Министерство образования Калининградской области

государственное бюджетное учреждение Калининградской области

профессиональная образовательная организация

«Колледж информационных технологий и строительства»

(ГБУ КО ПОО «КИТиС»)

**Отчет по учебной практике**

УП.01 Разработка модулей программного обеспечения для компьютерных систем

по ПМ.01 Разработка модулей программного обеспечения для компьютерных систем

Специальность 09.02.07 «Информационные системы и программирование»

Сроки прохождения практики:

с «09» сентября 2024 г. по «09» ноября 2024 г.

Место практики ГБУ КО ПОО «КИТиС»

|  |  |
| --- | --- |
| Выполнил: | студент 4 курса,  группы ИСп 21-2к  Толстов Владислав Андреевич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
| Проверила: | Большакова-Стрекалова Анна Викторовна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |

Калининград, 2024

Оглавление

[Введение 3](#_Toc182462010)

[1 Разработка программных модулей 4](#_Toc182462011)

[1.1 Задача 1 4](#_Toc182462012)

[1.2 Задача 2 7](#_Toc182462013)

[1.3 Задача 3 13](#_Toc182462014)

[1.4 Задача 4 18](#_Toc182462015)

[1.5 Задача 5 23](#_Toc182462016)

[1.6 Задача 6 25](#_Toc182462017)

[1.7 Задача 7 34](#_Toc182462018)

[1.8 Задача 8 37](#_Toc182462019)

[1.9 Задача 9 42](#_Toc182462020)

[1.10 Задача 10 55](#_Toc182462021)

[2 Разработка АИС 58](#_Toc182462022)

[2.1 Техническое задание 59](#_Toc182462023)

[2.2 Краткая спецификация разрабатываемой системы 61](#_Toc182462024)

[2.3 Тестирование АИС 77](#_Toc182462025)

[2.4 Оценка программного продукта 79](#_Toc182462026)

[2.5 Руководство пользователя 82](#_Toc182462027)

[2.6 Руководство программиста 83](#_Toc182462028)

[3 Программирование 1С 85](#_Toc182462029)

[4 Разработка мобильных приложений 87](#_Toc182462030)

[5 Разработка сайта 89](#_Toc182462031)

[Заключение 90](#_Toc182462032)

[Список используемой литературы 91](#_Toc182462033)

Введение

Настоящий отчёт представляет результаты прохождения учебной практики по программированию, проведённой в рамках образовательной программы. Целью практики было закрепление теоретических знаний и приобретение практических навыков в различных областях программирования. В ходе практики были изучены и применены на практике пять ключевых дисциплин:

* Разработка программных модулей
* Разработка мобильных приложений
* Поддержка и тестирование программных модулей
* Программирование на платформе 1С
* Системное программирование

Практическая часть также включала создание собственного веб-сайта. Разработка сайта предоставила возможность продемонстрировать выполненные задания, а также приобрести опыт веб-программирования и дизайна. Сайт служит платформой для презентации проектов и результатов, достигнутых в процессе практики.

Цели практики:

1. Расширение знаний и совершенствование навыков: углубить понимание теоретических основ и улучшить практические умения в каждой из изучаемых дисциплин;
2. Выполнение поставленных задач: решить комплекс практических заданий для оценки уровня усвоения материала и способности применять знания на практике;
3. Создание собственного веб-сайта: разработать и запустить веб-сайт для представления результатов практики и демонстрации профессиональных навыков.

1 Разработка программных модулей

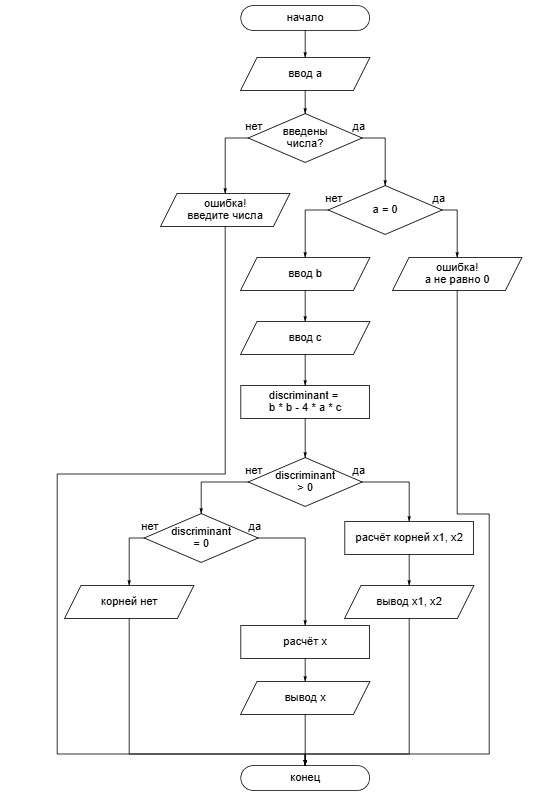
1.1 Задача 1

Найти корни квадратного уравнения. Параметры уравнения передавать с командной строкой.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Название переменной** |  |  | | --- | |  | | |  | | --- | | **Тип данных** |  |  | | --- | |  | | Параметр |
| scanner | Scanner | Вспомогательная |
| a | Double | Входная |
| b | Double | Входная |
| c | Double | Входная |
| discriminant | Double | Вспомогательная |
| X1 | Double | Выходная |
| X2 | Double | Выходная |
| x | Double | Выходная |
| e | NumberFormatException | Вспомогательная |

Таблица 1 – Список идентификаторов задачи 1

Блок-схема



Код

package org.example;  
  
import java.util.Scanner;  
  
public class QuadraticEquationSolver {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.println("Введите коэффициенты a, b и c для квадратного уравнения (ax^2 + bx + c = 0):");  
  
 try {  
 System.*out*.print("Введите коэффициент a: ");  
 double a = scanner.nextDouble();  
  
 if (a == 0) {  
 System.*out*.println("Коэффициент 'a' не может быть равен нулю.");  
 return;  
 }  
  
 System.*out*.print("Введите коэффициент b: ");  
 double b = scanner.nextDouble();  
  
 System.*out*.print("Введите коэффициент c: ");  
 double c = scanner.nextDouble();  
  
 double discriminant = b \* b - 4 \* a \* c;  
  
 if (discriminant > 0) {  
 double x1 = (-b + Math.*sqrt*(discriminant)) / (2 \* a);  
 double x2 = (-b - Math.*sqrt*(discriminant)) / (2 \* a);  
 System.*out*.println("Уравнение имеет два действительных корня:");  
 System.*out*.println("x1 = " + x1);  
 System.*out*.println("x2 = " + x2);  
 } else if (discriminant == 0) {  
 double x = -b / (2 \* a);  
 System.*out*.println("Уравнение имеет один действительный корень:");  
 System.*out*.println("x = " + x);  
 } else {  
 System.*out*.println("корней нет");  
 }  
 } catch (NumberFormatException e) {  
 System.*out*.println("Ошибка ввода. Пожалуйста, введите числовые значения.");  
 } finally {  
 scanner.close();  
 }  
 }  
}

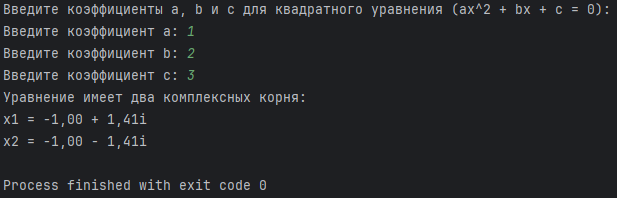


Рисунок 1 – Результат выполнения программы 1

1.2 Задача 2

Создать классы, спецификации которых приведены ниже. Определить конструкторы и методы setТип(), getТип(), toString(). Определить дополнительно методы в классе, создающем массив объектов. Задать критерий выбора данных и вывести эти данные на консоль. В каждом классе, обладающем информацией, должно быть объявлено несколько конструкторов.

Bus: Фамилия и инициалы водителя, Номер автобуса, Номер маршрута, Марка, Год начала эксплуатации, Пробег. Создать массив объектов. Вывести: a) список автобусов для заданного номера маршрута; b) список автобусов, которые эксплуатируются больше заданного срока; c) список автобусов, пробег у которых больше заданного расстояния.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Название переменной** |  |  | | --- | |  | | |  | | --- | | **Тип данных** |  |  | | --- | |  | | Параметр |
| DriverName | String | Входная |
| bus\_number | int | Входная |

Таблица 2 - Список идентификаторов задачи 2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Название переменной** |  |  | | --- | |  | | |  | | --- | | **Тип данных** |  |  | | --- | |  | | Параметр |
| route\_number | int | Входная |
| brand | String | Входная |
| |  | | --- | | start\_using\_year |  |  | | --- | |  | | int | Входная |
| mileage | int | Входная |
| buses | List<Bus> | Вспомогательная |
| routeNumber | int | Входная |
| yearsInUse | int | Входная |
| minMileage | int | Входная |
| bus | Bus | Вспомогательная |
| currentYear | int | Вспомогательная |

Таблица 2 - Список идентификаторов задачи 2. Продолжение

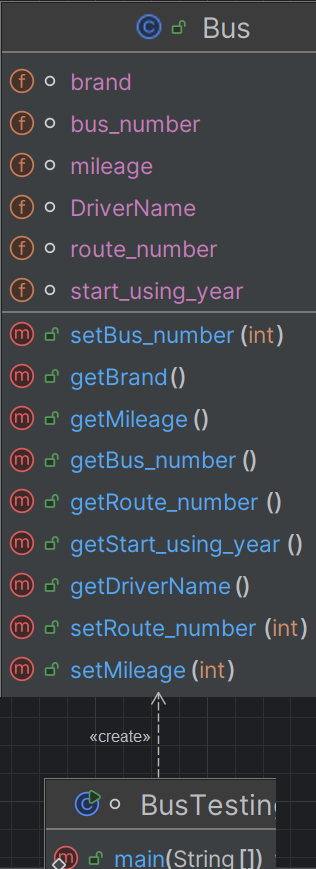


Рисунок 2 – Диаграмма классов задачи 2

Код

package org.example;  
  
import java.util.ArrayList;  
import java.util.Calendar;  
import java.util.List;  
  
public class Bus {  
 String DriverName;  
 int bus\_number;  
 int route\_number;  
 String brand;  
 int start\_using\_year;  
 int mileage;  
  
 public Bus(String driverName, int bus\_number, int route\_number, String brand, int start\_using\_year, int mileage) {  
 DriverName = driverName;  
 this.bus\_number = bus\_number;  
 this.route\_number = route\_number;  
 this.brand = brand;  
 this.start\_using\_year = start\_using\_year;  
 this.mileage = mileage;  
 }  
  
 @Override  
 public String toString() {  
 return "bus{" +  
 "DriverName='" + DriverName + '\'' +  
 ", bus\_number=" + bus\_number +  
 ", route\_number=" + route\_number +  
 ", brand=" + brand +  
 ", start\_using\_year=" + start\_using\_year +  
 ", mileage=" + mileage +  
 '}';  
 }  
  
 public String getDriverName() {  
 return DriverName;  
 }  
  
 public void setDriverName(String driverName) {  
 DriverName = driverName;  
 }  
  
 public int getBus\_number() {  
 return bus\_number;  
 }  
  
 public void setBus\_number(int bus\_number) {  
 this.bus\_number = bus\_number;  
 }  
  
 public int getRoute\_number() {  
 return route\_number;  
 }  
  
 public void setRoute\_number(int route\_number) {  
 this.route\_number = route\_number;  
 }  
  
 public String getBrand() {  
 return brand;  
 }  
  
 public void setBrand(String brand) {  
 this.brand = brand;  
 }  
  
 public int getStart\_using\_year() {  
 return start\_using\_year;  
 }  
  
 public void setStart\_using\_year(int start\_using\_year) {  
 this.start\_using\_year = start\_using\_year;  
 }  
  
 public int getMileage() {  
 return mileage;  
 }  
  
 public void setMileage(int mileage) {  
 this.mileage = mileage;  
 }  
}  
class BusTesting {  
 public static void main(String[] args) {  
 List<Bus> buses = new ArrayList<>();  
 buses.add(new Bus("Иванов И.И.", 101, 5, "Mercedes", 2015, 200000));  
 buses.add(new Bus("Петров П.П.", 102, 10, "Volvo", 2018, 150000));  
 buses.add(new Bus("Сидоров С.С.", 103, 5, "MAN", 2010, 300000));  
 buses.add(new Bus("Кузнецов К.К.", 104, 7, "Scania", 2012, 250000));  
 buses.add(new Bus("Смирнов С.С.", 105, 10, "Iveco", 2015, 180000));  
  
 int routeNumber = 5;  
 int yearsInUse = 5;  
 int minMileage = 200000;  
  
 // a) Список автобусов для заданного номера маршрута  
 System.*out*.println("Автобусы для маршрута " + routeNumber + ":");  
 for (Bus bus : buses) {  
 if (bus.getRoute\_number() == routeNumber) {  
 System.*out*.println(bus);  
 }  
 }  
  
 // b) Список автобусов, которые эксплуатируются больше заданного срока  
 int currentYear = Calendar.*getInstance*().get(Calendar.*YEAR*);  
 System.*out*.println("\nАвтобусы, которые эксплуатируются больше " + yearsInUse + " лет:");  
 for (Bus bus : buses) {  
 if (currentYear - bus.getStart\_using\_year() > yearsInUse) {  
 System.*out*.println(bus);  
 }  
 }  
  
 // c) Список автобусов, пробег у которых больше заданного расстояния  
 System.*out*.println("\nАвтобусы, пробег которых больше " + minMileage + " км:");  
 for (Bus bus : buses) {  
 if (bus.getMileage() > minMileage) {  
 System.*out*.println(bus);  
 }  
 }  
 }  
}

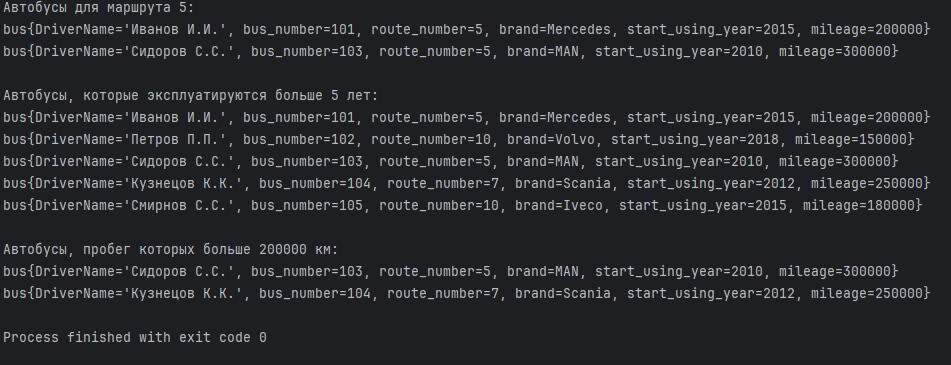


Рисунок 3 – Результат выполнения программы 2

1.3 Задача 3

Создать приложение, удовлетворяющее требованиям, приведенным в задании. Наследование применять только в тех заданиях, в которых это логически обосновано. Аргументировать принадлежность классу каждого создаваемого метода и корректно переопределить для каждого класса методы equals(), hashCode(), toString().

