Лабораторная работа № 5: Работа с файлами и многопоточность

1. Создайте файл `FilesExample.java`, демонстрирующий работу с файлами:

```
import java.io.*;
import java.nio.charset.Charset;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import static java.nio.file.StandardOpenOption.APPEND;

public class FilesExample {

    private static void inputStreamCopy(String path, String
fileInName, String fileOutName){
        FileInputStream in;
        FileOutputStream out;
        // побайтовые чтение и запись
        try {
            in = new FileInputStream(path + fileInName);
            out = new FileOutputStream(path + fileOutName);
            int c;
            while ((c = in.read()) != -1) {
                out.write(c);
            }
            System.out.println("File copied with
FileInputStream");
        }
        catch(IOException e){
            System.out.println("Exception caught:");
            // вывод полной информации об исключении
            e.printStackTrace();
        }
    }

    private static void bufferedReaderCopy(String path, String
fileInName, String fileOutName) {
        // построчные чтение и запись
        try (BufferedReader reader =
Files.newBufferedReader(Paths.get(path + fileInName),
                                    Charset.defaultCharset()))
{
            String line = null;
            try (BufferedWriter writer =
Files.newBufferedWriter(Paths.get(path + fileOutName),

Charset.defaultCharset(), APPEND)) {
                while ((line = reader.readLine()) != null) {
                    writer.write(line, 0, line.length());
                    writer.write('\n');
                }
```

```
                System.out.println("File       copied       with
BufferedReader");
            } catch (IOException e) {
                System.out.println("Exception caught (write):");
                System.err.format("IOException: %s%n", e);
            }
        } catch (IOException e) {
            System.out.println("Exception caught (read):");
            System.err.format("IOException: %s%n", e);
        }
    }

    public static void main(String[] args){
        String path = System.getProperty("user.dir") + "/src/";
        BufferedReader inputBuf;
        List<String> students = new ArrayList<>();
        try {
            inputBuf       =       new       BufferedReader(new
InputStreamReader(System.in)); // для ввода с клавиатуры
            System.out.println("Enter   student   names   ('0'   -
exit): ");
            while (true) {
                // ввод с клавиатуры
                String input = inputBuf.readLine();
                if ("0".equals(input)) break;
                else students.add(input);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.println(students);
        inputStreamCopy(path, "fileIn.txt","fileOut_01.txt");
        bufferedReaderCopy(path, "fileIn.txt","fileOut_02.txt");
    }
}
```

Также создайте пустые файлы `fileOut_01.txt` и `fileOut_02.txt` и файл `fileIn.txt` в папке `src` со следующим содержимым:

```
Information - knowledge that you get about someone or something: facts or
details about a subject.
Information - knowledge obtained from investigation, study, or instruction.
```

Запустите программу и удостоверьтесь в ее работоспособности.
Ознакомьтесь с выведенной информацией. Пример результата выполнения программы приведен ниже:

```
Enter student names ('0' - exit):
Student1
Student2
0
[Student1, Student2]
File copied with FileInputStream
File copied with BufferedReader
```

Содержимое файла `fileOut_01.txt`:

```
Information - knowledge that you get about someone or something: facts or details about a subject.
Information - knowledge obtained from investigation, study, or instruction.
```

Содержимое файла `fileOut_02.txt`:

```
Information - knowledge that you get about someone or something: facts or details about a subject.
Information - knowledge obtained from investigation, study, or instruction.
```

Работа с файлами

В Java существует несколько различных способов работы с файлами, основными из которых является использование `FileInputStream` и `BufferedReader`. Рассмотрим различные опции открытия файлов, хранящиеся в классе `java.nio.file.StandardOpenOption`, которые указываются при создании объектов работы с файлом:

- `WRITE` – запись в файл;
- `APPEND` – дополнение файла;
- `TRUNCATE_EXISTING` –удаление предыдущего содержимого;
- `CREATE_NEW` – создание нового файла, если файл с таким именем существует, то выдается исключение;
- `CREATE` – открытие файла, если он существует, или создание нового файла;
- `DELETE_ON_CLOSE` – удаление файла после закрытия (например, для временных файлов).

`FileInputStream` и `FileOutputStream`

Данные классы предназначены для побайтового считывания и записи в файл. `FileInputStream` служит для считывания данных из файла. Доступ к файлу устанавливается следующим образом:
`FileInputStream(File file)` – в качестве аргумента передается объект типа `File`.
`FileInputStream(String name)` – в качестве аргумента передается путь к файлу.

Основные функции:

- `available()` – проверка количества байт, доступных для считывания;
- `read()` – считывание байта из потока файла;
- `close()` – закрытие потока файла.

Для побайтового считывания данных из файла может быть использован объект класса `FileOutputStream`:
`FileOutputStream(. . .)` – в качестве аргумента передается объект типа `File` или путь к файлу.
`FileOutputStream(. . . , boolean append)` – первый аргумент – объект типа `File` или путь к файлу, второй аргумент – флаг открытия файла на дополнение.

Основные функции:

- `write()` – запись байта в файл;
- `close()` – закрытие потока файла.

BufferedReader

BufferedReader используется совместно с InputStreamReader или BufferedReader, при этом отдельно для входного и выходного файлового потока создается свой объект класса BufferedReader. Для чтения данных из файла может быть использована следующая запись:
BufferedReader reader = Files.newBufferedReader(Paths.get(name), Charset.defaultCharset()), где:

- Files – класс, содержащий static-методы для работы с файлами и директориями.
- newBufferedReader() – метод класса Files для получения объекта типа BufferedReader для считывания данных. Аргументы - объекты типов Path (путь к файлу) и Charset (кодировка файла).
- Paths.get(name) – позволяет получить объект типа Path.
- Charset.defaultCharset() – получение стандартной кодировки в виде объекта типа Charset

Для считывания данных доступны следующие функции объекта типа BufferedReader:

- read() – считывание символа из файла.
- readLine() – считывание строки из файла.
- skip() – пропуск некоторого количества символов, принимает на вход количество символов для пропуска.
- close() – закрытие потока.

Для записи в файл используется следующий формат:
BufferedWriter writer = Files.newBufferedWriter(Paths.get(name), Charset.defaultCharset(), APPEND), где:

- newBufferedWriter () – метод класса Files для получения объекта типа BufferedWriter для записи данных. Аргументы - объекты типов Path (путь к файлу), Charset (кодировка файла) и опция открытия файла.

Основные функции BufferedWriter:

- write(…) – в зависимости от входных аргументов позволяет записать отдельный символ, массив символов или подстроку.
- flush() – запись всех данных, находившихся в буфере.
- close() – закрытие потока.

