```java
import java.io.*;
import java.net.InetAddress;
import java.net.Socket;

/**
 * Класс {@code ClientConnection} представляет объект клиента, подключаемый к серверу для манипулирования коллекцией.
 * @author Артемий Кульбако
 * @version 1.2
 * @since 21.05.19
 */
class ClientConnection {

    private Commander commands;
    private static ClientConnection instance;

    public static ClientConnection getInstance() {
        if (instance == null) {
            instance = new ClientConnection();
        }
        return instance;
    }

    /**
     * Устанавливает активное соединение с сервером.
     */
    void start() {
        Console console = System.console();
            while (true) {
                try (Socket outcoming = new Socket(InetAddress.getLocalHost(), 8800)) {
                    outcoming.setSoTimeout(5000);
                    try (ObjectOutputStream outputStream = new ObjectOutputStream(outcoming.getOutputStream());
                         ObjectInputStream inputStream = new ObjectInputStream(outcoming.getInputStream())) {
                        commands = new Commander(console, outputStream, inputStream);
                        commands.interactiveMode();
                        System.out.println("Завершение программы.");
                    }
                } catch (IOException e) {
                    System.err.println("Нет связи с сервером. Подключиться ещё раз ({да} или {нет})?");
                    String answer;
                    while (!(answer = console.readLine()).equals("да")) {
                        switch (answer) {
                            case "":
                                break;
                            case "нет":
                                System.exit(0);
                                break;
                                default: System.out.println("Введите корректный ответ.");
                        }
                    }
                    System.out.print("Подключение ...");
                    //TODO запоминает логин и пароль и автоматически подключается
                    continue;
                }
            }
        }
    }
```

File - C:\Users\pugalol\Desktop\programming\lab7-09.05.19\Prometheus\src\ClientSide.java

```
1 public class ClientSide {
2 
3     /**
4      * Отправная точка клиента. Создает объект {@code ClientConnection}.
5      * @param args массив по умолчанию в основном методе. Не используется здесь.
6      */
7     public static void main(String[] args) {
8         System.out.println("Запуск клиентского модуля.\nПодключение к серверу ...");
9         ClientConnection.getInstance().start();
10        }
11 }
```

```java
import java.io.*;

class Commander {

    private Console fromKeyboard;
    private ObjectOutputStream toServer;
    private ObjectInputStream fromServer;

    Commander(Console fromKeyboard, ObjectOutputStream toServer, ObjectInputStream fromServer) {
        this.fromKeyboard = fromKeyboard;
        this.toServer = toServer;
        this.fromServer = fromServer;
    }

    /**
     * Парсит пользовательские команды и осуществляет обмен данными с сервером.
     * @throws IOException при отправке и получении данных с сервера.
     */
    void interactiveMode() throws IOException {
        try {
            System.out.println((String) fromServer.readObject()); //Приглашение ввода от сервера
            String command;
            while (fromKeyboard != null) {
                command = fromKeyboard.readLine();
                String[] parsedCommand = command.trim().split(" ", 2);
                try {
                    switch (parsedCommand[0]) {
                        case "":
                            break;
                        case "login":
                            login(parsedCommand[1]);
                            break;
                        case "import":
                            importingFile(parsedCommand[1]);
                            break;
                        case "exit":
                            System.exit(0);
                            break;
                        default:
                            toServer.writeObject(command);
                            System.out.println((String) fromServer.readObject());
                    }
                } catch (ArrayIndexOutOfBoundsException e) {
                    System.err.println("Отсутствует аргумент.");
                }
            }
        } catch (ClassNotFoundException e) {
            /* никогда не выбросит, так как используется String, но java требует явную обработку или указание
            в сигнатуре метода, так как это проверяемое исключение */
            System.err.println("Класс не найден: " + e.getMessage());
        }
    }

    /**
     * Импортирует локальный файл и отправляет на сервер.
     * @param path путь к файлу.
     * @throws SecurityException если отсутствуют права rw.
     * @throws IOException если файл не существует.
     */
    private void importingFile(String path) throws ClassNotFoundException {
        File localCollection = new File(path);
        String fileExtension = localCollection.getAbsolutePath().substring(localCollection.getAbsolutePath().lastIndexOf(".") + 1);
        try {
            if (!localCollection.exists() | localCollection.length() == 0  | !fileExtension.equals("json"))
                throw new FileNotFoundException();
            if (!localCollection.canRead()) throw new SecurityException();
            try (BufferedReader inputStreamReader = new BufferedReader(new FileReader(localCollection))) {
```

```
68                  String nextLine;
69                  StringBuilder result = new StringBuilder();
70                  while ((nextLine = inputStreamReader.readLine()) != null) result.append(
   nextLine);
71                  toServer.writeObject("import " + result.toString());
72                  System.out.println((String) fromServer.readObject());
73              }
74          } catch (FileNotFoundException e) {
75              System.err.println("Файл по указанному пути не найден.");
76          } catch (SecurityException e) {
77              System.err.println("Файл защищён от чтения.");
78          } catch (IOException e) {
79              System.err.println("Что-то не так с файлом.");
80          }
81      }
82
83      private void login(String email) throws IOException, ClassNotFoundException {
84          System.out.println("Введите пароль:");
85          String password = new String(fromKeyboard.readPassword());
86          toServer.writeObject("login " + email + " " + password);
87          System.out.println((String) fromServer.readObject());
88      }
89  }
90
```