Создать объект класса Хищник, используя классы Когти, Зубы. Методы: рычать, бежать, спать, добывать пищу

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Название переменной** |  |  | | --- | |  | | |  | | --- | | **Тип данных** |  |  | | --- | |  | | Параметр |
| sharpness | int | Входная |
| claws | Claws | Вспомогательная |
| |  | | --- | | size |  |  | | --- | |  | | int | Входная |
| teeth | Teeth | Вспомогательная |
| predator | Predator | Вспомогательная |
| result | int | Вспомогательная |

Таблица 3 – Список идентификаторов задачи 3

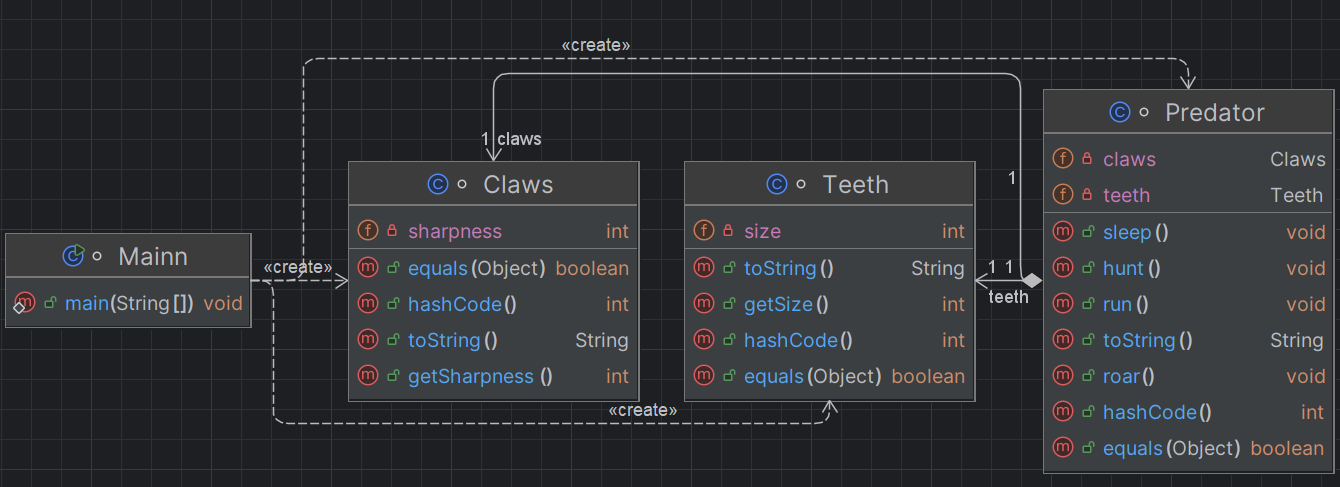


Рисунок 4 – Диаграмма классов задачи 3

Код

package org.example;  
  
// Класс Claws (Когти)  
class Claws {  
 private int sharpness; // Острота когтей  
  
 public Claws(int sharpness) {  
 this.sharpness = sharpness;  
 }  
  
 public int getSharpness() {  
 return sharpness;  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 if (this == obj) return true;  
 if (obj == null || getClass() != obj.getClass()) return false;  
 Claws claws = (Claws) obj;  
 return sharpness == claws.sharpness;  
 }  
  
 @Override  
 public int hashCode() {  
 return Integer.*hashCode*(sharpness);  
 }  
  
 @Override  
 public String toString() {  
 return "Claws{sharpness=" + sharpness + "}";  
 }  
}  
  
// Класс Teeth (Зубы)  
class Teeth {  
 private int size; // Размер зубов  
  
 public Teeth(int size) {  
 this.size = size;  
 }  
  
 public int getSize() {  
 return size;  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 if (this == obj) return true;  
 if (obj == null || getClass() != obj.getClass()) return false;  
 Teeth teeth = (Teeth) obj;  
 return size == teeth.size;  
 }  
  
 @Override  
 public int hashCode() {  
 return Integer.*hashCode*(size);  
 }  
  
 @Override  
 public String toString() {  
 return "Teeth{size=" + size + "}";  
 }  
}  
  
// Класс Predator (Хищник)  
class Predator {  
 private Claws claws;  
 private Teeth teeth;  
  
 public Predator(Claws claws, Teeth teeth) {  
 this.claws = claws;  
 this.teeth = teeth;  
 }  
  
 // Метод рычать  
 public void roar() {  
 System.*out*.println("Хищник рычит!");  
 }  
  
 // Метод бежать  
 public void run() {  
 System.*out*.println("Хищник бежит!");  
 }  
  
 // Метод спать  
 public void sleep() {  
 System.*out*.println("Хищник спит.");  
 }  
  
 // Метод добывать пищу  
 public void hunt() {  
 System.*out*.println("Хищник добывает пищу!");  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 if (this == obj) return true;  
 if (obj == null || getClass() != obj.getClass()) return false;  
 Predator predator = (Predator) obj;  
 return claws.equals(predator.claws) && teeth.equals(predator.teeth);  
 }  
  
 @Override  
 public int hashCode() {  
 int result = claws.hashCode();  
 result = 31 \* result + teeth.hashCode();  
 return result;  
 }  
  
 @Override  
 public String toString() {  
 return "Predator{" +  
 "claws=" + claws +  
 ", teeth=" + teeth +  
 '}';  
 }  
}  
  
// Класс для тестирования  
class Mainn {  
 public static void main(String[] args) {  
 Claws claws = new Claws(10);   
 Teeth teeth = new Teeth(15);   
  
 Predator predator = new Predator(claws, teeth);  
  
 predator.roar();  
 predator.run();  
 predator.sleep();  
 predator.hunt();  
  
 System.*out*.println(predator);  
 }  
}

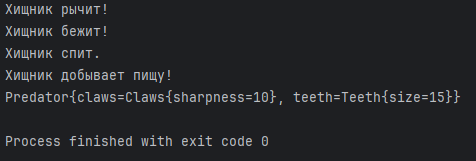


Рисунок 5 – Результат выполнения программы 3

1.4 Задача 4

Создать класс Зачетная Книжка с внутренним классом, с помощью объектов которого можно хранить информацию о сессиях, зачетах, экзаменах

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Название переменной** |  |  | | --- | |  | | |  | | --- | | **Тип данных** |  |  | | --- | |  | | Параметр |
| studentName | String | Входная |
| studentID | String | Входная |
| |  | | --- | | sessions |  |  | | --- | |  | | |  | | --- | | List<Session> |  |  | | --- | |  | | Вспомогательная |
| sessionNumber | int | Входная |
| newSession | Session | Вспомогательная |
| exams | |  | | --- | | List<Exam> |  |  | | --- | |  | | Вспомогательная |
| subject | String | Входная |
| type | String | Входная |
| grade | int | Входная |
| zachetka | Zachetka | Вспомогательная |
| firstSession | |  | | --- | | Zachetka.Session |  |  | | --- | |  | | Вспомогательная |

Таблица 4 – Список идентификаторов задачи 4

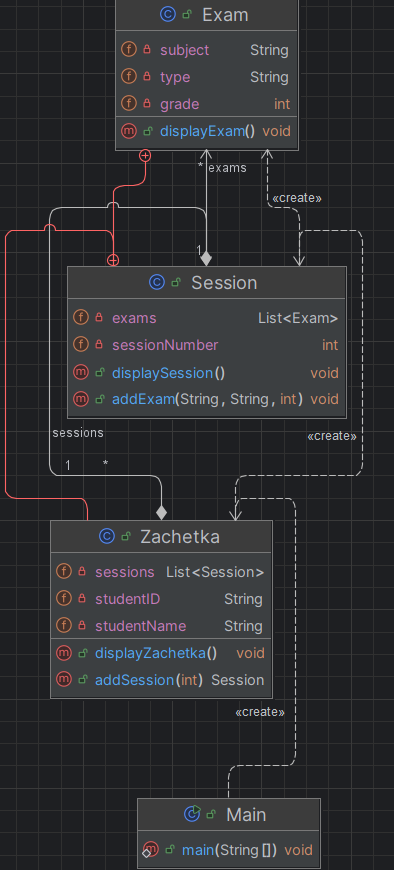
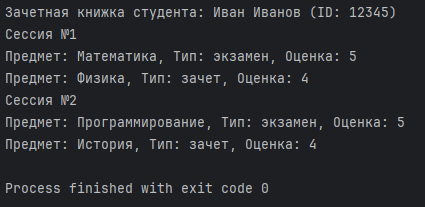


Рисунок 6 – Диаграмма классов задачи 4

Код

package org.example;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class Zachetka {  
 private String studentName;  
 private String studentID;  
 private List<Session> sessions;  
  
 public Zachetka(String studentName, String studentID) {  
 this.studentName = studentName;  
 this.studentID = studentID;  
 this.sessions = new ArrayList<>();  
 }  
  
 // Метод для добавления новой сессии  
 public Session addSession(int sessionNumber) {  
 Session newSession = new Session(sessionNumber);  
 sessions.add(newSession);  
 return newSession;  
 }  
  
 // Метод для отображения всей информации о зачётной книжке  
 public void displayZachetka() {  
 System.*out*.println("Зачетная книжка студента: " + studentName + " (ID: " + studentID + ")");  
 for (Session session : sessions) {  
 session.displaySession();  
 }  
 }  
  
 // Внутренний класс Session, представляющий отдельную сессию  
 public class Session {  
 private int sessionNumber;  
 private List<Exam> exams;  
  
 public Session(int sessionNumber) {  
 this.sessionNumber = sessionNumber;  
 this.exams = new ArrayList<>();  
 }  
  
 // Метод для добавления экзамена  
 public void addExam(String subject, String type, int grade) {  
 exams.add(new Exam(subject, type, grade));  
 }  
  
 // Отображение информации о сессии  
 public void displaySession() {  
 System.*out*.println("Сессия №" + sessionNumber);  
 for (Exam exam : exams) {  
 exam.displayExam();  
 }  
 }  
  
 // Внутренний класс Exam, представляющий отдельный экзамен  
 public class Exam {  
 private String subject;  
 private String type; // Например, "зачет" или "экзамен"  
 private int grade;  
  
 public Exam(String subject, String type, int grade) {  
 this.subject = subject;  
 this.type = type;  
 this.grade = grade;  
 }  
  
 // Отображение информации об экзамене  
 public void displayExam() {  
 System.*out*.println("Предмет: " + subject + ", Тип: " + type + ", Оценка: " + grade);  
 }  
 }  
 }  
}

public class Main {  
 public static void main(String[] args) {  
 // Создаем объект зачётной книжки  
 Zachetka zachetka = new Zachetka("Иван Иванов", "12345");  
  
 // Добавляем первую сессию и экзамены к ней  
 Zachetka.Session firstSession = zachetka.addSession(1);  
 firstSession.addExam("Математика", "экзамен", 5);  
 firstSession.addExam("Физика", "зачет", 4);  
  
 // Добавляем вторую сессию и экзамены к ней  
 Zachetka.Session secondSession = zachetka.addSession(2);  
 secondSession.addExam("Программирование", "экзамен", 5);  
 secondSession.addExam("История", "зачет", 4);  
  
