

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«МИРЭА - Российский технологический университет»**

**РТУ МИРЭА**

Институт Информационных технологий

Кафедра математического обеспечения и стандартизации

информационных технологий

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3**

по дисциплине «Объектно-ориентированное программирование»

|  |  |
| --- | --- |
| Выполнил студент группы ИКБО-12-17 | А.А. Лисовой |

|  |  |  |
| --- | --- | --- |
| Принял | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (должность, звание, ученая степень) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (И.О. Фамилия) |

|  |  |  |
| --- | --- | --- |
| Работа выполнена | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Подпись студента) | «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2018 г. |

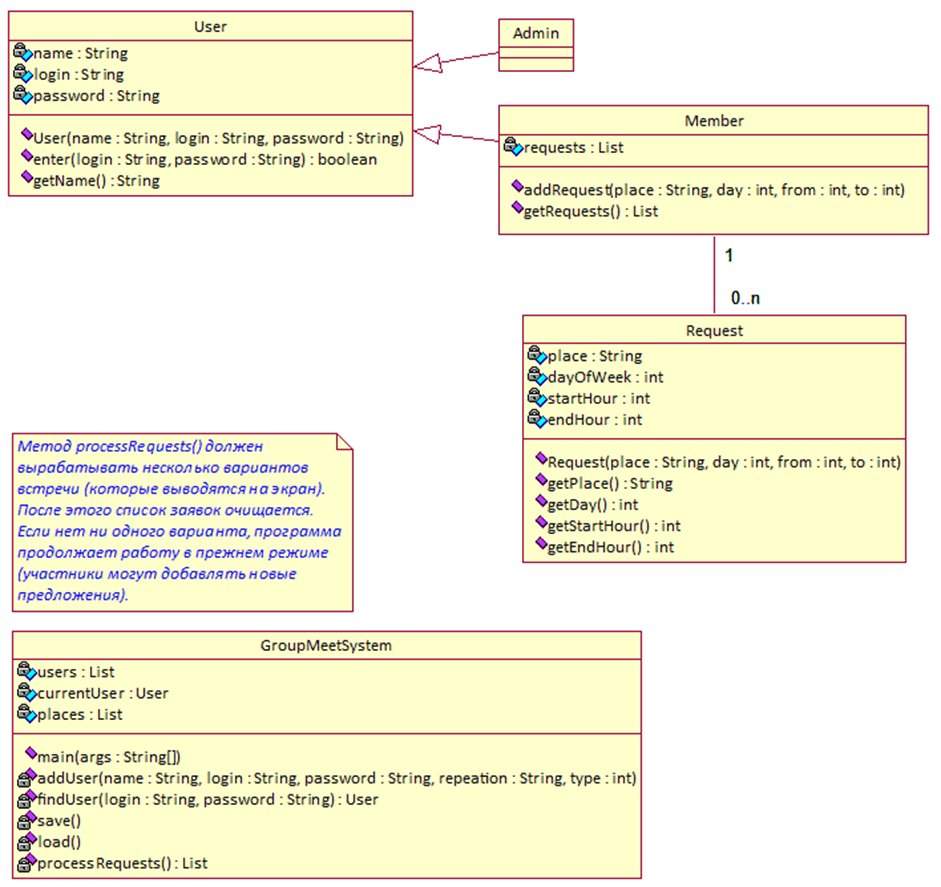
|  |  |  |
| --- | --- | --- |
| Работа зачтена | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Подпись преподавателя) | «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2018 г. |

Москва 2018

**Постановка задачи**

Необходимо разработать программу для координации группы участников проекта. Участникам проекта необходимо назначить время и место встречи для каждого дня недели. Каждый участник регистрируется в системе и указывает свои возможности (список промежутков времени и мест, где он может встретиться). Система подводит итоги, определяя те дни недели, что будут подходить всем участникам проекта.

Возможное время встречи должно задаваться сочетанием день недели + с какого часа + до какого часа. Место встречи выбирается из списка, который заполняется заранее.