Обработка исключений (try-catch-finally)

Обработка исключений в Java осуществляется блоком try-catch-finally:

```
try{
    . . .
}
catch (<тип_исключения> <имя_переменной>) {
    . . .
}
. . .
catch (<тип_исключения> <имя_переменной>) {
```

```
    . . .
}
finally {
    . . .
}
```

В блоке `try` указываются операторы, при выполнении которых могут возникнуть исключения. Исключения указываются в блоке `catch` совместно с их типом. В Java Иерархия исключений представляется следующим образом:

- `Throwable`
  - `Error`
    - `. . .`
  - `Exception`
    - `RuntimeException`
    - `. . .`

Примеры исключений:

- `RuntimeException` – ошибки выполнения программы;
- `NullPointerException` – ошибка (не-)указания значения `null`;
- `ArrayIndexOutOfBoundsException` – выход за границы массива;
- `IOException` – ошибка доступа к файлу.

В блоке `finally` определяются операторы, которые должны быть выполнены в любом случае после выполнения блоков `try` и `catch`.

Если известно, что метод может выдать некоторое исключение, можно прописать операторы без `try-catch`, указав, что метод выбрасывает исключение с помощью ключевого слова `throws`:

```
public static void main(String[] args) throws Throwable { . . .
}
```

Для самостоятельного вызова исключения используется ключевое слово `throw`:

```
public static void main(String[] args) {
    throw new Error();
}
```

2. Создайте файл `osi.txt` со следующим содержимым (данные разделены точкой с запятой):

```
Level 7; application; example: HTTP
Level 6; presentation; example: JPEG
Level 5; session; example: RPC
Level 4; transport; example: TCP
Level 3; network; example: IPv6
Level 2; data link; example: IEEE 802.2
Level 1; physical; example: USB
```

3. Для созданного на предыдущем шаге файла `osi.txt` двумя способами (с использованием `FileInputStream-FileOutputStream` и `BufferedReader`)

реализуйте считывание данных из файла `osi.txt` и запись обратно отсортированных данных (с 1 по 7 уровень модели OSI) в файл `osiReverse.txt`.

4. Создайте файл `Calculation.java`, демонстрирующий работу с реализацией интерфейса `Runnable` для работы с потоками:

```
public class Calculation implements Runnable {
    private int id;
    private double time;
    public Calculation(int id, double time) {
        this.id = id;
        this.time = time;
    }
    @Override
    public void run() {
        double passedTime = 0;
        while (true) {
            if (ThreadExample.currentCalculation == 0)
ThreadExample.currentCalculation = this.id;
            if (ThreadExample.currentCalculation == this.id) {
                passedTime += 0.5;
                System.out.print(".");
                try {
                    Thread.sleep(500);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
            if (passedTime >= this.time) {
                ThreadExample.currentCalculation = 0;
                System.out.println("\nCalculation #" + this.id +
" finished!");
                break;
            }
        }
    }
}
```

5. Создайте файл `ThreadExample.java`, демонстрирующий работу с потоками:

```
public class ThreadExample {
    volatile static int currentCalculation;
    public static void main(String[] args){
        Thread calc1 = new Thread(new Calculation(1,3.5));
        calc1.start();
        Thread calc2 = new Thread(new Calculation(2,5.5));
        calc2.start();
    }
}
```

Запустите программу и удостоверьтесь в ее работоспособности.
Ознакомьтесь с выведенной информацией. Пример результата выполнения программы приведен ниже:

```
........
Calculation #1 finished!
..........
Calculation #2 finished!
```

Многопоточность

Многопоточность часто используется для ускорения процессов сложных вычислений, реализации взаимодействия пользователя с интерфейсом приложений и т.д. В Java многопоточность реализуется с использованием класса `Thread` и классов, реализующих интерфейс `Runnable`, предоставляющий метод работы с внешне запускаемыми фрагментами кода. Для классов, реализующих интерфейс `Runnable` обязательно должен быть определен метод `run()`, отвечающий за запуск потока.

При создании двух и более потоков может быть реализовано параллельное выполнение некоторых необходимых вычислений. Для этого два потока должны быть запущены один за другим. Класс `Thread` отвечает непосредственно за сами потоки. Таким образом, класс, отвечающий за выполнение определенных вычислений, должен либо наследовать класс `Thread`, либо реализовывать интерфейс `Runnable`.

Создание потоков осуществляется следующим образом:
`Thread runner = new Thread(new Calculation(1,3.5));`
Если некоторый класс `Calculation` реализует `Runnable`.

Или же:
`Thread runner = new Calculation(1,3.5);`
Если класс `Calculation` наследует `Thread`.

Запуск потока осуществляется с помощью метода `start()`. static-метод `Thread.sleep(..)` позволяет поставить выполнение программы на паузы на число миллисекунд, переданное в качестве аргумента метода.

Для контроля доступа нескольких потоков к одному и тому же ресурсу необходимо использовать модификатор `volatile`, разрешающий изменение ресурса нескольким потокам одновременно. Для запрета выполнения определенного метода, если он уже выполняется в текущий момент времени другим потоком используется ключевое слово `synchronized`:
`public synchronized void f(){ . . . }`

Блокировка объектов может быть реализована следующим образом:

```
public synchronized void f(){
     synchronized (this){
          . . .
     }
}
```

В скобках может быть указан любой объект, необходимый для ограничения к нему доступа.

6. Реализуйте анализ некоторого текстового документа (записанного в строку) двумя потоками. При этом, первый поток должен составить контейнер с частотами появления всех символов, содержащихся в тексте, и записать его в общедоступную переменную типа `Map` главного запускаемого класса. После окончания вычислений, необходимо вывести информацию об окончании работы с документом. Второй поток ждет освобождения доступа к документу, пока первый поток не завершит работу, после чего он определяет три наиболее и три наименее встречающихся в тексте символа и выводит символ и частоту его появления в тексте на экран, после чего осуществляется вывод сообщения об окончании работы с документом. При реализации использовать `synchronized` и `volatile`. Документ должен содержать не менее 1000 символов.