 // Выводим информацию о зачётной книжке  
 zachetka.displayZachetka();  
 }  
}

  
Рисунок 7 – Результат выполнения программы 4

1.5 Задача 5

Все слова текста встречаются четное количество раз, за исключением одного. Определить это слово. При сравнении слов регистр не учитывать.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Название переменной** |  |  | | --- | |  | | |  | | --- | | **Тип данных** |  |  | | --- | |  | | Параметр |
| text | String | Входная |
| words | |  | | --- | | List<String> |  |  | | --- | |  | | Входная |
| |  | | --- | | wordCount |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Map<String, Integer> |  |  | | --- | |  | |  |  | | --- | |  | | Вспомогательная |
| word | String | Вспомогательная |
| entry | |  |  |  | | --- | --- | --- | | |  | | --- | | Map.Entry<String, Integer> |  |  | | --- | |  | |  |  | | --- | |  | | Вспомогательная |

Таблица 5 – Список идентификаторов задачи 5

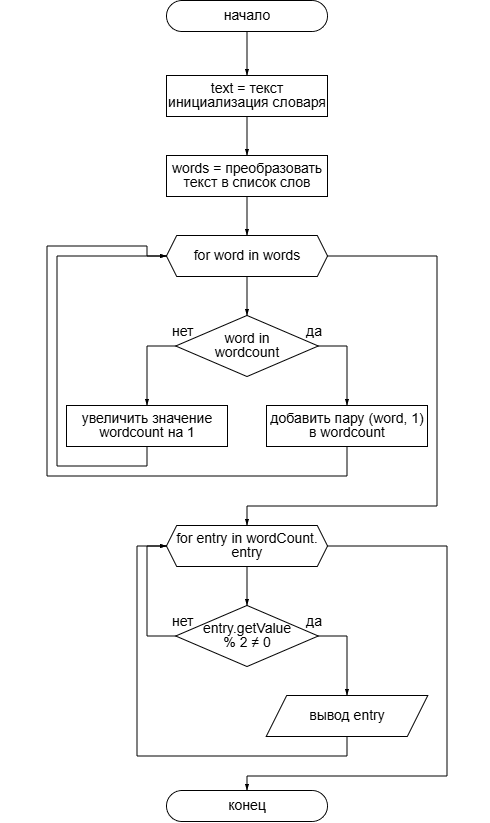


Рисунок 8 – Блок схема программы 5

Код

package org.example;  
  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
public class Nechet {  
 public static void main(String[] args) {  
 String text = "одно одно Да да НЕТ нет четыре четыре четыре кайф кайф";  
  
 // Преобразуем текст в список слов  
 List<String> words = List.*of*(text.toLowerCase().split(" "));  
  
 // Считаем количество каждого слова  
 Map<String, Integer> wordCount = new HashMap<>();  
 for (String word : words) {  
 wordCount.put(word, wordCount.getOrDefault(word, 0) + 1);  
 }  
  
 // Выводим слова, которые встречаются нечетное количество раз  
 for (Map.Entry<String, Integer> entry : wordCount.entrySet()) {  
 if (entry.getValue() % 2 != 0) {  
 System.*out*.println(entry.getKey() + " - " + entry.getValue());  
 }  
 }  
 }  
}

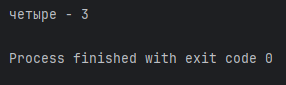


Рисунок 9 – Результат выполнения программы 5

1.6 Задача 6

Выполнить задания на основе задачи № 3, контролируя состояние потоков ввода/вывода. При возникновении ошибок, связанных с корректностью выполнения математических операций, генерировать и обрабатывать исключительные ситуации. Предусмотреть обработку исключений, возникающих при нехватке памяти, отсутствии требуемой записи (объекта) в файле, недопустимом значении поля и т. д.

Выполнить задания из задачи № 3, реализуя собственные обработчики исключений и исключения ввода/вывода.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Название переменной** |  |  | | --- | |  | | |  | | --- | | **Тип данных** |  |  | | --- | |  | | Параметр |
| sharpness | int | Входная |
| claws | Claws | Вспомогательная |
| |  | | --- | | size |  |  | | --- | |  | | int | Входная |
| teeth | Teeth | Вспомогательная |
| predator | Predator | Вспомогательная |
| result | int | Вспомогательная |

Таблица 6 – Список идентификаторов задачи 6

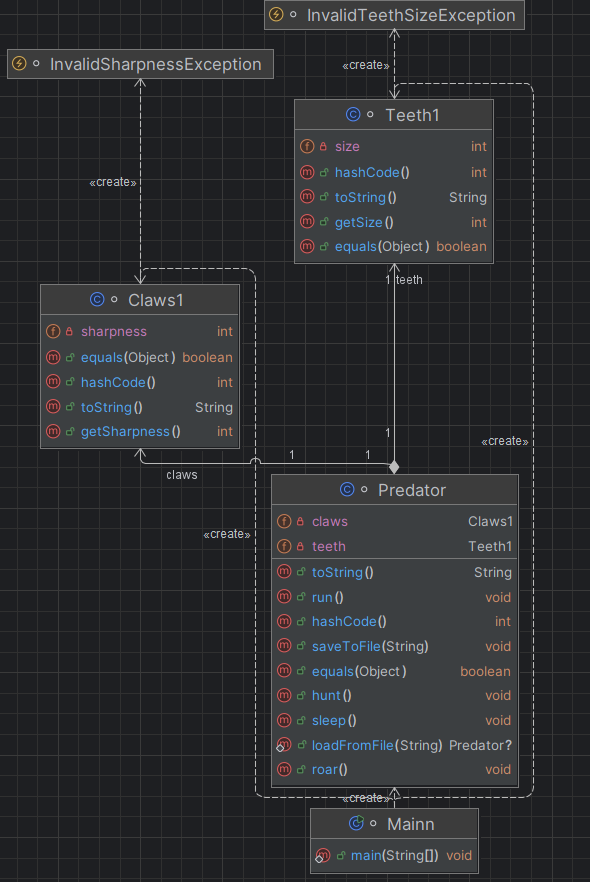


Рисунок 10 – Диаграмма классов задачи 6

Код

package org.example;  
  
import java.io.\*;  
import java.util.Objects;  
  
// Кастомное исключение для недопустимой остроты когтей  
class InvalidSharpnessException extends Exception {  
 public InvalidSharpnessException(String message) {  
 super(message);  
 }  
}  
  
// Кастомное исключение для недопустимого размера зубов  
class InvalidTeethSizeException extends Exception {  
 public InvalidTeethSizeException(String message) {  
 super(message);  
 }  
}  
  
// Класс Claws (Когти)  
class Claws {  
 private int sharpness; // Острота когтей  
  
 public Claws(int sharpness) throws InvalidSharpnessException {  
 if (sharpness <= 0) {  
 throw new InvalidSharpnessException(«Острота когтей должна быть положительным числом.»);  
 }  
 this.sharpness = sharpness;  
 }  
  
 public int getSharpness() {  
 return sharpness;  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 if (this == obj) return true;  
 if (obj == null || getClass() != obj.getClass()) return false;  
 Claws1 claws = (Claws1) obj;  
 return sharpness == claws.getSharpness();  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(sharpness);  
 }  
  
 @Override  
 public String toString() {  
 return «Claws{sharpness=» + sharpness + «}»;  
 }  
}  
  
// Класс Teeth (Зубы)  
class Teeth {  
 private int size; // Размер зубов  
  
 public Teeth(int size) throws InvalidTeethSizeException {  
 if (size <= 0) {  
 throw new InvalidTeethSizeException(«Размер зубов должен быть положительным числом.»);  
 }  
 this.size = size;  
 }  
  
 public int getSize() {  
 return size;  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 if (this == obj) return true;  
 if (obj == null || getClass() != obj.getClass()) return false;  
 Teeth1 teeth = (Teeth1) obj;  
 return size == teeth.getSize();  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(size);  
 }  
  
 @Override  
 public String toString() {  
 return «Teeth{size=» + size + «}»;  
 }  
}  
  
// Класс Predator (Хищник)  
class Predator {  
 private Claws1 claws;  
 private Teeth1 teeth;  
  
 public Predator(Claws1 claws, Teeth1 teeth) {  
 this.claws = claws;  
 this.teeth = teeth;  
 }  
  
 // Метод рычать  
 public void roar() {  
 System.*out*.println(«Хищник рычит!»);  
 }  
  
 // Метод бежать  
 public void run() {  
 System.*out*.println(«Хищник бежит!»);  
 }  
  
 // Метод спать  
 public void sleep() {  
 System.*out*.println(«Хищник спит.»);  
 }  
  
 // Метод добывать пищу  
 public void hunt() {  
 System.*out*.println(«Хищник добывает пищу!»);  
 }  
  
 // Метод записи объекта Predator в файл  
 public void saveToFile(String fileName) {  
 try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(fileName))) {  
 out.writeObject(this);  
 System.*out*.println(«Объект Predator успешно сохранен в файл.»);  
 } catch (IOException e) {  
 System.*err*.println(«Ошибка при сохранении объекта в файл: « + e.getMessage());  
 }  
 }  
  
 // Метод загрузки объекта Predator из файла  
 public static Predator loadFromFile(String fileName) {  
 try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(fileName))) {  
 return (Predator) in.readObject();  
 } catch (FileNotFoundException e) {  
 System.*err*.println(«Файл не найден: « + e.getMessage());  
 } catch (IOException e) {  
 System.*err*.println(«Ошибка при чтении объекта из файла: « + e.getMessage());  
 } catch (ClassNotFoundException e) {  
 System.*err*.println(«Класс Predator не найден: « + e.getMessage());  
 }  
 return null;  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 if (this == obj) return true;  
 if (obj == null || getClass() != obj.getClass()) return false;  
 Predator predator = (Predator) obj;  
 return claws.equals(predator.claws) && teeth.equals(predator.teeth);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(claws, teeth);  
 }  
  
 @Override  
 public String toString() {  
 return «Predator{« +  
 «claws=» + claws +  
 «, teeth=» + teeth +  
 '}';  
 }  
}  
  
// Класс для тестирования  
class Mainn {  
 public static void main(String[] args) {  
 try {  
 Claws1 claws = new Claws1(10); // Создаем когти с остротой 10  
 Teeth1 teeth = new Teeth1(15); // Создаем зубы с размером 15  
  
 Predator predator = new Predator(claws, teeth);  
  
 predator.roar();  
 predator.run();  
 predator.sleep();  
 predator.hunt();  
  
 System.*out*.println(predator);  
  
 // Сохранение объекта в файл  
 predator.saveToFile(«predator.dat»);  
  
 // Загрузка объекта из файла  
 Predator loadedPredator = Predator.*loadFromFile*(«predator.dat»);  
 System.*out*.println(«Загруженный объект: « + loadedPredator);  
  
 } catch (OutOfMemoryError e) {  
 System.*err*.println(«Ошибка: недостаточно памяти для создания объекта Predator.»);  
 }  
 }  
}

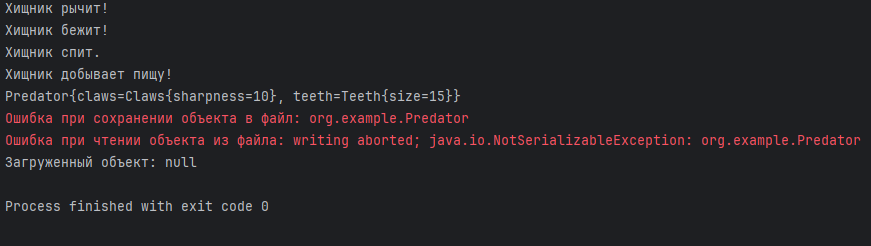


Рисунок 11 – Результат выполнения программы 6

1.7 Задача 7

Составить программу получения на экране рисунка, изображённого на рисунке:

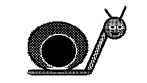


Рисунок 12 – Изображение к задаче 7

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Название переменной** |  |  | | --- | |  | | |  | | --- | | **Тип данных** |  |  | | --- | |  | | Параметр |
| g | Graphics | Вспомогательная |
| g2d | Graphics2D | Вспомогательная |
| |  | | --- | | frame |  |  | | --- | |  | | JFrame | Вспомогательная |

Таблица 7 – Список идентификаторов к задаче 7

Код

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.geom.AffineTransform;  
  
public class SnailDrawing extends JPanel {  
  
 @Override  
 protected void paintComponent(Graphics g) {  
 super.paintComponent(g);  
 Graphics2D g2d = (Graphics2D) g; // Преобразуем Graphics в Graphics2D  
  
 // Настроим качество рисования (антиалиасинг)  
 g2d.setRenderingHint(RenderingHints.*KEY\_ANTIALIASING*, RenderingHints.*VALUE\_ANTIALIAS\_ON*);  
  
  
  
 // Прямоугольник внизу (не повернутый)  
 g2d.setColor(Color.*orange*);  
 g2d.fillRect(150, 450, 400, 20); // Прямоугольник снизу  
  
 // Прямоугольник, повёрнутый на 135 градусов  
 g2d.setColor(Color.*orange*);  
 g2d.rotate(Math.*toRadians*(105), 500, 460); // Поворот вокруг правого конца прямоугольника  
 g2d.fillRect(180, 415, 300, 20); // Рисуем повернутый прямоугольник  
  
 // Восстановим трансформацию, чтобы другие объекты не были повернуты  
 g2d.setTransform(new AffineTransform());  
  
 // Овал на весь прямоугольник (в центре)  
 g2d.setColor(Color.*orange*);  
 g2d.fillOval(125, 220, 370, 230); // Овал на весь прямоугольник  
  
 // В центре овала маленький круг с белой обводкой  
 g2d.setColor(Color.*BLACK*); // Цвет круга  
 g2d.fillOval(220, 270, 190, 180); // Круг в центре  
 g2d.setColor(Color.*WHITE*); // Белая обводка  
 g2d.drawOval(220, 270, 190, 180); // Обводка  
  