**Программная реализация**

|  |
| --- |
| package Lab\_3\_13;  import java.util.ArrayList;  import java.util.List;  import java.util.Scanner;  public class GroupMeetSystem  {  // Поля класса  private static List<String> places; // Список мест для встречи  private static ArrayList<User> users; // Список пользователей  private static User current\_user;  // Основной метод  public static void main (String[] args)  {  int command;  Scanner input = new Scanner(System.in);  places = new ArrayList<String>();  users = new ArrayList<User>();  show\_menu(); // Основное меню программы  do  {  System.out.print("\nВведите команду: ");  while (!input.hasNextInt())  {  System.out.println("Ошибка ввода");  System.out.print("\nВведите команду: ");  input.next();  }  command = input.nextInt();  // Обработка опций  switch (command)  {  case 1:  // Вход в систему  system\_enter();  break;  case 2:  // Регистрация пользователя  register\_user();  break;  case 3:  // Вывести список зарегистрированных пользователей  output\_users();  break;  case 4:  //Опредедлить возможные дни встречи всех членов команды  process\_request();  case 5:  // Повторный вывод меню  show\_menu();  break;  case 6:  // Выход из программы  System.out.println("Выход из программы");  break;  default:  System.out.println("Ошибка ввода");  }  }  while (command != 6);  }  // Метод входа в систему  public static void system\_enter()  {  current\_user = login\_user();  if (current\_user == null)  {  System.out.println("Возвращение в главное меню");  }  else current\_user.use\_menu();  }  // Метод вывода списка зарегистрированных пользователей  public static void output\_users()  {  int size = users.size();  System.out.println("Текущий список пользователей: ");  if (size != 0)  {  for (int index = 0; index < size; index++)  {  System.out.println((index + 1) + ". " +  users.get(index).get\_username());  }  }  else System.out.println("Список пуст");  }  // Метод основного меню программы  public static void show\_menu()  {  System.out.println("Главное меню:");  System.out.println("1. Войти в систему");  System.out.println("2. Зарегистрироваться");  System.out.println("3. Вывести список зарегистрированных  пользователей");  System.out.println("4. Опредедлить возможные дни встречи всех  членов команды");  System.out.println("5. Повторный вывод меню");  System.out.println("6. Выход из программы");  }  // Метод входа в систему  public static User login\_user()  {  Scanner scanner = new Scanner(System.in);  System.out.print("Введите имя пользователя: ");  String current\_username = scanner.nextLine();  System.out.print("Введите пароль пользователя: ");  String current\_password = scanner.nextLine();  User temp\_user = new User();  // Проверка существования пользователя  boolean user\_exists = false;  int size = users.size();  for (int index = 0; index < size; index++)  {  if (users.get(index).get\_username().equals(current\_username))  {  user\_exists = true;  temp\_user = users.get(index);  break;  }  }  if (user\_exists == false)  {  System.out.println("Ошибка ввода имени пользователя");  return null;  }  else  {  // Проверка соответствия пароля пользователя  if (temp\_user.get\_password().equals(current\_password))  {  System.out.println("Вход произведён успешно");  System.out.println();  return temp\_user;  }  else  {  System.out.println("Ошибка ввода пароля пользователя");  return null;  }  }  }  // Метод регистрации пользователя  public static void register\_user()  {  int command;  Scanner input = new Scanner(System.in);  show\_register\_menu(); // Меню регистрации  do  {  System.out.print("\nВведите команду: ");  while (!input.hasNextInt())  {  System.out.println("Ошибка ввода");  System.out.print("\nВведите команду: ");  input.next();  }  command = input.nextInt();  switch (command)  {  case 1:  // Регистрация администратора  register\_admin();  break;  case 2:  // Регистрация фаната  register\_member();  break;  case 3:  // Повторный вывод меню  show\_register\_menu();  break;  case 4:  // Возвращение в главное меню  break;  default:  System.out.println("Ошибка ввода");  }  }  while (command != 4 && command != 1 && command != 2);  System.out.println("Возвращение в главное меню");  }  // Метод вывода меню регистрации  public static void show\_register\_menu()  {  System.out.println("Меню регистрации пользователя:");  System.out.println("1. Администратор");  System.out.println("2. Член команды");  System.out.println("3. Повторный вывод меню");  System.out.println("4. Вернуться в главное меню");  }  // Метод регистрации администратора  public static void register\_admin()  {  System.out.println("Регистрация администратора");  Scanner scanner = new Scanner(System.in);  System.out.print("Введите имя пользователя: ");  String current\_username = scanner.nextLine();  System.out.print("Введите пароль пользователя: ");  String current\_password = scanner.nextLine();  Admin admin = new Admin(current\_username, current\_password); //  Создание нового администратора  // Проверка существования пользователя  boolean user\_exists = false;  int size = users.size();  for (int index = 0; index < size; index++)  {  if (users.get(index).get\_username().equals(current\_username))  {  user\_exists = true;  break;  }  }  if (user\_exists == false)  {  users.add(admin);  System.out.println("Администратор зарегистрирован");  }  else System.out.println("Ошибка: пользователь с таким именем уже  существует");  }  // Метод регистрации члена команды  public static void register\_member()  {  System.out.println("Регистрация члена команды");  Scanner scanner = new Scanner(System.in);  System.out.print("Введите имя пользователя: ");  String current\_username = scanner.nextLine();  System.out.print("Введите пароль пользователя: ");  String current\_password = scanner.nextLine();  Member member = new Member(current\_username, current\_password);  //Создание нового члена команды  // Проверка существования пользователя  boolean user\_exists = false;  int size = users.size();  for (int index = 0; index < size; index++)  {  if (users.get(index).get\_username().equals(current\_username))  {  user\_exists = true;  break;  }  }  if (user\_exists == false)  {  users.add(member);  System.out.println("Член комманды зарегистрирован");  }  else System.out.println("Ошибка: пользователь с таким именем уже  существует");  }  // Метод очистки списка пользователей  public static void users\_clear()  {  System.out.println("Список пользователей очищен");  users.clear();  }  // Метод добавления нового места встречи  public static void add\_place()  {  String place = "";  Scanner input = new Scanner(System.in);  System.out.print("Введите новое место для встреч: ");  do  {  place = input.next();  }  while (place == "");  places.add(place);  System.out.println("Место встречи успешно добавлено");  }  // Метод показа списка мест встречи  public static void show\_places()  {  for (int index = 0; index < places.size(); index++)  {  System.out.println("Место встречи № " + (index + 1) + " - " +  places.get(index));  }  }  // Метод заполнения дня встречи для члена команды  public static void add\_day()  {  int from;  int to;  int day;  int place\_number;  Scanner input = new Scanner(System.in);  // Поиск члена команды  int index = 0;  while (current\_user.get\_username() !