7. Создайте файл `Server.java`, демонстрирующий работу с серверными сокетами:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

public class Server extends Thread {
    private int port;
    public Server(int port) {
        this.port = port;
    }
    @Override
    public void run() {
        BufferedReader in = null;
        PrintWriter out= null;
        ServerSocket server = null;
        Socket fromClient = null;
        try {
            server = new ServerSocket(port);
            fromClient = server.accept();
            in = new BufferedReader(new InputStreamReader(
                    fromClient.getInputStream()));
            out = new PrintWriter(fromClient.getOutputStream(),
true);
            String input;
            System.out.println("Client "
+in.readLine().split(":")[0]+ " connected");
            while ((input = in.readLine()) != null) {
                if (input.contains("exit")) {
                    System.out.println("Shutting down...");
                    break;
                }
                else {
                    String[] msg = input.split(":");
                    out.println("received from " + msg[0] + " :
" + msg[1]);
                }
```

```
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                out.close();
                in.close();
                fromClient.close();
                server.close();
            } catch (IOException ex) {}
        }
    }
}
```

8. Создайте файл `Client.java`, демонстрирующий работу с клиентскими сокетами:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;

public class Client extends Thread {
    private String host;
    private int port;
    private int id;
    public Client(String host, int port, int id) {
        this.host = host;
        this.port = port;
        this.id = id;
    }
    private String formatMessage(String msg) {
        return "client#" + this.id + ":" + msg;
    }
    @Override
    public void run() {
        Socket fromServer = null;
        BufferedReader in = null;
        PrintWriter out = null;
        BufferedReader inputBuf = null;
        try {
            String clientMsg, serverMsg;
            System.out.println("Connecting to " + host + ":" +
port);
            fromServer = new Socket(host, port);
            in = new BufferedReader(new
InputStreamReader(fromServer.getInputStream()));
            out = new PrintWriter(fromServer.getOutputStream(),
true);
            inputBuf = new BufferedReader(new
InputStreamReader(System.in));
            // start message (to be sure client is connected)
            out.println(formatMessage(""));
```

```
            // read message from console and send to server
            while ((clientMsg = inputBuf.readLine()) != null) {
                out.println(formatMessage(clientMsg));
                serverMsg = in.readLine();
                System.out.println("Server: "+serverMsg);
                if (clientMsg.contains("exit")) {
                    break;
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                out.close();
                in.close();
                inputBuf.close();
                fromServer.close();
            } catch (IOException ex) {}
        }
    }
}
```

9. Создайте файл `SocketExample.java`, демонстрирующий работу с сокетами:

```
public class SocketExample {
    public static void main(String[] args) {
        String host = "0.0.0.0";
        int port = 32000;
        Thread server = new Server(port);
        // server thread
        server.start();
        Thread client = new Client(host, port, 1);
        // client thread
        client.start();
    }
}
```

Запустите программу и удостоверьтесь в ее работоспособности.
Ознакомьтесь с выведенной информацией. Пример результата выполнения программы приведен ниже:

```
Connecting to 0.0.0.0:32000
Client client#1 connected
Hello
Server: received from client#1 : Hello
Have a nice day!
Server: received from client#1 : Have a nice day!
exit
Shutting down...
Server: null
```

Работа с сокетами

Сокеты являются низкоуровневыми сетевыми соединениями и позволяют осуществлять быстрый обмен данными, в частности при создании клиент-серверных приложений. Для каждого из клиентов приложения должны быть созданы клиентские сокеты (объекты типа `Socket`), а для серверов – серверные сокеты – объекты типа `ServerSocket`.

Взаимодействие происходит следующим образом – клиенту известен порт и адрес сервера. Изначально клиент запрашивает у сервера доступ к нему и устанавливается соединение. Клиент может отправлять некоторые сообщения серверу и ожидать от него ответа. Сервер принимает некоторые сообщения или запросы от клиента и отсылает ему ответы.

В примере рассмотрена реализация некоторого echo-сервера, который устанавливает соединение с клиентом и, получив от клиента сообщение, отсылает его ему обратно. `PrintWriter` позволяет реализовать потоковую отправку текстовых сообщений. Создание объекта типа `PrintWriter` происходит следующим образом:

```
PrintWriter out = new PrintWriter(fromClient.getOutputStream(), true);
```

Первый аргумент представленного конструктора – поток для отправки (записи) сообщений клиенту, второй аргумент – признак автоматической отправки данных.

При реализации работы сокетного взаимодействия нескольких клиентов и некоторого сервера необходимо для каждого сокета-клиента создавать серверный поток для осуществления попарного взаимодействия. Для этого необходимо связывать идентификационные данные клиента и сервера, чтобы не возникло коллизий при передаче сообщений.

10. Создайте пакет `chat`. Разработайте клиент-серверное приложение на основе сокетов, реализующее функции чата для двух клиентов. Клиенты могут отправлять друг другу сообщения. Сообщение от клиента попадает на сервер, сервер его обрабатывает и возвращает сообщение второму клиенту. Необходимо осуществить ввод сообщений с клавиатуры для обоих клиентов и вывод переданных сервером сообщений от другого клиента (вместе с именем-идентификатором второго клиента) в консоль. Вывод для каждого клиента осуществлять в его консоль в следующем формате:

```
Me:
. . . // сообщение введенное с клавиатуры текущим клиентом
Other: // идентификатор второго клиента
. . . // сообщение-ответ второго клиента
...
```

11. Создайте пакет `multiapp`. Разработайте многопользовательское клиент-серверное приложение на основе сокетов, реализующее функции автоматизированного бота, принимающего сообщения-команды от клиентов и выполняющего действия в соответствии с переданной командой согласно вашему варианту. К боту (серверу) могут одновременно подключаться несколько клиентов, при этом их запросы обрабатываются параллельно, информация, полученная от экземпляра бота для первого клиента, не отображается второму клиенту и т.д. При вводе команд указываются параметры, установленные вашим вариантом. Для формирования набора данных, возвращаемых ботом клиенту, проанализируйте и промоделируйте предметную область, к которой относится ваш бот, создав соответствующий набор классов. Для созданной модели

системы сформируйте набор данных, состоящий из не менее чем 10 объектов с различными параметрами.

При подключении клиента к боту перед началом взаимодействия с ним посредством команд необходимо выполнить следующие действия:

1) Автоматически при подключении отправить от лица бота сообщение с уточнением имени клиента, после чего поприветствовать клиента с указанным им именем
2) Отобразить клиенту доступный перечень команд

Формат каждой из команд, отправляемых боту:
`/[command-name] [param]`
Пример команды:
`/find city`

Вывод для каждого клиента осуществлять в его консоль в следующем формате:

```
[BOT]:
. . . // сообщение бота
[Me]:
. . . // сообщение клиента
[BOT]:
. . . // сообщение бота
[Name]: // имя необходимо подставить после указания его клиентом
. . . // сообщение-команда клиента
```

Вариант	**Описание систем**
1	Бот: Система подбора сериалов. В зависимости от введенной клиентом команды выдается перечень объектов, подходящих под переданные параметры, с указанием полной информации по каждому объекту. Команды, доступные клиентам: [1] – поиск по жанру (параметр – жанр) [2] – поиск по году выпуска (параметр – год выпуска) [3] – поиск по названию (параметр – название, учесть, что может быть введено неполное название) [4] – поиск по языку (параметр – язык сериала, учесть, что для сериала может быть указано наличие субтитров с нужным языком) [5] – просмотр общего перечня сериалов