 // Уши (антенны)  
 g2d.setColor(Color.*BLACK*);  
 g2d.drawLine(270+250+70, 230-100+40, 250+250+70, 180-100+40); // Левое ухо  
 g2d.drawLine(430+250-70, 230-100+40, 450+250-70, 180-100+40); // Правое ухо  
 // Окружность в центре (для лица)  
 g2d.setColor(Color.*orange*); // Цвет лица  
 g2d.fillOval(300+250, 250-100, 100, 100); // Рисуем лицо (круг)  
  
 g2d.setColor(Color.*WHITE*);  
 g2d.fillOval(330+250-5-5, 280-100-5, 30, 30); // Левый глаз  
 g2d.fillOval(350+250+5-5, 280-100-5, 30, 30); // Правый глаз  
 // Глаза (два маленьких круга)  
 g2d.setColor(Color.*BLACK*);  
 g2d.fillOval(330+250-5, 280-100, 20, 20); // Левый глаз  
 g2d.fillOval(350+250+5, 280-100, 20, 20); // Правый глаз  
  
  
  
 // Улыбка  
 g2d.setColor(Color.*BLACK*);  
 g2d.drawArc(320+250, 310-100, 60, 30, 0, -180); // Рисуем улыбку (дуга)  
  
  
 }  
  
 public static void main(String[] args) {  
 JFrame frame = new JFrame();  
 frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 frame.setSize(700, 500);  
 frame.setLocationRelativeTo(null); // Центрируем окно на экране  
 frame.add(new SnailDrawing());  
 frame.setVisible(true);  
 }  
}

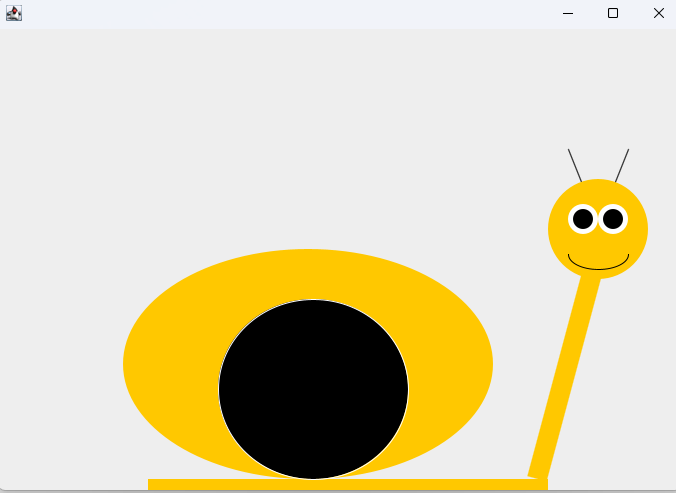


Рисунок 13 – Результат выполнения программы 7

1.8 Задача 8

Позволить щукам поедать карпов, как только они их увидят, для чего следует установить, какого именно карпа видит щука. Проблема решается просмотром всей стаи карпов и определения того из них, чьи координаты близки к координатам данной щуки. Найденного карпа удалить из стаи.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Название переменной** |  |  | | --- | |  | | |  | | --- | | **Тип данных** |  |  | | --- | |  | | Параметр |
| g | Graphics | Вспомогательная |
| g2d | Graphics2D | Вспомогательная |
| |  | | --- | | frame |  |  | | --- | |  | | JFrame | Вспомогательная |

Таблица 8 – Список идентификаторов к задаче 8

Код

import javax.swing.\*;  
import java.awt.\*;  
import java.util.ArrayList;  
import java.util.Iterator;  
import java.util.List;  
  
class TFish {  
 protected int x, y; // Координаты  
 protected int speed;  
 protected int size;  
 protected Color color;  
 protected int direction; // Направление движения в градусах  
  
 public TFish(int x, int y, int speed, int size, Color color, int direction) {  
 this.x = x;  
 this.y = y;  
 this.speed = speed;  
 this.size = size;  
 this.color = color;  
 this.direction = direction;  
 }  
  
 public void draw(Graphics g) {  
 // Абстрактный метод для отображения рыбки  
 }  
  
 public boolean look(List<TFish> others) {  
 for (TFish other : others) {  
 if (other != this && distanceTo(other) < size \* 2) { // в пределах видимости  
 return true;  
 }  
 }  
 return false;  
 }  
  
 public void run() {  
 // Перемещение в текущем направлении  
 x += Math.*cos*(Math.*toRadians*(direction)) \* speed;  
 y += Math.*sin*(Math.*toRadians*(direction)) \* speed;  
 }  
  
 protected double distanceTo(TFish other) {  
 return Math.*sqrt*(Math.*pow*(x - other.x, 2) + Math.*pow*(y - other.y, 2));  
 }  
}  
  
class TPike extends TFish {  
 public TPike(int x, int y, int speed, int size, int direction) {  
 super(x, y, speed, size, Color.*GREEN*, direction);  
 }  
  
 @Override  
 public void draw(Graphics g) {  
 // Отображаем щуку как зеленую стрелку  
 g.setColor(color);  
 int[] xPoints = {x, x - size, x + size};  
 int[] yPoints = {y, y + size, y + size};  
 g.fillPolygon(xPoints, yPoints, 3);  
 }  
  
 public void eat(List<TFish> karps) {  
 Iterator<TFish> iterator = karps.iterator();  
 while (iterator.hasNext()) {  
 TFish karp = iterator.next();  
 if (karp instanceof TKarp && distanceTo(karp) < size \* 2) {  
 iterator.remove();  
 break;  
 }  
 }  
 }  
}  
  
class TKarp extends TFish {  
 public TKarp(int x, int y, int speed, int size, int direction) {  
 super(x, y, speed, size, Color.*RED*, direction);  
 }  
  
 @Override  
 public void draw(Graphics g) {  
 // Отображаем карпа как красный треугольник  
 g.setColor(color);  
 int[] xPoints = {x, x - size, x + size};  
 int[] yPoints = {y, y - size, y - size};  
 g.fillPolygon(xPoints, yPoints, 3);  
 }  
}  
  
public class AquariumSimulation extends JPanel {  
 private List<TFish> fishList = new ArrayList<>();  
  
 public AquariumSimulation() {  
 // Добавляем рыб в аквариум  
 fishList.add(new TPike(100, 100, 2, 10, 0));  
 fishList.add(new TKarp(150, 150, 2, 8, 180));  
 fishList.add(new TKarp(200, 200, 2, 8, 180));  
  
 Timer timer = new Timer(50, e -> {  
 updateFishPositions();  
 repaint();  
 });  
 timer.start();  
 }  
  
 private void updateFishPositions() {  
 // Обновляем позиции рыб  
 for (TFish fish : fishList) {  
 fish.run();  
 }  
  
 // Проверяем, может ли щука съесть карпа  
 for (TFish fish : fishList) {  
 if (fish instanceof TPike) {  
 ((TPike) fish).eat(fishList);  
 }  
 }  
 }  
  
 @Override  
 protected void paintComponent(Graphics g) {  
 super.paintComponent(g);  
 for (TFish fish : fishList) {  
 fish.draw(g);  
 }  
 }  
  
 public static void main(String[] args) {  
 JFrame frame = new JFrame("Aquarium Simulation");  
 AquariumSimulation aquarium = new AquariumSimulation();  
 frame.add(aquarium);  
 frame.setSize(800, 600);  
 frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 frame.setVisible(true);  
 }  
}

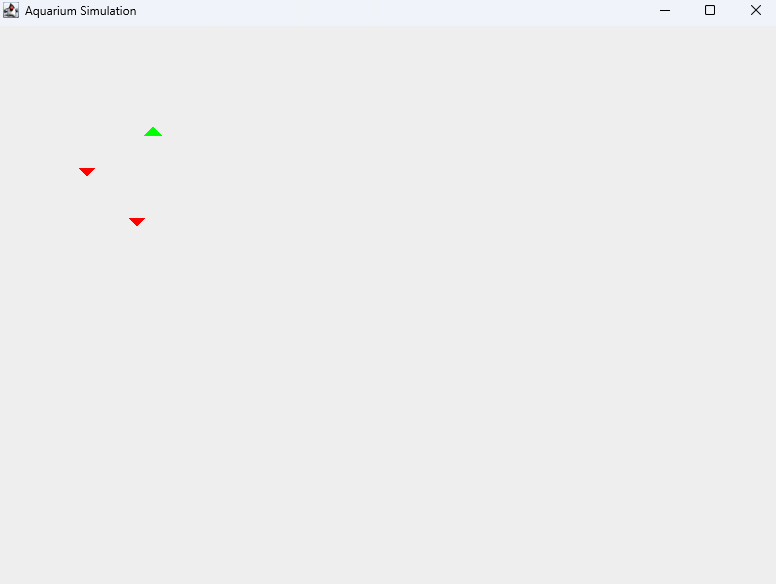


Рисунок 14 – Результат выполнения программы 8

1.9 Задача 9

Реализовать программу «Итальянская игра "Математико"»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Параметр |
| playerField, computerField | int[][] | Вспомогательная |
| currentCard, score | int | Вспомогательная |

Таблица 9 – Список идентификаторов к задаче 9

Код

import java.util.ArrayList;  
import java.util.List;  
import java.util.Random;  
  
public class AIPlayer {  
 private final Random random = new Random();  
  
 public boolean executeMove(GameBoard board, int number) {  
 List<int[]> availableCells = getAvailableCells(board);  
 if (availableCells.isEmpty()) {  
 return false;  
 } else {  
 int[] cell = availableCells.get(random.nextInt(availableCells.size()));  
 board.setCellValue(cell[0], cell[1], number);  
 return true;  
 }  
 }  
  
 private List<int[]> getAvailableCells(GameBoard board) {  
 List<int[]> availableCells = new ArrayList<>();  
 for (int i = 0; i < board.getRowCount(); i++) {  
 for (int j = 0; j < board.getColCount(); j++) {  
 if (board.isCellEmpty(i, j)) {  
 availableCells.add(new int[]{i, j});  
 }  
 }  
 }  
 return availableCells;  
 }  
}

import java.util.ArrayList;  
import java.util.Collections;  
  
public class CardDeck {  
 private ArrayList<Integer> cardPile;  
  
 public CardDeck() {  
 cardPile = new ArrayList<>();  
 for (int i = 1; i <= 13; i++) {  
 for (int j = 0; j < 4; j++) {  
 cardPile.add(i);  
 }  
 }  
 Collections.*shuffle*(cardPile);  
 }  
  
 public int draw() {  
 return cardPile.remove(0);  
 }  
  
 public boolean hasMoreCards() {  
 return !cardPile.isEmpty();  
 }  
}

import javax.swing.\*;  
import java.awt.\*;  
  
public class GameBoard extends JPanel {  
 protected JButton[][] cells;  
 private String boardName;  
 private int[][] cellValues;  
 private MathBattleGame gameController;  
  
 public GameBoard(String boardName, MathBattleGame gameController) {  
 this.boardName = boardName;  
 this.gameController = gameController;  
 this.cellValues = new int[5][5];  
 setLayout(new BorderLayout());  
 JLabel nameLabel = new JLabel(boardName, SwingConstants.CENTER);  
 nameLabel.setFont(new Font("Arial", Font.BOLD, 18));  
 add(nameLabel, BorderLayout.NORTH);  
  
 JPanel gridPanel = new JPanel(new GridLayout(5, 5));  
 cells = new JButton[5][5];  
  
 for (int row = 0; row < 5; row++) {  
 for (int col = 0; col < 5; col++) {  
 cells[row][col] = new JButton("");  
 cells[row][col].setFont(new Font("Arial", Font.PLAIN, 20));  
 cells[row][col].setEnabled(false);  
 gridPanel.add(cells[row][col]);  
  
 if (boardName.equals("Игровое поле")) {  
 int finalRow = row;  
 int finalCol = col;  
 cells[row][col].addActionListener(e -> {  
 if (cells[finalRow][finalCol].isEnabled()) {  
 int number = gameController.getDrawnNumber();  
 cells[finalRow][finalCol].setText(String.valueOf(number));  
 cellValues[finalRow][finalCol] = number;  
 cells[finalRow][finalCol].setEnabled(false);  
 gameController.userMoveCompleted();  
 }  
 });  
 }  
 }  
 }  
 add(gridPanel, BorderLayout.CENTER);  
 }  
  
 public void activateCells(int number) {  
 if (boardName.equals("Игровое поле")) {  
 for (int row = 0; row < 5; row++) {  
 for (int col = 0; col < 5; col++) {  
 if (cells[row][col].getText().equals("")) {  
 cells[row][col].setEnabled(true);  
 }  
 }  
 }  
 }  
 }  
  
 public int getRowCount() {  
 return cellValues.length;  
 }  
  
 public int getColCount() {  
 return cellValues[0].length;  
 }  
  
 public boolean isCellEmpty(int row, int col) {  
 return cellValues[row][col] == 0;  
 }  
  
 public void deactivateCells() {  
 if (boardName.equals("Игровое поле")) {  
 for (int row = 0; row < 5; row++) {  
 for (int col = 0; col < 5; col++) {  
 cells[row][col].setEnabled(false);  
 }  
 }  
 }  
 }  
  
 public void setCellValue(int row, int col, int number) {  
 cells[row][col].setText(String.valueOf(number));  
 cellValues[row][col] = number;  
 }  
  
 public int[][] getCellValues() {  
 return cellValues;  
 }  
  
 public boolean isBoardFull() {  
 for (int row = 0; row < 5; row++) {  
 for (int col = 0; col < 5; col++) {  
 if (cells[row][col].getText().equals("")) {  
 return false;  
 }  
 }  
 }  
 return true;  
 }  
}