=  users.get(index).get\_username() && index < users.size())  {  index++;  }  // Проверка на найденность  if (current\_user.get\_username() == users.get(index).get\_username())  {  System.out.println("Введите день недели, в который  намереваетесь добавить расписание: ");  while (!input.hasNextInt())  {  System.out.println("Ошибка ввода");  System.out.print("Введите день: ");  input.next();  }  day = input.nextInt() - 1;  // Проверказа невыход рамок недели  if (day <= 7 && day >= 0)  {  // Ввод времени начала  System.out.print("Введите время начала возможной встречи:  ");  while (!input.hasNextInt())  {  System.out.println("Ошибка ввода");  System.out.print("Введите время: ");  input.next();  }  from = input.nextInt();  // Ввод времени конца  System.out.print("Введите время конца возможной встречи:  ");  while (!input.hasNextInt())  {  System.out.println("Ошибка ввода");  System.out.print("Введите время: ");  input.next();  }  to = input.nextInt();  // Показ списка возможных мест для встречи  show\_places();  // Выбор места встречи  System.out.print("Введите номер места встречи: ");  while (!input.hasNextInt())  {  System.out.println("Ошибка ввода");  System.out.print("Введите номер: ");  input.next();  }  place\_number = input.nextInt() - 1;  Request request = new Request(places.get(place\_number),  day, from, to); // Добавляемый запрос  ArrayList<Request> current\_member = ((Member)  users.get(index)).get\_member(); // Имеющийся список дней  current\_member.set(day, request);  User update\_user = users.get(index);  ((Member) update\_user).set\_member(current\_member);  users.set(index, update\_user);  System.out.println("День был успешно добавлен");  }  else  {  System.out.println("Введен некорректный номер дня недели");  }  }  else  {  System.out.println("Член комманды не найден");  }  }  // Метод составления списка дней, когда все члены команды смогут  встретиться  public static void process\_request()  {  // Проверка на заполненность  if (users.size() > 2)  {  User current\_member = ((Member)users.get(1)); // Член команды,  с которого начнется поиск  User compared\_member; // Сравниваемый с ним член команды  boolean flag = true; // Логическая переменная  System.out.println("Список возможных встреч по дням");  for (int index = 0; index < 7; index++)  {  // Проверка всех членов команды для текущего дня недели  for (int index\_user = 2; index\_user < users.size();  index\_user++)  {  compared\_member = users.get(index\_user);  if (((Member) current\_member).get\_member().get(index).  get\_start\_hour() != ((Member)  compared\_member).get\_member().get(index)  .get\_start\_hour()  || ((Member) current\_member).get\_member().get(index)  .get\_end\_hour() != ((Member)  compared\_member).get\_member().get(index)  .get\_end\_hour()  || ((Member) current\_member).get\_member().get(index)  .get\_place() != ((Member)  compared\_member).get\_member().get(index)  .get\_place()  || ((Member) current\_member).get\_member().get(index)  .get\_start\_hour() <= 0 || ((Member)  current\_member).get\_member().get(index)  .get\_end\_hour() <= 0  || ((Member) current\_member).get\_member().get(index)  .get\_place() == "" || ((Member)  compared\_member).get\_member().get(index)  .get\_start\_hour() <= 0  || ((Member) compared\_member).get\_member().get(index)  .get\_end\_hour() <= 0 || ((Member)  compared\_member).get\_member().get(index)  .get\_place() == "")  {  flag = false;  }  }  // Если все члены команды совпадают  if (flag)  {  int final\_start\_hour = ((Member)  current\_member).get\_member().get(index)  .get\_start\_hour();  int final\_end\_hour = ((Member)  current\_member).get\_member().get(index)  .get\_end\_hour();  String final\_place = ((Member)  current\_member).get\_member().  get(index).get\_place();  System.out.println("В " + (index + 1) + " день недели  члены команды смогут провести встречу с " +  final\_start\_hour + " по " + final\_end\_hour + " в " +  final\_place);  }  else  {  flag = true;  }  }  }  else  {  System.out.println("Список членов команды не был заполнен \n");  }  }  } |