2	Бот: Система «умный дом». В зависимости от введенной клиентом команды происходит изменение параметров системы и выдается полная информация о текущем ее состоянии. Команды, доступные клиентам: [1] – установка температуры помещения (параметры – температура [пробел] название помещения-комнаты в доме) [2] – включение/выключение света (параметр – состояние освещения {true, false}, учесть невозможность выключения выключенного света и включения включенного) [3] – открытие/закрытие форточки (параметр – состояние форточки {true, false}, учесть невозможность закрытия закрытой форточки и открытия открытой) [4] – просмотр данных вебкамеры помещения (параметр – название помещения, выводит данные вебкамеры в символьном виде) [5] – включение/выключение сигнализации (параметр – состояние сигнализации {true, false}, учесть невозможность выключения выключенной сигнализации и включения включенной)
3	Бот: Система просмотра информации о погодных условиях. В зависимости от введенной клиентом команды выдается перечень объектов, подходящих под переданные параметры, с указанием полной информации по каждому объекту. Команды, доступные клиентам: [1] – получение информации о погоде в городе (параметр – город) [2] – получение информации о погоде в городе на определенный день (параметры – город [пробел] дата) [3] – получение информации о погоде в городе на определенный период времени (параметры – город [пробел] дата начала [пробел] дата окончания) [4] – получение информации о городах с заданными температурными условиями (параметры – минимальная температура [пробел] максимальная температура) [5] – получение информации о городах с заданными параметрами осадков (параметр – наличие осадков {true,false})
4	Бот: Система подбора смартфонов. В зависимости от введенной клиентом команды выдается перечень объектов, подходящих под переданные параметры, с указанием полной информации по каждому объекту. Команды, доступные клиентам: [1] – поиск по производителю (параметр – название компании-производителя) [2] – поиск по году выпуска (параметр – год выпуска) [3] – поиск по диагонали экрана (параметр – минимальное значение диагонали экрана) [4] – поиск по марке процессора и внутренней памяти (параметры – марка процессора [пробел] объем памяти в Гб, учесть, что может быть указано неполное наименование марки процессора) [5] – поиск по поддерживаемым сетям (параметры – сеть_01 [пробел] сеть_02 и т.д., например, 4G, Wi-Fi, Bluetooth, …,)

5	Бот: Система просмотра новостной ленты. В зависимости от введенной клиентом команды выдается перечень объектов, подходящих под переданные параметры, с указанием полной информации по каждому объекту. Команды, доступные клиентам: [1] – получение информации о новостях определенной категории (параметр – наименование категории) [2] – получение информации о новостях на определенный день (параметр –дата) [3] – получение информации о новостях определенной категории на определенный день (параметры – наименование категории [пробел] дата) [4] – поиск новостей по названию (параметр – название, учесть, что может быть указано неполное название) [5] – поиск новостей по участникам (параметры – участник_01 [пробел] участник_02 и т.д., например, дуров, роскомнадзор, …)
6	Бот: Система заказа такси. В зависимости от введенных клиентом данных бот запрашивает дополнительную необходимую ему информацию и после получения всех данных выдает полную информацию о сформированном заказе. Если заказ не может быть сформирован – выдается соответствующая информация. Команды, доступные клиентам: [1] – вызвать такси (начало взаимодействия с ботом) [2] – подтвердить заказ Уточняющие вопросы от бота: [1] – укажите пункт отправления [2] – укажите пункт прибытия [3] – укажите максимальное желаемое время в пути [4] – укажите наличие багажа [5] – укажите наличие ребенка до 10 лет [6] – укажите общее количество человек
7	Бот: Система заказа пиццы. В зависимости от введенных клиентом данных бот запрашивает дополнительную необходимую ему информацию и после получения всех данных выдает полную информацию о сформированном заказе. Если заказ не может быть сформирован – выдается соответствующая информация. Команды, доступные клиентам: [1] – сформировать заказ (начало взаимодействия с ботом) [2] – подтвердить заказ Уточняющие вопросы от бота: [1] – укажите название пиццы и их количество [2] – перейти к заказу напитков? {да, нет} [3] – укажите название напитка и их количество [4] – рассчитать предварительную стоимость заказа? {да, нет} [5] – закончить формирование заказа? {да, нет}

8	Бот: Система подбора книг. В зависимости от введенной клиентом команды выдается перечень объектов, подходящих под переданные параметры, с указанием полной информации по каждому объекту. Команды, доступные клиентам: [1] – поиск по жанру (параметр – жанр) [2] – поиск по годам издания (параметры – начальный год издания [пробел] конечный год издания) [3] – поиск по названию книги (параметр – название книги, учесть, что может быть введено неполное название) [4] – поиск по языку (параметр – язык книги, учесть, что может быть указано наличие субтитров с нужным языком) [5] – поиск по издательству (параметр – название издательства, учесть, что может быть введено неполное название)
9	Бот: Система поиска друзей в социальной сети. В зависимости от введенной клиентом команды выдается перечень объектов, подходящих под переданные параметры, с указанием полной информации по каждому объекту. Команды, доступные клиентам: [1] – поиск по ФИО (параметры – имя [пробел] фамилия) [2] – поиск по годам рождения (параметры – начальный год рождения [пробел] конечный год рождения) [3] – поиск по месту проживания (параметр – название места проживания, учесть, что может быть введено неполное название) [4] – поиск по образовательному учреждению (параметр – название учреждения, учесть, что может быть указано несколько образовательных учреждений для каждого человека) [5] – поиск по образовательному учреждению и году выпуска (параметры – название учреждения [пробел] год выпуска, учесть, что может быть введено неполное название)
10	Бот: Система бронирования билетов на рейс. В зависимости от введенных клиентом данных бот запрашивает дополнительную необходимую ему информацию и после получения всех данных выдает полную информацию о сформированном заказе. Если заказ не может быть сформирован – выдается соответствующая информация. Команды, доступные клиентам: [1] – бронировать билеты (начало взаимодействия с ботом) [2] – подтвердить заказ Уточняющие вопросы от бота: [1] – укажите пункт отправления [2] – укажите пункт прибытия [3] – укажите желаемую дату отправления [4] – укажите максимальное желаемое время в пути [5] – укажите наличие габаритного багажа {да, нет} [6] – укажите общее количество билетов