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.\*;  
  
public class MathBattleGame extends JFrame implements ActionListener {  
 private CardDeck cardDeck;  
 private GameBoard playerBoard;  
 private GameBoard aiBoard;  
 private JLabel numberDisplayLabel;  
 private JButton generateNumberButton;  
 private int drawnNumber;  
 private AIPlayer aiPlayer;  
 private boolean gameIsOver;  
  
 public MathBattleGame() {  
 setTitle("Битва чисел");  
 setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 setSize(1200, 700);  
 setLayout(new BorderLayout(20, 20)); // Добавляем отступы по краям  
  
 cardDeck = new CardDeck();  
 playerBoard = new GameBoard("Игровое поле", this);  
 aiBoard = new GameBoard("Поле AI", this);  
 aiPlayer = new AIPlayer();  
  
 // Верхняя панель с текущим числом и кнопкой  
 JPanel headerPanel = new JPanel();  
 headerPanel.setLayout(new BoxLayout(headerPanel, BoxLayout.*Y\_AXIS*));  
 headerPanel.setBorder(BorderFactory.*createEmptyBorder*(10, 20, 10, 20)); // Отступы вокруг панели  
  
 numberDisplayLabel = new JLabel("Нажмите 'Сгенерировать число' для начала", SwingConstants.*CENTER*);  
 numberDisplayLabel.setFont(new Font("Arial", Font.*BOLD*, 20)); // Увеличен размер шрифта  
 numberDisplayLabel.setAlignmentX(Component.*CENTER\_ALIGNMENT*);  
  
 generateNumberButton = new JButton("Сгенерировать число");  
 generateNumberButton.setFont(new Font("Arial", Font.*PLAIN*, 18)); // Увеличен размер шрифта кнопки  
 generateNumberButton.addActionListener(this);  
 generateNumberButton.setAlignmentX(Component.*CENTER\_ALIGNMENT*);  
 generateNumberButton.setMargin(new Insets(10, 20, 10, 20)); // Увеличенные отступы внутри кнопки  
  
 headerPanel.add(numberDisplayLabel);  
 headerPanel.add(Box.*createRigidArea*(new Dimension(0, 10))); // Вертикальное пространство между элементами  
 headerPanel.add(generateNumberButton);  
  
 // Центральная панель с полями игрока и компьютера  
 JPanel gridPanel = new JPanel(new GridLayout(1, 2, 20, 20)); // Пространство между полями  
 gridPanel.setBorder(BorderFactory.*createEmptyBorder*(10, 20, 10, 20)); // Внешние отступы панели  
 gridPanel.add(playerBoard);  
 gridPanel.add(aiBoard);  
  
 add(headerPanel, BorderLayout.*NORTH*);  
 add(gridPanel, BorderLayout.*CENTER*);  
  
 gameIsOver = false;  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (!gameIsOver) {  
 if (cardDeck.hasMoreCards()) {  
 drawnNumber = cardDeck.draw();  
 numberDisplayLabel.setText("Сгенерировано число: " + drawnNumber);  
  
 // Ход AI  
 boolean aiMoved = aiPlayer.executeMove(aiBoard, drawnNumber);  
  
 // Разрешаем игроку сделать ход  
 playerBoard.activateCells(drawnNumber);  
 generateNumberButton.setEnabled(false);  
  
 // Если AI не смог сделать ход (поле заполнено)  
 if (!aiMoved) {  
 proceedToNextTurn();  
 }  
 }  
 }  
 }  
  
 public void proceedToNextTurn() {  
 generateNumberButton.setEnabled(true);  
 playerBoard.deactivateCells();  
  
 // Проверяем, заполнены ли поля  
 if (playerBoard.isBoardFull() && aiBoard.isBoardFull()) {  
 gameIsOver = true;  
 numberDisplayLabel.setText("Игра завершена!");  
 generateNumberButton.setEnabled(false);  
  
 // Подсчет очков  
 int playerScore = ScoreCalculator.*computeScore*(playerBoard);  
 int aiScore = ScoreCalculator.*computeScore*(aiBoard);  
  
 // Отображение результатов  
 JOptionPane.*showMessageDialog*(this,  
 "Ваш результат: " + playerScore + "\nРезультат AI: " + aiScore,  
 "Итоги игры",  
 JOptionPane.*INFORMATION\_MESSAGE*);  
 }  
 }  
  
 public void userMoveCompleted() {  
 // Этот метод вызывается после хода игрока  
 proceedToNextTurn();  
 }  
  
 public int getDrawnNumber() {  
 return drawnNumber;  
 }  
  
 public static void main(String[] args) {  
 SwingUtilities.*invokeLater*(() -> {  
 MathBattleGame game = new MathBattleGame();  
 game.setVisible(true);  
 });  
 }  
}

import java.util.\*;  
  
public class ScoreCalculator {  
 public static int computeScore(GameBoard board) {  
 int[][] cellValues = board.getCellValues();  
 int totalScore = 0;  
  
 for (int i = 0; i < 5; i++) {  
 int[] row = cellValues[i];  
 int[] col = new int[5];  
 for (int j = 0; j < 5; j++) {  
 col[j] = cellValues[j][i];  
 }  
 totalScore += evaluateLine(row, false);  
 totalScore += evaluateLine(col, false);  
 }  
  
 int[] diag1 = new int[5];  
 int[] diag2 = new int[5];  
 for (int i = 0; i < 5; i++) {  
 diag1[i] = cellValues[i][i];  
 diag2[i] = cellValues[i][4 - i];  
 }  
 totalScore += evaluateLine(diag1, true);  
 totalScore += evaluateLine(diag2, true);  
  
 return totalScore;  
 }  
  
 private static int evaluateLine(int[] line, boolean isDiagonal) {  
 int score = 0;  
 Map<Integer, Integer> counts = new HashMap<>();  
 for (int num : line) {  
 counts.put(num, counts.getOrDefault(num, 0) + 1);  
 }  
  
 List<Integer> numberList = new ArrayList<>();  
 for (int num : line) {  
 numberList.add(num);  
 }  
  
 boolean lineScored = false;  
  
 if (!lineScored && Collections.frequency(numberList, 1) == 4) {  
 score += isDiagonal ? 210 : 200;  
 lineScored = true;  
 }  
  
 if (!lineScored && numberList.containsAll(Arrays.asList(1, 10, 11, 12, 13))) {  
 score += isDiagonal ? 160 : 150;  
 lineScored = true;  
 }  
  
 if (!lineScored && Collections.frequency(numberList, 1) == 3 && Collections.frequency(numberList, 13) == 2) {  
 score += isDiagonal ? 110 : 100;  
 lineScored = true;  
 }  
  
 if (!lineScored && isConsecutive(numberList)) {  
 score += isDiagonal ? 60 : 50;  
 lineScored = true;  
 }  
  
 if (!lineScored && counts.containsValue(4)) {  
 score += isDiagonal ? 170 : 160;  
 lineScored = true;  
 }  
  
 if (!lineScored && counts.size() == 2 && counts.containsValue(3) && counts.containsValue(2)) {  
 score += isDiagonal ? 90 : 80;  
 lineScored = true;  
 }  
  
 if (!lineScored && counts.containsValue(3)) {  
 score += isDiagonal ? 50 : 40;  
 lineScored = true;  
 }  
  
 if (!lineScored && counts.size() == 3 && Collections.frequency(new ArrayList<>(counts.values()), 2) == 2) {  
 score += isDiagonal ? 30 : 20;  
 lineScored = true;  
 }  
  
 if (!lineScored && counts.containsValue(2)) {  
 score += isDiagonal ? 20 : 10;  
 lineScored = true;  
 }  
  
 return score;  
 }  
  
 private static boolean isConsecutive(List<Integer> numbers) {  
 Set<Integer> uniqueNumbers = new HashSet<>(numbers);  
 if (uniqueNumbers.size() != 5) {  
 return false;  
 }  
 List<Integer> sortedNumbers = new ArrayList<>(uniqueNumbers);  
 Collections.sort(sortedNumbers);  
 for (int i = 0; i < sortedNumbers.size() - 1; i++) {  
 if (sortedNumbers.get(i + 1) - sortedNumbers.get(i) != 1) {  
 return false;  
 }  
 }  
 return true;  
 }  
}

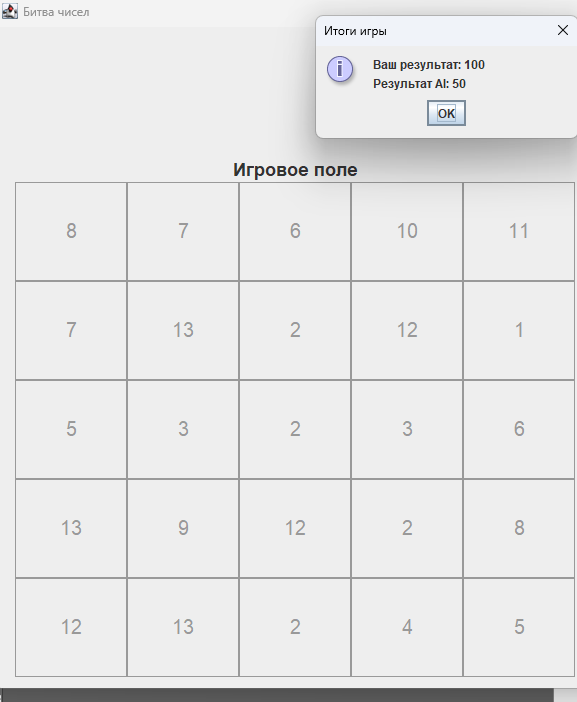


Рисунок 15 – Результат выполнения программы 9

1.10 Задача 10

Задание 1

N DQ ? ; число без знака

Не используя арифметические команды, реализовать оператор

N := N \* 2

Решение должно содержать не более трёх команд.

Критерии оценивания: неверное направление сдвига, теряется раз-

ряд в середине числа (например, сдвиг не через CF) - сразу минус;

не учтено перевёрнутое представление - минус 2/3; другие ошибки,

каждая лишняя команда - минус 1/3 [5].

section .data

N dd 5

result db 'Result: ', 0

newline db 10, 0

section .bss

buf resb 10

section .text

global \_start

\_start:

mov eax, [N]

shl eax, 1

mov [N], eax

mov edi, buf

call int\_to\_string

mov eax, 1

mov edi, 1

mov rsi, result

mov edx, 8

syscall

mov eax, 1

mov edi, 1

mov rsi, buf

mov edx, 10

syscall

mov eax, 1

mov edi, 1

mov rsi, newline

mov edx, 1

syscall

mov eax, 60

xor edi, edi

syscall

int\_to\_string:

mov ecx, 10

xor ebx, ebx

convert\_loop:

xor edx, edx

div ecx

add dl, '0'

mov [rdi], dl

inc rdi

test eax, eax

jnz convert\_loop

ret

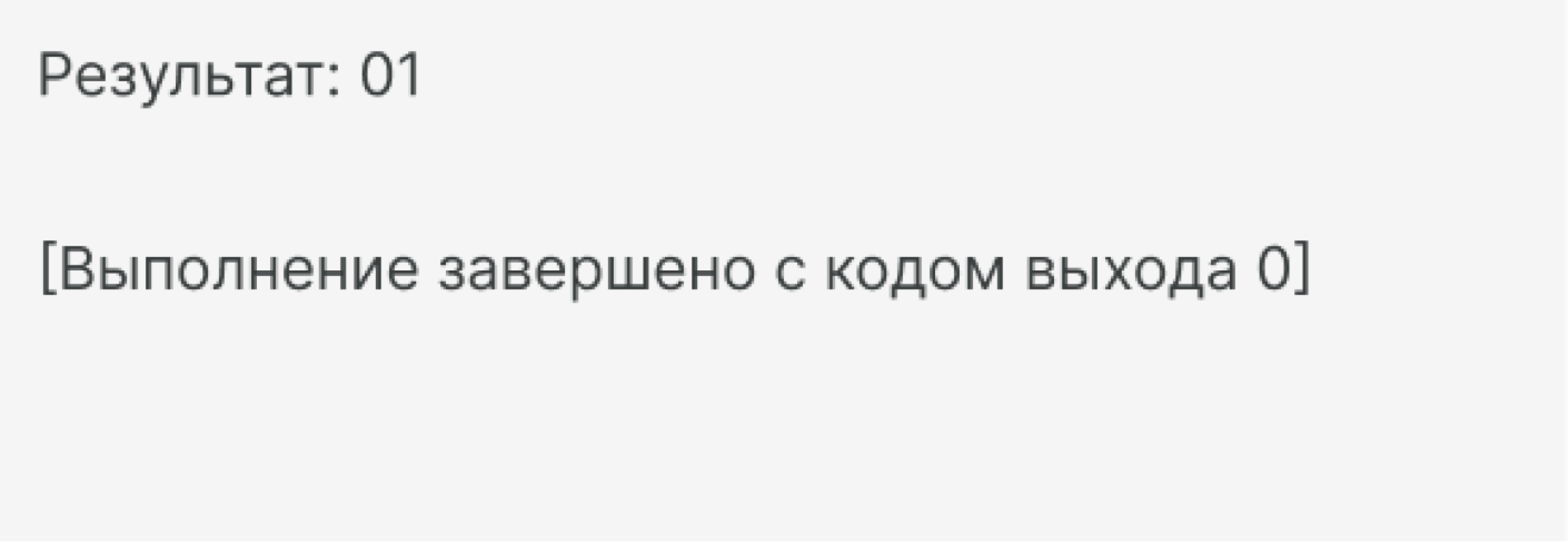


Рисунок 16 – Результат программы Задачи 9 (задание 1)

Задание 2

Реализовать процедуру Pad(x,а), которая реализует оперотор х:=x+а. Передача параметров в стеке, стандартные соглашения о связях.