Листинг 1. Исходный код класса ConcertSystem.

|  |
| --- |
| package Lab\_3\_13;  // Наследуемый класс пользователя  public class User  {  // Имя пользователя  private String username;  // Пароль пользователя  private String password;  // Метод установки значения имени пользователя  public void set\_username(String string)  {  username = string;  }  // Метод установки значения пароля пользователя  public void set\_password(String string)  {  password = string;  }  // Метод получения значения имени пользователя  public String get\_username()  {  return username;  }  // Метод получения значения пароля пользователя  public String get\_password()  {  return password;  }  // Конструктор по-умолчанию  User()  {  }  // Конструктор с параметрами  User(String username, String password)  {  this.username = username;  this.password = password;  }  // Метод меню  public void use\_menu()  {  }  // Метод вывода меню  public void show\_menu()  {  }  } |

Листинг 2. Исходный код класса User.

|  |
| --- |
| package Lab\_3\_13;  import java.util.ArrayList;  import java.util.Scanner;  public class Member extends User  {  // Список дней  private ArrayList<Request> member;  private Request request = new Request();  // Конструктор с параметрами  Member(String username, String password)  {  super(username, password);  member = new ArrayList<Request>(6);  for (int index = 0; index < 7; index++)  member.add(request);  }  // Получение списка дней  public ArrayList<Request> get\_member()  {  return member;  }  // Обновление списка дней  public void set\_member(ArrayList<Request> new\_member)  {  member = new\_member;  }  // Метод меню члена команды  public void use\_menu()  {  int command;  Scanner input = new Scanner(System.in);  show\_menu();  do  {  System.out.print("\nВведите комманду: ");  while (!input.hasNextInt())  {  System.out.println("Ошибка ввода");  System.out.print("\nВведите команду: ");  input.next();  }  command = input.nextInt();  switch (command)  {  case 1:  // Вывести список уже заполненных дней  output\_member();  break;  case 2:  // Добавить расписание для дня  GroupMeetSystem.add\_day();  break;  case 3:  show\_menu();  break;  case 4:  // Вернуться в главное меню  System.out.println("Возвращение в главное меню");  break;  default:  System.out.println("Ошибка ввода");  }  }  while (command != 4);  }  // Метод вывода меню члена команды  public void show\_menu()  {  System.out.println("Меню члена команды " + get\_username() + " ");  System.out.println("1. Показать список дней с расписанием");  System.out.println("2. Добавить расписание для нового дня");  System.out.println("3. Повторный вывод меню");  System.out.println("4. Вернуться в главное меню");  }  // Метод вывода заполненных дней  public void output\_member()  {  System.out.println("Список уже заполненных дней");  for (int index = 0; index < member.size(); index++)  {  if (member.get(index).get\_start\_hour() > 0 &&  member.get(index).get\_end\_hour() > 0)  {  System.out.println("День №" + (index + 1) + " время: с " +  member.get(index).get\_start\_hour() + " по " +  member.get(index).get\_end\_hour() + " в месте " +  member.get(index).get\_place());  }  }  }  } |

Листинг 3. Исходный код класса Member.

|  |
| --- |
| package Lab\_3\_13;  import java.util.Scanner;  public class Admin extends User  {  // Конструктор с параметрами  Admin(String username, String password)  {  super(username, password);  }  // Метод меню администратора  public void use\_menu()  {  int command;  Scanner input = new Scanner(System.in);  show\_menu();  do  {  System.out.print("\nВведите команду: ");  while (!input.hasNextInt())  {  System.out.println("Ошибка ввода");  System.out.print("\nВведите команду: ");  input.next();  }  command = input.nextInt();  switch (command)  {  case 1:  // Очистка списка пользователей  GroupMeetSystem.users\_clear();  break;  case 2:  // Добавление нового места встречи  GroupMeetSystem.add\_place();  break;  case 3:  // Вывод уже добавленных мест встречи  GroupMeetSystem.show\_places();  break;  case 4:  // Повторный вывод меню  show\_menu();  break;  case 5:  // Вернуться в главное меню  System.out.println("Возвращение в главное меню");  break;  default:  System.out.println("Ошибка ввода");  }  }  while (command != 5);  }  // Метод вывода меню администратора  public void show\_menu()  {  System.out.println("Меню действий администратора");  System.out.println("1 - Очистить базу данных пользователей");  System.out.println("2 - Добавить новое место встречи");  System.out.println("3 - Показать список мест встречи");  System.out.println("4 - Повторный вывод меню");  System.out.println("5 - Возврат в главное меню");  }  } |

Листинг 4. Исходный код класса Admin.