11	Бот: Система заказа билетов в кинотеатр. В зависимости от введенных клиентом данных бот запрашивает дополнительную необходимую ему информацию и после получения всех данных выдает полную информацию о сформированном заказе. Если заказ не может быть сформирован – выдается соответствующая информация. Команды, доступные клиентам: [1] – просмотреть информацию об идущих в кинотеатре фильмах [2] – заказать билеты (начало взаимодействия с ботом) [3] – подтвердить заказ Уточняющие вопросы от бота: [1] – укажите название фильма [2] – укажите желаемую дату [3] – укажите желание просмотра в VIP-зале {да, нет} [4] – укажите диапазон рядов в зале {1-10} [5] – укажите положение мест в ряду {край, центр}
12	Бот: Система подбора онлайн-курсов. В зависимости от введенной клиентом команды выдается перечень объектов, подходящих под переданные параметры, с указанием полной информации по каждому объекту. Команды, доступные клиентам: [1] – поиск по направлению (параметр – название направления, учесть, что может быть введено неполное название) [2] – поиск по годам размещения (параметры – начальный год размещения [пробел] конечный год размещения) [3] – поиск по названию курса (параметр – название курса, учесть, что может быть введено неполное название) [4] – поиск по языку (параметр – язык курса, учесть, что может быть указано наличие субтитров с нужным языком) [5] – поиск по университету-производителю (параметр – название университета, учесть, что может быть введено неполное название)
13	Бот: Система заказа билетов в театр. В зависимости от введенных клиентом данных бот запрашивает дополнительную необходимую ему информацию и после получения всех данных выдает полную информацию о сформированном заказе. Если заказ не может быть сформирован – выдается соответствующая информация. Команды, доступные клиентам: [1] – просмотреть информацию об идущих в театре представлениях [2] – заказать билеты (начало взаимодействия с ботом) [3] – подтвердить заказ Уточняющие вопросы от бота: [1] – укажите название представления [2] – укажите желаемую дату [3] – укажите ложу {партер, амфитеатр, бельэтаж, балкон} [4] – укажите ряд [5] – укажите положение мест в ряду {край, центр}

14	Бот: Система заказа еды в супермаркете. В зависимости от введенных клиентом данных бот запрашивает дополнительную необходимую ему информацию и после получения всех данных выдает полную информацию о сформированном заказе. Если заказ не может быть сформирован – выдается соответствующая информация. Команды, доступные клиентам: [1] – просмотреть перечень продукции [2] – сформировать заказ (начало взаимодействия с ботом) [3] – подтвердить заказ Уточняющие вопросы от бота: [1] – укажите название продукта и их количество [2] – рассчитать предварительную стоимость заказа? {да, нет} [3] – закончить формирование заказа? {да, нет}
15	Бот: Система подбора ноутбуков. В зависимости от введенной клиентом команды выдается перечень объектов, подходящих под переданные параметры, с указанием полной информации по каждому объекту. Команды, доступные клиентам: [1] – поиск по производителю (параметр – название компании-производителя) [2] – поиск по году выпуска (параметр – год выпуска) [3] – поиск по диагонали экрана (параметр – минимальное значение диагонали экрана) [4] – поиск по марке процессора и внутренней памяти (параметры – марка процессора [пробел] объем памяти в Гб, учесть, что может быть указано неполное наименование марки процессора) [5] – поиск по цене (параметры – минимальная цена [пробел] максимальная цена)
16	Бот: Система заказа лекарств в аптеке. В зависимости от введенных клиентом данных бот запрашивает дополнительную необходимую ему информацию и после получения всех данных выдает полную информацию о сформированном заказе. Если заказ не может быть сформирован – выдается соответствующая информация. Команды, доступные клиентам: [1] – просмотреть перечень продукции [2] – сформировать заказ (начало взаимодействия с ботом) [3] – подтвердить заказ Уточняющие вопросы от бота: [1] – укажите название продукта и их количество [2] – рассчитать предварительную стоимость заказа? {да, нет} [3] – закончить формирование заказа? {да, нет}
17	Бот: Система формирования плана тренировки. В зависимости от введенных клиентом данных бот запрашивает дополнительную необходимую ему информацию и после получения всех данных выдает полную информацию о сформированном плане. Если план не может быть сформирован – выдается соответствующая информация. Команды, доступные клиентам: [1] – просмотреть список тренажеров [2] – сформировать план (начало взаимодействия с ботом) [3] – подтвердить план Уточняющие вопросы от бота: [1] – укажите тип `n`-ой тренировки {кардио, силовые и т.д.} [2] – укажите желаемое время длительности тренировки [3] – укажите желаемый тренажер (бот должен проверить занятость тренажера в этот момент) [4] – закончить формирование плана? {да, нет}

18	Бот: Система бронирования помещения под конференцию. В зависимости от введенных клиентом данных бот запрашивает дополнительную необходимую ему информацию и после получения всех данных выдает полную информацию о сформированном заказе. Если заказ не может быть сформирован – выдается соответствующая информация. Команды, доступные клиентам: [1] – просмотреть информацию о доступных помещениях [2] – бронировать помещение (начало взаимодействия с ботом) [3] – подтвердить бронь Уточняющие вопросы от бота: [1] – укажите город проведения конференции [2] – укажите желаемый диапазон дат [3] – укажите количество человек [4] – укажите необходимость пространства под фуршет [5] – укажите требования технической оснащенности помещения
19	Бот: Система подбора подарков. В зависимости от введенной клиентом команды выдается перечень объектов, подходящих под переданные параметры, с указанием полной информации по каждому объекту. Команды, доступные клиентам: [1] – поиск по производителю (параметр – название компании-производителя) [2] – поиск по тематике (параметр – наименование тематики, например, день рождения, 23 февраля, новый год и т.д., учесть, что может быть указано неполное наименование) [3] – поиск по стоимости (параметр – минимальная цен [пробел] максимальная цена) [4] – поиск по категории подарка (параметр – наименование категории, например, бытовая техника, сувениры, садоводство и т.д., учесть, что может быть указано неполное наименование) [5] – поиск по адресату (параметр – тип адресата, например, мужчине, женщине, девочке, мальчику, …)
20	Бот: Система подбора компьютерных игр. В зависимости от введенной клиентом команды выдается перечень объектов, подходящих под переданные параметры, с указанием полной информации по каждому объекту. Команды, доступные клиентам: [1] – поиск по производителю (параметр – название компании-производителя, учесть, что может быть указано неполное название) [2] – поиск по рейтингу (параметр – минимальное значение рейтинга) [3] – поиск по категории (параметр – наименование категории, учесть, что может быть указано неполное название) [4] – поиск по годам выпуска (параметры – минимальное значение года [пробел] максимальное значение года) [5] – поиск по типу (параметры – тип, например, мультиплеер, одиночная)