Критерии оценивания: запись результата в стек, а не в сегмент дан-

ных (неумение работать с параметром-переменной) - минус; каждая

другая ошибка - минус 1/3 [6].

section .data

msg db "Result: ", 0

result db 0, 0, 0, 0, 0

section .text

global \_start

\_start:

mov rdi, 10

mov rsi, 5

call Padd

mov rbx, result

call int\_to\_str

mov rax, 1

mov rdi, 1

mov rsi, msg

mov rdx, 8

syscall

mov rax, 1

mov rdi, 1

mov rsi, result

mov rdx, 5

syscall

mov rax, 60

xor rdi, rdi

syscall

Padd:

add rdi, rsi

mov rax, rdi

ret

int\_to\_str:

mov rcx, 0

mov rbx, result

convert\_loop:

xor rdx, rdx

div rsi

add dl, '0'

mov [rbx + rcx], dl

inc rcx

test rax, rax

jnz convert\_loop

mov rdi, result

mov rsi, rbx

dec rsi

reverse\_loop:

cmp rdi, rsi

jge reverse\_done

mov al, [rdi]

mov bl, [rsi]

mov [rdi], bl

mov [rsi], al

inc rdi

dec rsi

jmp reverse\_loop

reverse\_done:

ret

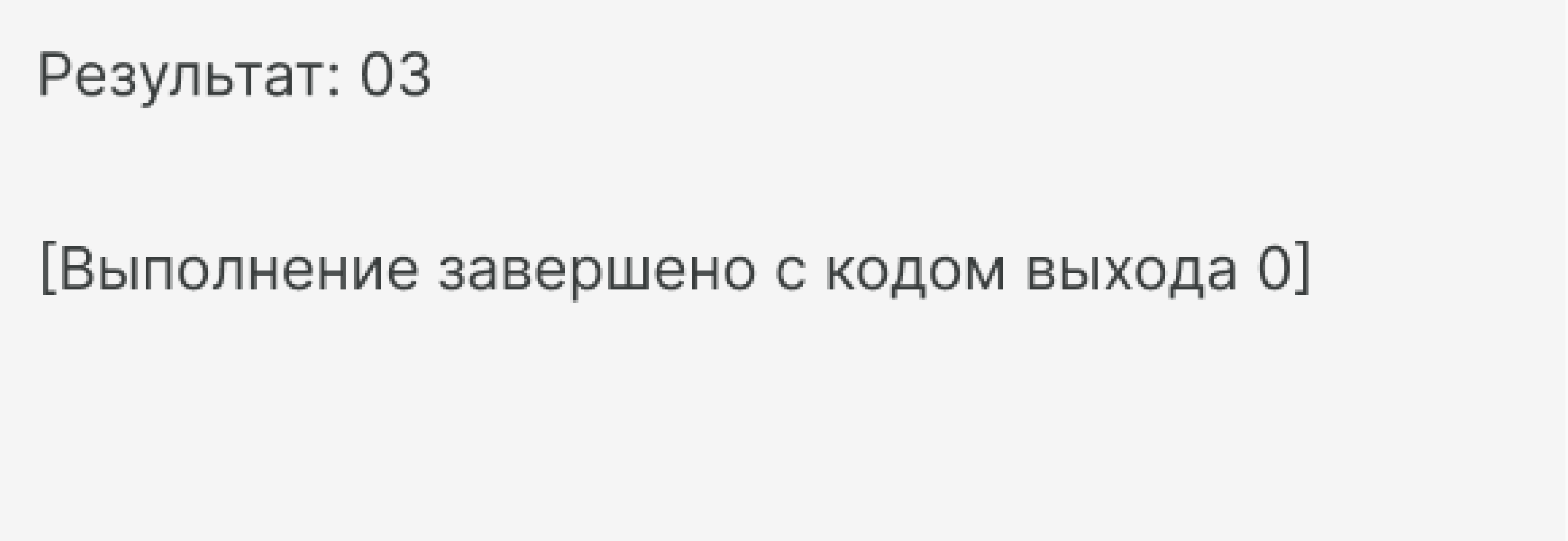


Рисунок 17 – Результат программы Задачи 9 (задание 2)

2 Разработка АИС

Тема: Тех. Поддержка

В качестве серверной части разработать базу данных в СУБД MySQL в соответствии с вариантом. База данных содержит таблицу, состоящую из не менее чем 5-ти полей и 20-ти записей [7].

Клиентская часть должна быть разработана на языке Java. Каждый запрос должен быть реализован отдельным методом. Все методы могут принадлежать одному классу. Отдельно должен быть реализован класс, содержащий метод main().

Таким образом, структура клиента должна быть такой:

1. Класс, содержащий статический метод main() для демонстрации возможностей программы.

2. Класс, содержащий поля и методы для работы с удалённой базой данных:

3. Обязательные методы (запросы) клиента:

- выборка содержимого таблиц базы данных;

- запись в файл результата выборки содержимого таблиц БД.

- вывод в консоль результатов выполнения каждого метода.

4. Дополнительные методы (запросы) клиента

a. добавление записи в таблицу.

b. удаление записи из таблицы.

c. обновление записи в таблице.

d. поиск записи по признаку (на ваш выбор).

2.1 Техническое задание

1. Общие сведения

1.1. Название системы: «Система технической поддержки»  
1.2. Разработчик: Толстов Владислав Андреевич  
1.3. Основание для разработки: необходимость автоматизации обработки заявок на техническую поддержку, улучшение учета тикетов и управления ролями пользователей.

2. Назначение системы

2.1. Система предназначена для автоматизации процесса управления заявками на техническую поддержку, учета тикетов, а также организации доступа к функционалу системы в зависимости от роли пользователя (администратор, пользователь).  
 2.2. Система должна обеспечивать:

Ведение базы данных тикетов, включая информацию о клиенте, описании проблемы, статусе заявки и дате создания;

Разделение функциональности для администраторов и пользователей, где администраторы могут добавлять и изменять тикеты, а пользователи могут только просматривать.

3. Требования к системе

3.1. Функциональные требования

Модуль управления тикетами:

* Просмотр списка тикетов с указанием идентификатора, имени клиента, описания проблемы, статуса и даты создания.
* Добавление новых тикетов (доступно только администратору).
* Редактирование статуса существующих тикетов (доступно только администратору).
* Поиск тикетов по имени клиента или описанию проблемы.
* Фильтрация тикетов по статусу.
* Удаление тикетов (доступно только администратору).

Модуль аутентификации:

* Авторизация пользователей с ролями «администратор» и «пользователь»;
* Ограничение доступа к функциональности на основе роли.

3.2. Нефункциональные требования

* Удобный и интуитивно понятный пользовательский интерфейс для управления и отображения тикетов.
* Обеспечение сохранности данных и защиты от несанкционированного доступа.

4. Требования к интерфейсу пользователя

4.1. Система должна иметь простой интерфейс, разделённый на экран авторизации для входа в систему, главное меню с отображением списка тикетов и доступными действиями в зависимости от роли пользователя.

4.2. Интерфейс должен быть адаптирован для работы на компьютерах и не требовать сложного обучения пользователя.

Требования к безопасности

5.1. Система должна требовать ввод логина и пароля для входа;

5.2. Необходимо разграничение прав доступа:

5.3 Администратор имеет доступ к полному функционалу (добавление, изменение, удаление тикетов);

5.4 Пользователь может только просматривать список тикетов.

2.2 Краткая спецификация разрабатываемой системы

Цель системы:  
Создание системы управления заявками на техническую поддержку для упрощения процессов учета, обработки заявок и управления доступом пользователей к системе.

Сущности системы:

1. Заявки (тикеты)  
   Хранит информацию о заявках на техническую поддержку. Включает данные о клиенте, описании проблемы, статусе заявки и дате создания.
2. Пользователи  
   Содержит данные о пользователях системы, включая их логин, роль (администратор или пользователь) и пароль для аутентификации.

Входные данные:

* Заявки: информация о клиенте, описании проблемы, статусе и дате создания заявки.
* Пользователи: данные для входа в систему (логин и пароль), а также информация о роли пользователя (администратор или пользователь).
* Выходные данные:
* Отчёты по заявкам: информация о текущих заявках, их статусе и дате создания.
* Информация по пользователям: данные о пользователях системы, включая их роли и права доступа.

Основные функции системы:

1. Управление заявками на техническую поддержку:
   * Создание новых заявок (только для администратора).
   * Просмотр всех заявок.
   * Обновление статуса заявок (только для администратора).
   * Поиск заявок по клиенту или описанию проблемы.
   * Фильтрация заявок по статусу.
   * Удаление заявок (только для администратора).
2. Управление пользователями и ролями:
   * Аутентификация пользователей с помощью логина и пароля.
   * Разграничение прав доступа на основе роли пользователя:
     + Администратор: полный доступ к функционалу системы.
     + Пользователь: доступ только для просмотра списка заявок.
3. Управление интерфейсом:
   * Отображение главного меню с доступным функционалом в зависимости от роли пользователя.
   * Удобный интерфейс для поиска и фильтрации заявок.

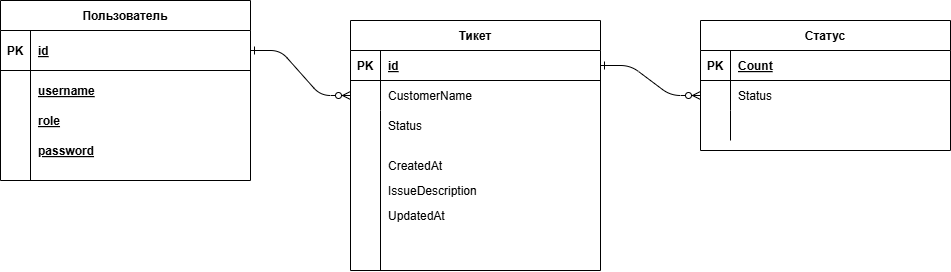


Рисунок 18 – Er-диаграмма

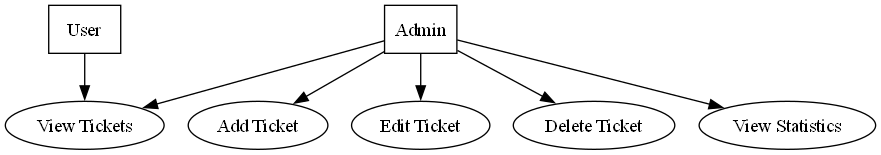


Рисунок 19 – use-case диаграмма

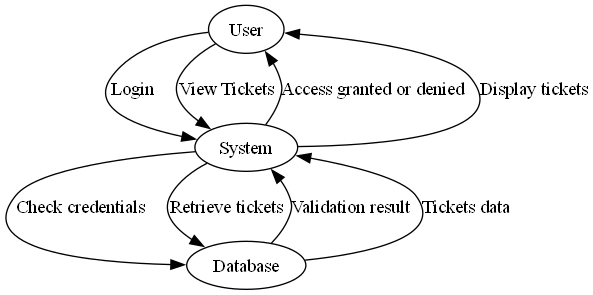


Рисунок 20 – Sequence диаграмма

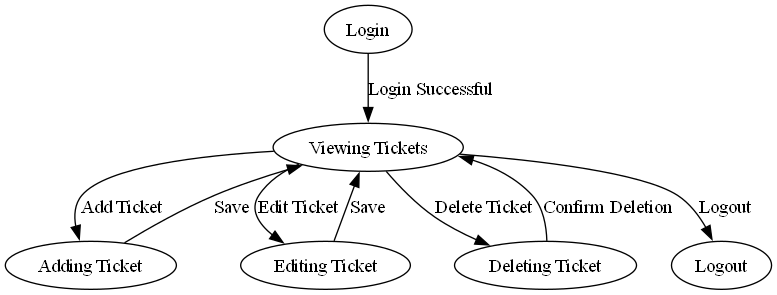


Рисунок 21 – State диаграмма

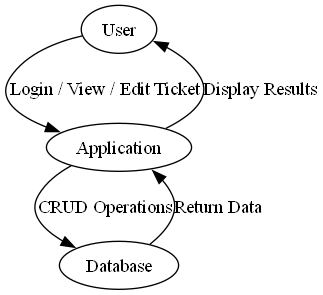


Рисунок 22 – Диаграмма кооперации

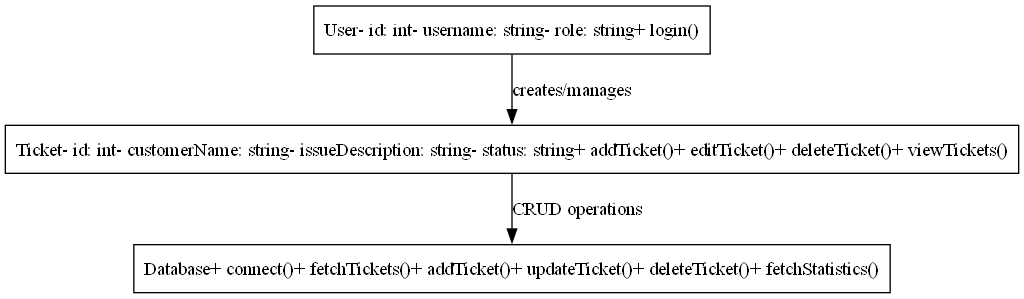


Рисунок 23 – Диаграмма классов

Код

Backend.py

import mysql.connector  
from mysql.connector import Error  
import csv  
  
class DatabaseClient:  
 def \_\_init\_\_(self):  
 try:  
 # Подключение к удаленной базе данных  
 self.connection = mysql.connector.connect(  
 host='autorack.proxy.rlwy.net'**,** user='root'**,** password='viHZOzjbzONsUjKTjRzscrtAeNBnJWie'**,** database='railway'**,** port=20456**,** charset="utf8mb4" # Устанавливаем кодировку соединения  
 )  
  
