тарифного плана пока нет!");

} else {

System.out.println(" - Подключен тарифный план:");

// загружаем данные об операторе и тарифе

MongoCollection<Document> operators = db.getCollection(DBContract.COLLECTION\_OPERATOR);

Document operator =

operators.find(eq("\_id", connection.getObjectId(DBContract.FIELD\_CONNECTION\_OPERATOR))).first();

if(operator != null) {

String tariffName = null;

Iterator<Document> tariffsIterator =

operator.getList(DBContract.FIELD\_OPERATOR\_TARIFFS, Document.class).iterator();

while (tariffsIterator.hasNext() && tariffName == null) {

Document document = tariffsIterator.next();

if (document.getObjectId("\_id").equals(connection.getObjectId(DBContract.FIELD\_CONNECTION\_TARIFF\_PLAN))) {

tariffName = document.getString(DBContract.FIELD\_TARIFF\_NAME);

}

}

System.out.println(" + Оператор:" +

((operator != null) ?

(operator.getString(DBContract.FIELD\_OPERATOR\_NAME)) :

("Отсутствует")));

System.out.println(" + Тарифный план:" + tariffName);

}else{

System.out.println(" + Оператор:Отсутствует \n + Тарифный план:Отсутствует");

}

System.out.println(" + Дата подключения:" + connection.getString(DBContract.FIELD\_CONNECTION\_START\_DATE));

System.out.println(" + Телефон:" + connection.getString(DBContract.FIELD\_CONNECTION\_PHONE\_NUMBER));

System.out.println(" + Задолженность:" + connection.getInteger(DBContract.FIELD\_CONNECTION\_ARREARS));

}

System.out.println("Покупка/смена тарифного плана: ");

// вывод доступных операторов

MongoCollection<Document> operators = db.getCollection(DBContract.COLLECTION\_OPERATOR);

operators.find().forEach((Consumer<Document>) document ->

System.out.printf(" - %s Код оператора: %s \n",

document.getString(DBContract.FIELD\_OPERATOR\_NAME),

document.getString(DBContract.FIELD\_OPERATOR\_CODE)));

// выбор оператора

System.out.println("Введите код оператора: ");

Document selectedOperator = operators.find(eq(DBContract.FIELD\_OPERATOR\_CODE, scanner.nextLine())).first();

if (selectedOperator == null) {

System.out.println("Оператор не найден.");

} else {

// Правда код с List не будет работать на большом количестве данных

// (он загружает сразу всю коллекцию в отличие от курсора(FindIterable))

// по этому тарифов у оператора не может быть больше например 100

// получение списка тарифов

List<Document> tariffs = selectedOperator

.getList(DBContract.FIELD\_OPERATOR\_TARIFFS, Document.class);

{// вывод тарифов

Iterator<Document> tariffsI = tariffs.iterator();

System.out.println("Список тарифов оператора:");

while (tariffsI.hasNext()) {

Document tariff = tariffsI.next();

System.out.println(" - " + tariff.getString(DBContract.FIELD\_TARIFF\_NAME));

}

}

System.out.println("Введите название тарифа: ");

{// выбор тарифа

String input = scanner.nextLine();

ObjectId finDtariff = null;

// проходимся по коллекции тарифов

Iterator<Document> tariffsI = tariffs.iterator();

while (tariffsI.hasNext() && finDtariff == null) {

// если у этого тарифа нужное имя

Document tariffUnit = tariffsI.next();

if (input.equals(tariffUnit.getString(DBContract.FIELD\_TARIFF\_NAME))) {

finDtariff = tariffUnit.getObjectId("\_id");

}

}

// Если такой тариф есть, добавляем CONNECTION

if (finDtariff != null) {

// создаем соединение

Document document = new Document()

.append(DBContract.FIELD\_CONNECTION\_OPERATOR, selectedOperator.getObjectId("\_id"))

.append(DBContract.FIELD\_CONNECTION\_TARIFF\_PLAN, finDtariff)

.append(DBContract.FIELD\_CONNECTION\_START\_DATE,

new SimpleDateFormat("yyyy.MM.dd").format(new Date()))

.append(DBContract.FIELD\_CONNECTION\_PHONE\_NUMBER, "88000000")

.append(DBContract.FIELD\_CONNECTION\_ARREARS, 0);

// добавляем CONNECTION в Caller

callers.updateOne(

eq("\_id", selectedCaller.getObjectId("\_id")),

set(DBContract.FIELD\_CALLER\_CONNECTION, document));

System.out.println("Подключение выполнено успешно");

}

}

}

}

}

}