21	Бот: Система подбора фильмов. В зависимости от введенной клиентом команды выдается перечень объектов, подходящих под переданные параметры, с указанием полной информации по каждому объекту. Команды, доступные клиентам: [1] – поиск по жанру (параметр – жанр) [2] – поиск по году выпуска (параметр – год выпуска) [3] – поиск по названию (параметр – название, учесть, что может быть введено неполное название) [4] – поиск по языку (параметр – язык фильма, учесть, что может быть указано наличие субтитров с нужным языком) [5] – просмотр общего перечня фильмов
22	Бот: Система подбора места отдыха. В зависимости от введенной клиентом команды выдается перечень объектов, подходящих под переданные параметры, с указанием полной информации по каждому объекту. Команды, доступные клиентам: [1] – поиск по региону (параметр – название региона, учесть, что может быть указано неполное название) [2] – поиск по рейтингу (параметр – минимальное значение рейтинга места) [3] – поиск по названию отеля (параметр – название отеля, учесть, что может быть указано неполное название) [4] – поиск по категории отдыха (параметр – наименование категории, например, активный, спокойный, семейный и т.д.) [5] – поиск по близости к морю (параметр – максимальное расстояние до моря)
23	Бот: Система подбора квартиры. В зависимости от введенной клиентом команды выдается перечень объектов, подходящих под переданные параметры, с указанием полной информации по каждому объекту. Команды, доступные клиентам: [1] – поиск по региону (параметр – название региона, учесть, что может быть указано неполное название) [2] – поиск по рейтингу строительной компании (параметр – минимальное значение рейтинга) [3] – поиск по количеству комнат (параметр – количество комнат) [4] – поиск по категории комфорта (параметр – наименование категории, например, эконом, комфорт, бизнес и т.д.) [5] – поиск по стоимости (параметры – минимальное значение стоимости [пробел] максимальное значение стоимости)
24	Бот: Система подбора рецептов. В зависимости от введенной клиентом команды выдается перечень объектов, подходящих под переданные параметры, с указанием полной информации по каждому объекту. Команды, доступные клиентам: [1] – поиск по региону (параметр – название региона, к которому относится рецепт, учесть, что может быть указано неполное название) [2] – поиск по рейтингу (параметры – минимальное значение рейтинга рецепта) [3] – поиск по времени приготовления (параметры – минимальное значение времени [пробел] максимальное значение времени) [4] – поиск по категории блюда (параметр – наименование категории, например, завтрак, обед, десерт и т.д.) [5] – поиск по ингредиентам (параметры – ингредиент_01 [пробел] ингредиент _02 и т.д., например, сыр, сливки, …,)

25	Бот: Система подбора PR-менеджера. В зависимости от введенной клиентом команды выдается перечень объектов, подходящих под переданные параметры, с указанием полной информации по каждому объекту. Команды, доступные клиентам: [1] – поиск по количеству выполненных проектов (параметр – минимальное количество выполненных проектов) [2] – поиск по опыту работы (параметр – количество лет работы на предприятиях) [3] – поиск по диапазону возраста человека (параметры – минимальное значение [пробел] максимальное значение) [4] – поиск по диапазону желаемой заработной платы (параметры – минимальное значение [пробел] максимальное значение) [5] – поиск по наличию сертификатов повышения квалификации (параметр – флаг получения сертификатов {да, нет})
26	Бот: Система подбора экскурсионных маршрутов. В зависимости от введенной клиентом команды выдается перечень объектов, подходящих под переданные параметры, с указанием полной информации по каждому объекту. Команды, доступные клиентам: [1] – поиск по стране (параметр – название страны) [2] – поиск по городу (параметр – название города) [3] – поиск по времени (параметры – минимальное значение [пробел] максимальное значение) [4] – поиск по способу передвижения (параметр – название вида транспорта) [5] – поиск по категории (параметр – название категории, например, активный маршрут, спокойный, сухопутный, морской и т.д.)
27	Бот: Система подбора сотрудника HR-менеджера. В зависимости от введенной клиентом команды выдается перечень объектов, подходящих под переданные параметры, с указанием полной информации по каждому объекту. Команды, доступные клиентам: [1] – поиск по количеству найденных ранее сотрудников (параметр – минимальное количество сотрудников) [2] – поиск по опыту работы (параметр – количество лет работы на предприятиях) [3] – поиск по диапазону возраста человека (параметры – минимальное значение [пробел] максимальное значение) [4] – поиск по диапазону желаемой заработной платы (параметры – минимальное значение [пробел] максимальное значение) [5] – поиск по наличию сертификатов повышения квалификации (параметр – флаг получения сертификатов {да, нет})
28	Бот: Система бронирования танцевального зала. В зависимости от введенных клиентом данных бот запрашивает дополнительную необходимую ему информацию и после получения всех данных выдает полную информацию о сформированном заказе. Если заказ не может быть сформирован – выдается соответствующая информация. Команды, доступные клиентам: [1] – просмотреть информацию о доступных помещениях [2] – бронировать помещение (начало взаимодействия с ботом) [3] – подтвердить бронь Уточняющие вопросы от бота: [1] – укажите город проведения бала [2] – укажите желаемый диапазон дат [3] – укажите количество человек [4] – укажите площадь танцпола [5] – укажите требования технической оснащенности помещения

29	Бот: Система подбора сотрудника-программиста. В зависимости от введенной клиентом команды выдается перечень объектов, подходящих под переданные параметры, с указанием полной информации по каждому объекту. Команды, доступные клиентам: [1] – поиск по языку программирования (параметр – название языка программирования) [2] – поиск по опыту работы (параметр – количество лет работы на предприятиях) [3] – поиск по диапазону возраста человека (параметры – минимальное значение [пробел] максимальное значение) [4] – поиск по диапазону желаемой заработной платы (параметры – минимальное значение [пробел] максимальное значение) [5] – поиск по наличию сертификатов повышения квалификации (параметр – флаг получения сертификатов {да, нет})
30	Бот: Система бронирования помещения под выставку. В зависимости от введенных клиентом данных бот запрашивает дополнительную необходимую ему информацию и после получения всех данных выдает полную информацию о сформированном заказе. Если заказ не может быть сформирован – выдается соответствующая информация. Команды, доступные клиентам: [1] – просмотреть информацию о доступных помещениях [2] – бронировать помещение (начало взаимодействия с ботом) [3] – подтвердить бронь Уточняющие вопросы от бота: [1] – укажите город проведения выставки [2] – укажите желаемый диапазон дат [3] – укажите количество секций [4] – укажите площадь под каждую секцию [5] – укажите требования технической оснащенности помещения
31	Бот: Система подбора тренировочных маршрутов для бега. В зависимости от введенной клиентом команды выдается перечень объектов, подходящих под переданные параметры, с указанием полной информации по каждому объекту. Команды, доступные клиентам: [1] – поиск по городу (параметр – название города) [2] – поиск по городу и району (параметры – название города [пробел] название района) [3] – поиск по времени (параметры – минимальное значение [пробел] максимальное значение) [4] – поиск по сложности (параметр – уровень сложности) [5] – поиск по категории (параметр – название категории, например, интенсивный маршрут, спокойный)
32	Бот: Система подбора принтеров. В зависимости от введенной клиентом команды выдается перечень объектов, подходящих под переданные параметры, с указанием полной информации по каждому объекту. Команды, доступные клиентам: [1] – поиск по производителю (параметр – название компании-производителя) [2] – поиск по году выпуска (параметр – год выпуска) [3] – поиск по технологии печати (параметр – технология печати, учесть, что может быть введено неполное наименование) [4] – поиск по диапазону скоростей печати (параметры – минимальная скорость печати [пробел] максимальная скорость печати) [5] – поиск по размеру бумаги (параметры – минимальный размер бумаги [пробел] максимальный размер бумаги)