 if self.connection.is\_connected():  
 print("Connected to MySQL database")  
 except Error as e:  
 print(f"Error: {e}")  
  
 def fetch\_all\_tickets(self):  
 try:  
 cursor = self.connection.cursor()  
 cursor.execute("SELECT \* FROM tickets")  
 records = cursor.fetchall()  
 for row in records:  
 print(row)  
 return records  
 except Error as e:  
 print(f"Error fetching tickets: {e}")  
  
 def add\_ticket(self**,** customer\_name**,** issue\_description**,** status="open"):  
 try:  
 cursor = self.connection.cursor()  
 cursor.execute(  
 "INSERT INTO tickets (customer\_name, issue\_description, status) VALUES (%s, %s, %s)"**,** (customer\_name**,** issue\_description**,** status)  
 )  
 self.connection.commit()  
 print("Ticket added successfully.")  
 except Error as e:  
 print(f"Error adding ticket: {e}")  
  
 def delete\_ticket(self**,** ticket\_id):  
 try:  
 cursor = self.connection.cursor()  
 cursor.execute("DELETE FROM tickets WHERE id = %s"**,** (ticket\_id**,**))  
 self.connection.commit()  
 print(f"Ticket with ID {ticket\_id} deleted successfully.")  
 except Error as e:  
 print(f"Error deleting ticket: {e}")  
  
 def update\_ticket(self**,** ticket\_id**,** status):  
 try:  
 cursor = self.connection.cursor()  
 cursor.execute("UPDATE tickets SET status = %s, updated\_at = NOW() WHERE id = %s"**,** (status**,** ticket\_id))  
 self.connection.commit()  
 print("Ticket updated successfully.")  
 except Error as e:  
 print(f"Error updating ticket: {e}")  
  
 def search\_tickets(self**,** search\_text):  
 try:  
 cursor = self.connection.cursor()  
 query = "SELECT \* FROM tickets WHERE customer\_name LIKE %s OR issue\_description LIKE %s"  
 cursor.execute(query**,** (f"%{search\_text}%"**,** f"%{search\_text}%"))  
 results = cursor.fetchall()  
 for row in results:  
 print(row)  
 return results  
 except Error as e:  
 print(f"Error searching tickets: {e}")  
  
 def fetch\_tickets\_by\_status(self**,** status):  
 try:  
 cursor = self.connection.cursor()  
 cursor.execute("SELECT \* FROM tickets WHERE status = %s"**,** (status**,**))  
 return cursor.fetchall()  
 except Error as e:  
 print(f"Error fetching tickets by status: {e}")  
  
 def get\_ticket\_statistics(self):  
 try:  
 cursor = self.connection.cursor(dictionary=True)  
 cursor.execute("SELECT status, COUNT(\*) AS count FROM tickets GROUP BY status")  
 stats = {row['status']: row['count'] for row in cursor.fetchall()}  
 return {"open": stats.get("open"**,** 0)**,** "in\_progress": stats.get("in\_progress"**,** 0)**,** "closed": stats.get("closed"**,** 0)}  
 except Error as e:  
 print(f"Error fetching ticket statistics: {e}")  
 return {"open": 0**,** "in\_progress": 0**,** "closed": 0}  
  
 def save\_to\_file(self**,** filename="tickets.txt"):  
 tickets = self.fetch\_all\_tickets()  
 with open(filename**,** "w") as file:  
 for ticket in tickets:  
 file.write(str(ticket) + "\n")  
 print(f"Saved tickets to {filename}")  
  
 def export\_to\_csv(self**,** filepath):  
 try:  
 tickets = self.fetch\_all\_tickets()  
 with open(filepath**,** mode='w'**,** newline='') as file:  
 writer = csv.writer(file)  
 writer.writerow(["ID"**,** "Customer Name"**,** "Issue Description"**,** "Status"**,** "Created At"**,** "Updated At"])  
 writer.writerows(tickets)  
 print(f"Tickets exported to {filepath}")  
 except Error as e:  
 print(f"Error exporting to CSV: {e}")  
  
 # Add this method in DatabaseClient in Backend.py  
 def verify\_user(self**,** username**,** password):  
 try:  
 cursor = self.connection.cursor(dictionary=True)  
 cursor.execute("SELECT role FROM users WHERE username = %s AND password = %s"**,** (username**,** password))  
 user = cursor.fetchone()  
 return user['role'] if user else None  
 except Error as e:  
 print(f"Error verifying user: {e}")  
 return None

frontend.py

from PyQt6.QtWidgets import QApplication**,** QMainWindow**,** QVBoxLayout**,** QPushButton**,** QTableWidget**,** QTableWidgetItem**,** \  
 QWidget**,** QLineEdit**,** QLabel**,** QComboBox**,** QDialog**,** QMessageBox**,** QFileDialog**,** QInputDialog  
from Backend import DatabaseClient  
import sys  
  
  
class LoginDialog(QDialog):  
 def \_\_init\_\_(self**,** client):  
 super().\_\_init\_\_()  
 self.client = client  
 self.role = None  
 self.setWindowTitle("Login")  
  
 self.layout = QVBoxLayout()  
  
 self.username\_input = QLineEdit()  
 self.username\_input.setPlaceholderText("Username")  
 self.layout.addWidget(self.username\_input)  
  
 self.password\_input = QLineEdit()  
 self.password\_input.setPlaceholderText("Password")  
 self.password\_input.setEchoMode(QLineEdit.EchoMode.Password)  
 self.layout.addWidget(self.password\_input)  
  
 self.login\_button = QPushButton("Login")  
 self.login\_button.clicked.connect(self.login)  
 self.layout.addWidget(self.login\_button)  
  
 self.setLayout(self.layout)  
  
 def login(self):  
 username = self.username\_input.text()  
 password = self.password\_input.text()  
 self.role = self.client.verify\_user(username**,** password)  
 if self.role:  
 self.accept()  
 else:  
 QMessageBox.warning(self**,** "Login Failed"**,** "Invalid credentials")  
  
  
class TicketApp(QMainWindow):  
 def \_\_init\_\_(self**,** role):  
 super().\_\_init\_\_()  
 self.client = DatabaseClient()  
 self.role = role  
 self.setWindowTitle("Technical Support System")  
 self.setGeometry(100**,** 100**,** 800**,** 600)  
  
 self.layout = QVBoxLayout()  
  
 # Ticket Table  
 self.ticket\_table = QTableWidget()  
 self.layout.addWidget(self.ticket\_table)  
  
 # Search Box  
 self.search\_box = QLineEdit()  
 self.search\_box.setPlaceholderText("Search by customer name or issue description")  
 self.layout.addWidget(self.search\_box)  
  
 # Buttons  
 self.refresh\_button = QPushButton("Refresh Tickets")  
 self.refresh\_button.clicked.connect(self.load\_tickets)  
 self.layout.addWidget(self.refresh\_button)  
  
 # Buttons restricted for 'user' role  
 if self.role == "admin":  
 self.add\_ticket\_button = QPushButton("Add New Ticket")  
 self.add\_ticket\_button.clicked.connect(self.add\_ticket)  
 self.layout.addWidget(self.add\_ticket\_button)  
  
 self.search\_button = QPushButton("Search Tickets")  
 self.search\_button.clicked.connect(self.search\_tickets)  
 self.layout.addWidget(self.search\_button)  
  
 self.edit\_ticket\_button = QPushButton("Edit Ticket")  
 self.edit\_ticket\_button.clicked.connect(self.edit\_ticket)  
 self.layout.addWidget(self.edit\_ticket\_button)  
  
 self.status\_filter = QComboBox()  
 self.status\_filter.addItem("All")  
 self.status\_filter.addItem("open")  
 self.status\_filter.addItem("in\_progress")  
 self.status\_filter.addItem("closed")  
 self.status\_filter.currentTextChanged.connect(self.apply\_status\_filter)  
 self.layout.addWidget(self.status\_filter)  
  
 self.delete\_ticket\_button = QPushButton("Delete Selected Ticket")  
 self.delete\_ticket\_button.clicked.connect(self.delete\_ticket)  
 self.layout.addWidget(self.delete\_ticket\_button)  
  
 self.export\_button = QPushButton("Export to CSV")  
 self.export\_button.clicked.connect(self.export\_to\_csv)  
 self.layout.addWidget(self.export\_button)  
  
 self.stats\_button = QPushButton("Show Statistics")  
 self.stats\_button.clicked.connect(self.show\_ticket\_statistics)  
 self.layout.addWidget(self.stats\_button)  
  
 container = QWidget()  
 container.setLayout(self.layout)  
 self.setCentralWidget(container)  
 self.load\_tickets()  
  
 def load\_tickets(self):  
 tickets = self.client.fetch\_all\_tickets()  
 self.display\_tickets(tickets)  
  
 def display\_tickets(self**,** tickets):  
 self.ticket\_table.setRowCount(len(tickets))  
 self.ticket\_table.setColumnCount(6)  
 self.ticket\_table.setHorizontalHeaderLabels(["ID"**,** "Customer Name"**,** "Issue Description"**,** "Status"**,** "Created At"**,** "Updated At"])  
 for row\_num**,** row\_data in enumerate(tickets):  
 for col\_num**,** data in enumerate(row\_data):  
 self.ticket\_table.setItem(row\_num**,** col\_num**,** QTableWidgetItem(str(data)))  
  
 def add\_ticket(self):  
 dialog = QDialog(self)  
 dialog.setWindowTitle("Добавить новый тикет")  
  
 layout = QVBoxLayout()  
  
 # Поле для ввода имени клиента  
 name\_label = QLabel("Имя клиента:")  
 layout.addWidget(name\_label)  
 name\_input = QLineEdit()  
 layout.addWidget(name\_input)  
  
 # Поле для описания проблемы  
 description\_label = QLabel("Описание проблемы:")  
 layout.addWidget(description\_label)  
 description\_input = QLineEdit()  
 layout.addWidget(description\_input)  
  
 # Поле выбора статуса  
 status\_label = QLabel("Статус:")  
 layout.addWidget(status\_label)  
 status\_input = QComboBox()  
 status\_input.addItems(["open"**,** "in\_progress"**,** "closed"])  
 layout.addWidget(status\_input)  
  
 # Кнопка для подтверждения добавления тикета  
 add\_button = QPushButton("Добавить тикет")  
 add\_button.clicked.connect(lambda: self.confirm\_add\_ticket(dialog**,** name\_input.text()**,** description\_input.text()**,** status\_input.currentText()))  
 layout.addWidget(add\_button)  
  
 dialog.setLayout(layout)  
 dialog.exec()  
  
 def confirm\_add\_ticket(self**,** dialog**,** customer\_name**,** issue\_description**,** status):  
 if customer\_name and issue\_description:  
 self.client.add\_ticket(customer\_name**,** issue\_description**,** status)  
 self.load\_tickets()  
 dialog.accept()  
 else:  
 QMessageBox.warning(self**,** "Ошибка"**,** "Пожалуйста, заполните все поля перед добавлением.")  
  
 def search\_tickets(self):  
 search\_text = self.search\_box.text()  
 results = self.client.search\_tickets(search\_text)  
 self.display\_tickets(results)  
  
 def edit\_ticket(self):  
 selected\_row = self.ticket\_table.currentRow()  
 if selected\_row < 0:  
 QMessageBox.warning(self**,** "Редактирование тикета"**,** "Выберите тикет для редактирования.")  
 return  
 ticket\_id = self.ticket\_table.item(selected\_row**,** 0).text()  
 new\_status**,** ok = QInputDialog.getText(self**,** "Редактировать статус тикета"**,** "Введите новый статус (open, in\_progress, closed):")  
 if ok and new\_status in ["open"**,** "in\_progress"**,** "closed"]:  
 self.client.update\_ticket(ticket\_id**,** new\_status)  
 self.load\_tickets()  
  
 def apply\_status\_filter(self):  
 selected\_status = self.status\_filter.currentText()  
 if selected\_status == "Все":  
 self.load\_tickets()  
 else:  
 tickets = self.client.fetch\_tickets\_by\_status(selected\_status)  
 self.display\_tickets(tickets)  
  
 def delete\_ticket(self):  
 selected\_row = self.ticket\_table.currentRow()  
 if selected\_row < 0:  
 QMessageBox.warning(self**,** "Удаление тикета"**,** "Выберите тикет для удаления.")  
 return  
 ticket\_id = self.ticket\_table.item(selected\_row**,** 0).text()  
 self.client.delete\_ticket(ticket\_id)  
 self.load\_tickets()  
 QMessageBox.information(self**,** "Удаление тикета"**,** "Тикет успешно удален.")  
  