|  |
| --- |
| package Lab\_3\_13;  // Класс запроса  public class Request  {  private String place; // Место встречи  private int day; // День недели  private int start\_hour; // Время время начала  private int end\_hour; // Время окончания  // Констурктор по умолчанию  public Request()  {  this.place = "";  this.day = 0;  this.start\_hour = 0;  this.end\_hour = 0;  }  // Конструктор с параметрами  public Request(String place, int day, int from, int to)  {  this.place = place;  this.day = day;  this.start\_hour = from;  this.end\_hour = to;  }  // Вернуть место встречи  public String get\_place()  {  return place;  }  // Вернуть день недели встречи  public int get\_day()  {  return day;  }  // Вернуть время начала встречи  public int get\_start\_hour()  {  return start\_hour;  }  // Вернуть время завершения встречи  public int get\_end\_hour()  {  return end\_hour;  }  } |

Листинг 5. Исходный код класса Request.

**Тестирование**

Главное меню:

1. Войти в систему

2. Зарегистрироваться

3. Вывести список зарегистрированных пользователей

4. Опредедлить возможные дни встречи всех членов команды

5. Повторный вывод меню

6. Выход из программы

Введите комманду: g

Ошибка ввода

Введите команду: 4222

Ошибка ввода

Введите комманду: -23

Ошибка ввода

Введите комманду: 2

Меню регистрации пользователя:

1. Администратор

2. Член команды

3. Повторный вывод меню

4. Вернуться в главное меню

Введите команду: 1

Регистрация администратора

Введите имя пользователя: Admin

Введите пароль пользователя: qwerty

Администратор зарегистрирован

Возвращение в главное меню

Введите комманду: 5

Главное меню:

1. Войти в систему

2. Зарегистрироваться

3. Вывести список зарегистрированных пользователей

4. Опредедлить возможные дни встречи всех членов команды

5. Повторный вывод меню

6. Выход из программы

Введите комманду: 1

Введите имя пользователя: Admin

Введите пароль пользователя: qwerty

Вход произведён успешно

Меню действий администратора

1 - Очистить базу данных пользователей

2 - Добавить новое место встречи

3 - Показать список мест встречи

4 - Повторный вывод меню

5 - Возврат в главное меню

Введите команду: 2

Введите новое место для встреч: Парк

Место встречи успешно добавлено

Введите команду: 2

Введите новое место для встреч: Кафе

Место встречи успешно добавлено

Введите команду: 2

Введите новое место для встреч: Офис

Место встречи успешно добавлено

Введите команду: 5

Возвращение в главное меню

Введите комманду: 2

Меню регистрации пользователя:

1. Администратор

2. Член команды

3. Повторный вывод меню

4. Вернуться в главное меню

Введите команду: 2

Регистрация члена команды

Введите имя пользователя: Иван

Введите пароль пользователя: 123

Член комманды зарегистрирован

Возвращение в главное меню

Введите комманду: 1

Введите имя пользователя: Иван

Введите пароль пользователя: 123

Вход произведён успешно

Меню члена команды Иван

1. Показать список дней с расписанием

2. Добавить расписание для нового дня

3. Повторный вывод меню

4. Вернуться в главное меню

Введите комманду: 2

Введите день недели, в который намереваетесь добавить расписание: 2

Введите время начала возможной встречи: 12

Введите время конца возможной встречи: 16

Место встречи № 1 - Парк

Место встречи № 2 - Кафе

Место встречи № 3 - Офис

Введите номер места встречи: 2

День был успешно добавлен

Введите комманду: 1

Список уже заполненных дней

День №2 время: с 12 по 16 в месте Кафе

Введите комманду: 2

Введите день недели, в который намереваетесь добавить расписание: 5

Введите время начала возможной встречи: 15

Введите время конца возможной встречи: 18

Место встречи № 1 - Парк

Место встречи № 2 - Кафе

Место встречи № 3 - Офис

Введите номер места встречи: 3

День был успешно добавлен

Введите комманду: 4

Возвращение в главное меню

Введите комманду: 2

Меню регистрации пользователя:

1. Администратор

2. Член команды

3. Повторный вывод меню

4. Вернуться в главное меню

Введите команду: 2

Регистрация члена команды

Введите имя пользователя: Саша

Введите пароль пользователя: 456

Член комманды зарегистрирован

Возвращение в главное меню

Введите комманду: 1

Введите имя пользователя: Саша

Введите пароль пользователя: 456

Вход произведён успешно

Меню члена команды Саша

1. Показать список дней с расписанием

2. Добавить расписание для нового дня

3. Повторный вывод меню

4. Вернуться в главное меню

Введите комманду: 2

Введите день недели, в который намереваетесь добавить расписание: 2

Введите время начала возможной встречи: 12

Введите время конца возможной встречи: 16

Место встречи № 1 - Парк

Место встречи № 2 - Кафе

Место встречи № 3 - Офис

Введите номер места встречи: 2

День был успешно добавлен

Введите комманду: 2

Введите день недели, в который намереваетесь добавить расписание: 3

Введите время начала возможной встречи: 8

Введите время конца возможной встречи: 19

Место встречи № 1 - Парк

Место встречи № 2 - Кафе

Место встречи № 3 - Офис

Введите номер места встречи: 1

День был успешно добавлен

Введите комманду: 2

Введите день недели, в который намереваетесь добавить расписание: 5

Введите время начала возможной встречи: 15

Введите время конца возможной встречи: 18

Место встречи № 1 - Парк

Место встречи № 2 - Кафе

Место встречи № 3 - Офис

Введите номер места встречи: 3

День был успешно добавлен

Введите комманду: 3

Меню члена команды Саша

1. Показать список дней с расписанием

2. Добавить расписание для нового дня

3. Повторный вывод меню

4. Вернуться в главное меню

Введите комманду: 1

Список уже заполненных дней

День №2 время: с 12 по 16 в месте Кафе

День №3 время: с 8 по 19 в месте Парк

День №5 время: с 15 по 18 в месте ОФис

Введите комманду: 4

Возвращение в главное меню

Введите комманду: 5

Главное меню:

1. Войти в систему

2. Зарегистрироваться

3. Вывести список зарегистрированных пользователей

4. Опредедлить возможные дни встречи всех членов команды

5. Повторный вывод меню

6. Выход из программы

Введите комманду: 3

Текущий список пользователей:

1. Admin

2. Иван

3. Саша

Введите комманду: 4

Список возможных встреч по дням

В 2 день недели члены команды смогут провести встречу с 14 по 16 в Кафе

В 5 день недели члены команды смогут провести встречу с 15 по 18 в Офис

Главное меню:

1. Войти в систему

2. Зарегистрироваться

3. Вывести список зарегистрированных пользователей

4. Опредедлить возможные дни встречи всех членов команды

5. Повторный вывод меню

6. Выход из программы

Введите команду: прпк

Ошибка ввода

Введите команду: 344

Ошибка ввода

Введите команду: 6

Выход из программы

Введите команду: 1

Введите имя пользователя: Admin

Введите пароль пользователя: qwerty

Вход произведён успешно

Меню действий администратора

1 - Очистить базу данных пользователей

2 - Добавить новое место встречи

3 - Показать список мест встречи

4 - Повторный вывод меню

5 - Возврат в главное меню

Введите команду: 1

Список пользователей очищен

Введите команду: 5

Возвращение в главное меню

Введите команду: 5

Главное меню:

1. Войти в систему

2. Зарегистрироваться

3. Вывести список зарегистрированных пользователей

4. Опредедлить возможные дни встречи всех членов команды

5. Повторный вывод меню

6. Выход из программы

Введите команду: 6

Выход из программы

**Выводы**

В ходе выполнения лабораторной работы были получены практические навыки по работе с взаимодействующими пользовательскими классами и библиотекой STL. Для удобства работы с программой было разработано консольное меню.