33	Бот: Система подбора домашнего животного. В зависимости от введенной клиентом команды выдается перечень объектов, подходящих под переданные параметры, с указанием полной информации по каждому объекту. Команды, доступные клиентам: [1] – поиск по виду животного (параметр – название вида) [2] – поиск по окрасу (параметр – описание окраски, учесть, что может быть указано неполное описание) [3] – поиск по диапазону возраста (параметры – минимальное значение [пробел] максимальное значение) [4] – поиск по породе (параметр – название породы) [5] – поиск по необходимости выгуливания (параметр – флаг необходимости выгуливания {да, нет})
34	Бот: Система бронирования ресторана. В зависимости от введенных клиентом данных бот запрашивает дополнительную необходимую ему информацию и после получения всех данных выдает полную информацию о сформированном заказе. Если заказ не может быть сформирован – выдается соответствующая информация. Команды, доступные клиентам: [1] – просмотреть информацию о доступных ресторанах [2] – бронировать ресторан (начало взаимодействия с ботом) [3] – подтвердить бронь Уточняющие вопросы от бота: [1] – укажите местоположение ресторана (город) [2] – укажите желаемый диапазон дат [3] – укажите количество человек [4] – укажите тип кухни (блюд) [5] – укажите требования технической оснащенности ресторана
35	Бот: Система подбора администратора. В зависимости от введенной клиентом команды выдается перечень объектов, подходящих под переданные параметры, с указанием полной информации по каждому объекту. Команды, доступные клиентам: [1] – поиск по желаемому местоположению (параметр – местоположение, учесть, что может быть указано неполное название) [2] – поиск по опыту работы (параметр – количество лет работы на предприятиях) [3] – поиск по диапазону возраста человека (параметры – минимальное значение [пробел] максимальное значение) [4] – поиск по диапазону желаемой заработной платы (параметры – минимальное значение [пробел] максимальное значение) [5] – поиск по наличию сертификатов повышения квалификации (параметр – флаг получения сертификатов {да, нет})
36	Бот: Система подбора дизайнера. В зависимости от введенной клиентом команды выдается перечень объектов, подходящих под переданные параметры, с указанием полной информации по каждому объекту. Команды, доступные клиентам: [1] – поиск по количеству выполненных проектов (параметр – количество выполненных проектов) [2] – поиск по опыту работы (параметр – количество лет работы на предприятиях) [3] – поиск по диапазону возраста человека (параметры – минимальное значение [пробел] максимальное значение) [4] – поиск по диапазону желаемой заработной платы (параметры – минимальное значение [пробел] максимальное значение) [5] – поиск по наличию сертификатов повышения квалификации (параметр – флаг получения сертификатов {да, нет})

37	Бот: Система бронирования теплохода. В зависимости от введенных клиентом данных бот запрашивает дополнительную необходимую ему информацию и после получения всех данных выдает полную информацию о сформированном заказе. Если заказ не может быть сформирован – выдается соответствующая информация. Команды, доступные клиентам: [1] – просмотреть информацию о доступных теплоходах [2] – бронировать теплоход (начало взаимодействия с ботом) [3] – подтвердить бронь Уточняющие вопросы от бота: [1] – укажите местоположение теплохода (город) [2] – укажите желаемый диапазон дат [3] – укажите количество человек [4] – укажите тип кухни (блюд) [5] – укажите требования технической оснащенности теплохода
38	Бот: Система получения информации о маршрутах. В зависимости от введенной клиентом команды выдается перечень маршрутов, подходящих под переданные параметры, с указанием полной информации по каждому маршруту. Команды, доступные клиентам: [1] – поиск по пункту отправления (параметр – пункт отправления, выдаются все маршруты из данного пункта) [2] – поиск по пунктам отправления/прибытия (параметры – пункт отправления и пункт прибытия) [3] – поиск по пунктам отправления/прибытия и способу передвижения (параметры – пункт отправления [пробел] пункт прибытия [пробел] вид транспорта) [4] – поиск по пунктам отправления/прибытия и времени в пути (параметр – пункт отправления [пробел] пункт прибытия [пробел] максимальное время в пути, выдаются маршруты, время которых не превышает заданное) [5] – поиск по пунктам отправления/прибытия, способу передвижения, времени в пути (параметры – пункт отправления [пробел] пункт прибытия [пробел] вид транспорта [пробел] максимальное время в пути)
39	Бот: Система бронирования времени для подачи документов. В зависимости от введенных клиентом данных бот запрашивает дополнительную необходимую ему информацию и после получения всех данных выдает полную информацию о сформированном заказе. Если заказ не может быть сформирован – выдается соответствующая информация. Команды, доступные клиентам: [1] – просмотреть общее расписание офиса приема документов (доступное для записи время) [2] – бронировать время (начало взаимодействия с ботом) [3] – подтвердить бронь Уточняющие вопросы от бота: [1] – укажите место подачи [2] – укажите желаемые дату и время подачи (бот должен проверить занятость, и предложить другие ближайшие доступные варианты, если желаемое время недоступно) [3] – выберите время из предложенных (клиент должен указать порядковый номер желаемой записи из общего списка предложенных) [3] – завершить процедуру? {да, нет}

40	Бот: Система подбора автомобиля. В зависимости от введенной клиентом команды выдается перечень объектов, подходящих под переданные параметры, с указанием полной информации по каждому объекту. Команды, доступные клиентам: [1] – поиск по производителю (параметр – название компании-производителя) [2] – поиск по году выпуска (параметр – год выпуска) [3] – поиск по типу корпуса (параметр – название типа корпуса, например, седан, хетчбэк и т.д.) [4] – поиск по пробегу (параметр – значение пробега) [5] – поиск по стоимости (параметры – минимальное значение [пробел] максимальное значение)
41	Бот: Система сбора отзывов об отеле. В зависимости от введенных клиентом данных бот запрашивает дополнительную необходимую ему информацию и после получения всех данных выдает полную информацию об отзыве. Команды, доступные клиентам: [1] – просмотреть перечень отелей [2] – написать отзыв (начало взаимодействия с ботом) [3] – отправить отзыв Уточняющие вопросы от бота: [1] – укажите название отеля [2] – укажите дату и время вашего последнего посещения отеля [3] – укажите ваш рейтинг для отеля [4] – удовлетворены ли вы обслуживанием? {да, нет} [5] – удовлетворены ли вы качеством интернет-связи? {да, нет} [6] – добавьте текстовое описание
42	Бот: Система бронирования билетов на паром. В зависимости от введенных клиентом данных бот запрашивает дополнительную необходимую ему информацию и после получения всех данных выдает полную информацию о сформированном заказе. Если заказ не может быть сформирован – выдается соответствующая информация. Команды, доступные клиентам: [1] – бронировать билеты (начало взаимодействия с ботом) [2] – подтвердить заказ Уточняющие вопросы от бота: [1] – укажите пункт отправления [2] – укажите пункт прибытия [3] – укажите желаемую дату отправления [4] – укажите максимальное желаемое время в пути [5] – укажите наличие габаритного багажа {да, нет} [6] – укажите необходимость места для автомобиля {да, нет} [7] – укажите общее количество билетов

43	Бот: Система бронирования услуги. В зависимости от введенных клиентом данных бот запрашивает дополнительную необходимую ему информацию и после получения всех данных выдает полную информацию о сформированном заказе. Если заказ не может быть сформирован – выдается соответствующая информация. Команды, доступные клиентам: [1] – просмотреть перечень услуг [2] – сформировать заказ (начало взаимодействия с ботом) [3] – подтвердить заказ Уточняющие вопросы от бота: [1] – укажите название услуги [2] – укажите желаемые дату и время [3] – рассчитать предварительную стоимость? {да, нет} [4] – закончить бронирование? {да, нет}
44	Бот: Система сбора отзывов о работе магазина. В зависимости от введенных клиентом данных бот запрашивает дополнительную необходимую ему информацию и после получения всех данных выдает полную информацию об отзыве. Команды, доступные клиентам: [1] – просмотреть перечень магазинов [2] – написать отзыв (начало взаимодействия с ботом) [3] – отправить отзыв Уточняющие вопросы от бота: [1] – укажите название магазина [2] – укажите дату и время вашего последнего посещения магазина [3] – укажите ваш рейтинг для магазина [4] – удовлетворены ли вы обслуживанием? {да, нет} [5] – удовлетворены ли вы ассортиментом? {да, нет} [6] – удовлетворены ли вы качеством продукции? {да, нет} [7] – добавьте текстовое описание
45	Бот: Система оплаты счета за интернет-услуги. В зависимости от введенных клиентом данных бот запрашивает дополнительную необходимую ему информацию и после получения всех данных выдает полную информацию об оплате. Команды, доступные клиентам: [1] – просмотреть перечень интернет-услуг [2] – заказ услуги (начало взаимодействия с ботом) [3] – оплатить Уточняющие вопросы от бота: *Для команды [2]* [1] – укажите название услуги [2] – дату начала и окончания использования услуги [3] – рассчитать предварительную стоимость? {да, нет} *Для команды [3]* [4] – укажите способ оплаты [5] – укажите платежную информацию? {да, нет} [6] – подтвердить оплату? {да, нет}

46	Бот: Система подбора мотоцикла. В зависимости от введенной клиентом команды выдается перечень объектов, подходящих под переданные параметры, с указанием полной информации по каждому объекту. Команды, доступные клиентам: [1] – поиск по производителю (параметр – название компании-производителя) [2] – поиск по году выпуска (параметр – год выпуска) [3] – поиск по типу корпуса (параметр – название типа корпуса, например, седан, хетчбэк и т.д.) [4] – поиск по стоимости (параметры – параметры – минимальное значение [пробел] максимальное значение) [5] – поиск по мощности двигателя (параметры – минимальное значение [пробел] максимальное значение)
47	Бот: Система оплаты интернет-заказа. В зависимости от введенных клиентом данных бот запрашивает дополнительную необходимую ему информацию и после получения всех данных выдает полную информацию об оплате. Команды, доступные клиентам: [1] – просмотреть перечень интернет-услуг [2] – заказ услуги (начало взаимодействия с ботом) [3] – оплатить Уточняющие вопросы от бота: *Для команды [2]* [1] – укажите идентификатор заказа [2] – укажите желаемую дату получения заказа [3] – рассчитать предварительную стоимость? {да, нет} *Для команды [3]* [4] – укажите способ оплаты [5] – укажите платежную информацию? {да, нет} [6] – подтвердить оплату? {да, нет}
48	Бот: Система сбора отзывов о ресторане. В зависимости от введенных клиентом данных бот запрашивает дополнительную необходимую ему информацию и после получения всех данных выдает полную информацию об отзыве. Команды, доступные клиентам: [1] – просмотреть перечень ресторанов [2] – написать отзыв (начало взаимодействия с ботом) [3] – отправить отзыв Уточняющие вопросы от бота: [1] – укажите название ресторана [2] – укажите дату и время вашего последнего посещения ресторана [3] – укажите ваш рейтинг для ресторана [4] – удовлетворены ли вы обслуживанием? {да, нет} [5] – удовлетворены ли вы меню? {да, нет} [6] – добавьте текстовое описание

49	Бот: Система технической поддержки программного обеспечения. В зависимости от введенных клиентом данных бот запрашивает дополнительную необходимую ему информацию и после получения всех данных выдает полную информацию о дальнейших действиях клиента. Команды, доступные клиентам: [1] – сформировать запрос (начало взаимодействия с ботом) Уточняющие вопросы от бота: [1] – укажите название программного обеспечения (ПО) [2] – укажите версию используемого ПО [3] – укажите вашу проблему [4] – данная проблема систематическая или произошла впервые? {да, нет} [5] – далее приведено потенциальное решение возникшей проблемы: …
50	Бот: Система подбора фотоаппаратов. В зависимости от введенной клиентом команды выдается перечень объектов, подходящих под переданные параметры, с указанием полной информации по каждому объекту. Команды, доступные клиентам: [1] – поиск по производителю (параметр – название компании-производителя) [2] – поиск по году выпуска (параметр – год выпуска) [3] – поиск по разрешению матрицы (параметры – минимальное значение мегапикселей матрицы [пробел] максимальное значение мегапикселей матрицы) [4] – поиск по типу матрицы (параметр – тип матрицы фотоаппарата) [5] – поиск по оптическому zoom’у (параметры – минимальное значение [пробел] максимальное значение)