 def export\_to\_csv(self):  
 path**,** \_ = QFileDialog.getSaveFileName(self**,** "Сохранить CSV"**,** ""**,** "CSV files (\*.csv)")  
 if path:  
 self.client.export\_to\_csv(path)  
  
 def show\_ticket\_statistics(self):  
 stats = self.client.get\_ticket\_statistics()  
 message = f"Открытые тикеты: {stats['open']}\nВ процессе: {stats['in\_progress']}\nЗакрытые тикеты: {stats['closed']}"  
 QMessageBox.information(self**,** "Статистика тикетов"**,** message)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app = QApplication(sys.argv)  
  
 # Database client  
 client = DatabaseClient()  
  
 # Display login dialog  
 login\_dialog = LoginDialog(client)  
 if login\_dialog.exec() == QDialog.DialogCode.Accepted:  
 # Start main application with the appropriate role  
 window = TicketApp(login\_dialog.role)  
 window.show()  
 sys.exit(app.exec())

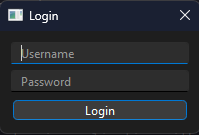


Рисунок 24 – Окно авторизации

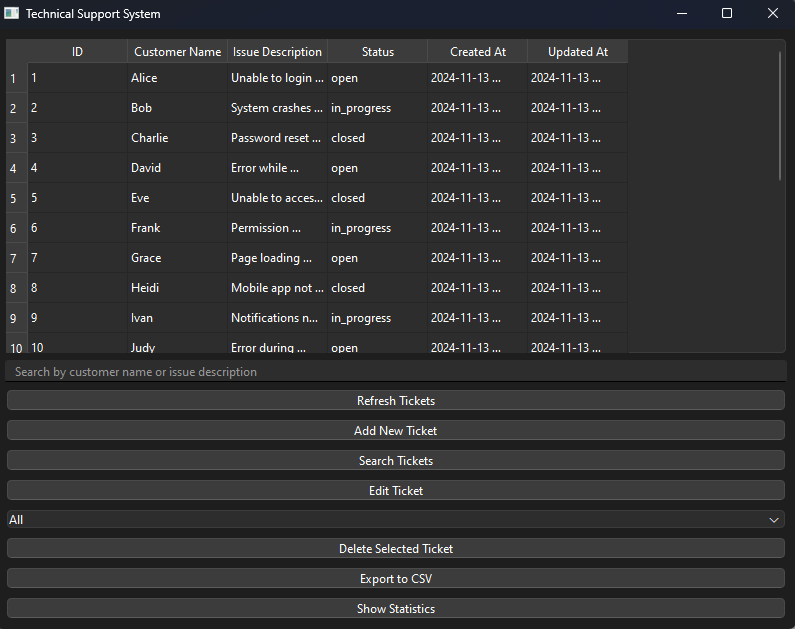


Рисунок 25 – Главное меню (Админ)

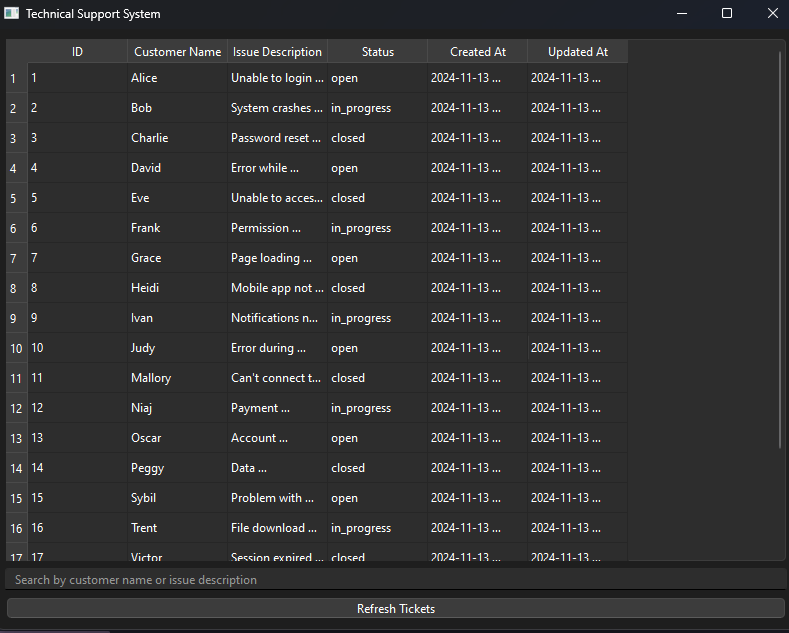


Рисунок 26 – Главное меню (Пользователь)

2.3 Тестирование АИС

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Шаги | Ожидаемый результат | Статус |
| Тестирование аутентификации | | | | |
| 1 | Успешная аутентификация обычного пользователя | Войти в систему через учётную запись обычного пользователя | Вход в систему без возможности редактировать данные | Успешно пройден |
| 2 | Успешная аутентификация администратора | Войти в систему через учётную запись администратора | Вход в систему с возможностью редактировать данные | Успешно пройден |
| 3 | Ошибка при вводе неправильных данных | Войти в систему используя неправильные пароль или номер телефона | Появление ошибки неправильных данных | Успешно пройден |

|  |
| --- |
| Проверка отображения данных |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | Успешное отображение информации о тикете | Выбрать тикет из списка тикетов | Отображение информации о выбранном тикете | Успешно пройден |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | Успешное отображение информации о пользователе | Выбрать пользователя из списка пользователей | Отображение информации о выбранном пользователе | Успешно пройден |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6 | Успешное отображение информации о категории | Выбрать категорию из списка категорий | Отображение информации о выбранной категории | Не пройден (функция отсутствует) |
| Тестирование редактирования данных | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | Добавление нового тикета | |  | | --- | | Заполнить поля с информацией о новом тикете, нажать кнопку «добавить» |  |  | | --- | |  | | |  | | --- | | Тикет добавлен и отображается в списке тикетов |  |  | | --- | |  | | Успешно пройден |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | Добавление нового пользователя | |  | | --- | | Заполнить поля с информацией о новом пользователе, нажать кнопку «добавить» |  |  | | --- | |  | | |  | | --- | | Пользователь добавлен и отображается в списке пользователей |  |  | | --- | |  | | Успешно пройден |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9 | Добавление новой категории | |  | | --- | | Заполнить поля с информацией о новой категории, нажать кнопку «добавить» |  |  | | --- | |  | | |  | | --- | | Категория добавлена и отображается в списке категорий |  |  | | --- | |  | | Не пройден (функция отсутствует) | | |
| 10 | |  | | --- | | Удаление тикета |  |  | | --- | |  | | |  | | --- | | Ввести идентификатор тикета, нажать кнопку «удалить» |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Пользователь удален и не отображается в списке пользователей |  |  | | --- | |  | |  |  | | --- | |  | | | Успешно пройден |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 11 | |  | | --- | | Удаление пользователя |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Ввести идентификатор категории, нажать кнопку «удалить» |  |  | | --- | |  | |  |  | | --- | |  | | Отображение информации о выбранном пользователе | Успешно пройден |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 12 | |  | | --- | | Удаление категории |  |  | | --- | |  | | Выбрать категорию из списка категорий | |  | | --- | | Категория удалена и не отображается в списке категорий |  |  | | --- | |  | | Не пройден (функция отсутствует) |

Таблица 10 – Таблица тест-кейсов

2.4 Оценка программного продукта

Выбранный паттерн проектирования: Model-View-Presenter (MVP)

Обоснование: Паттерн MVP обеспечивает более явное разделение между слоями и улучшает тестируемость приложения. Model отвечает за управление данными, их получение и обработку. Presenter обрабатывает логику, взаимодействует с Model для получения данных и обновления представления. View отвечает только за отображение данных и отправляет действия пользователя в Presenter.

|  |  |  |  |
| --- | --- | --- | --- |
| Критерий | Цель | Критерий оценки | Оценка (2-5) |
| Чёткость разделения слоев | Проверка, что каждый слой выполняет только свои задачи | Model управляет данными; Presenter обрабатывает логику и контролирует поток данных; View — только UI. | 4 |
| Гибкость и расширяемость | Оценка легкости изменения логики, интерфейса, добавления функционала | Новые функции добавляются изменением Presenter и Model, без необходимости изменять View. | 4 |
| Тестируемость | Легкость тестирования слоев независимо | Presenter тестируется с моками для Model и View; View тестируется с фиктивными данными. | 3 |
| Переиспользуемость кода | Возможность использовать код в других частях приложения или проектах | Компоненты Presenter и Model могут быть переиспользованы в других проектах и частях приложения. | 3 |
| Производительность и оптимизация | Проверка эффективности работы приложения | Model минимизирует обращения к базе данных; Presenter поддерживает оптимальные обновления View. | 4 |
| Безопасность и изоляция доступа | Убедиться, что каждый слой защищен и изолирован от несанкционированного доступа | Presenter контролирует доступ к данным; Model не раскрывает данные напрямую в View. | 3 |

Таблица 11 – Таблица оценки программы

Итого: 3.5

Пояснение к оценке

1. Чёткость разделения слоев (Оценка: 4)  
   MVP обеспечивает разделение обязанностей. Model управляет только данными, View — интерфейсом, а Presenter — логикой, что делает код проще и модульнее.
2. Гибкость и расширяемость (Оценка: 4)  
   MVP позволяет добавлять или изменять функционал, минимально влияя на другие слои. Новые функции обычно требуют изменений только в Presenter и Model, что упрощает развитие приложения.
3. Тестируемость (Оценка: 3)  
   Presenter можно тестировать отдельно с моками, так как он не зависит напрямую от View. Тем не менее, для тестирования View требуется настройка окружения.
4. Переиспользуемость кода (Оценка: 3)  
   Код в Model и Presenter можно переиспользовать. Presenter содержит логику, а Model хранит данные и может быть перенесена в другие приложения.
5. Производительность и оптимизация (Оценка: 4)  
   Model эффективно работает с данными, а Presenter минимизирует обновления UI, что способствует быстрому выполнению операций и повышает производительность.
6. Безопасность и изоляция доступа (Оценка: 3)  
   В MVP доступ к данным идет через Presenter, который выполняет проверки и передает только необходимую информацию, защищая Model и View от прямого доступа.

2.5 Руководство пользователя

1. Общее описание

Программа «Техническая поддержка» предназначена для оптимизации работы службы технической поддержки, позволяя отслеживать обращения, управлять их статусом, получать информацию о клиентах и сотрудниках. Программа упрощает обработку заявок и повышает эффективность управления задачами.

2. Системные требования

* Операционная система: Windows 10 и выше
* Память: минимум 4 ГБ ОЗУ
* Свободное место на диске: минимум 500 МБ
* База данных: MySQL или SQLite (.db)

3. Установка программы

1. Перейдите на официальный сайт ([insil3nc3/UP: Практические задания](https://github.com/insil3nc3/UP)).
2. На странице проекта можно увидеть информацию о программе и скачать установочный файл.

4. Интерфейс программы

Окно авторизации включает поля ввода логина и пароля, а также кнопку входа. При некорректных данных отображается сообщение об ошибке. При успешной авторизации открывается основное окно работы с заявками.

Основное окно

* Левая панель: отображает список всех заявок (по умолчанию) и кнопку для переключения к отображению списка клиентов.
* Центральная панель: отображает подробную информацию о выбранной заявке (или клиенте).
* Правая панель: отображает список всех сотрудников и важную информацию о выбранном сотруднике.

5. Функциональность

1. Управление заявками:

* Отображение информации о выбранной заявке;
* Изменение статуса заявки;
* Удаление заявки.

1. Управление клиентами:

* Отображение информации о выбранном клиенте;
* Управление сотрудниками:
* Просмотр и редактирование информации о сотрудниках.

6. Часто возникающие ошибки

1. Ошибка «Заполните все поля»  
   Проверьте, что все обязательные поля заполнены.
2. Ошибка подключения к базе данных  
   Убедитесь, что база данных доступна и правильно настроена. Проверьте конфигурацию подключения к базе.

7. Советы по эффективному использованию

1. Регулярно обновляйте информацию о заявках и статусах для актуальности.
2. Поддерживайте актуальную базу данных клиентов и сотрудников.
3. Используйте программу на последней версии.

8. При возникновении вопросов

Обратитесь в службу поддержки по почте agarkacc@gmail.com.  
График работы службы поддержки:

* ПН-ПТ: с 9:00 до 18:00
* СБ-ВС: выходной

2.6 Руководство программиста

Руководство программиста

1. Архитектура

Уровни архитектуры:

* View: отображение интерфейса программы и получение взаимодействия с пользователем.
* Presenter: обработка действий пользователя, логика программы и взаимодействие с данными.
* Model: управление данными, включая доступ к базе данных и их обработку.
* Основные компоненты:
* Аутентификация: разделение доступа для пользователей и администраторов.
* Управление заявками: просмотр и изменение статуса заявок (доступно только администраторам).
* Управление клиентами и сотрудниками: предоставление информации о клиентах и сотрудниках.

2. Технологии и зависимости

* Backend: Python
* Frontend: PyQt5
* База данных: MySQL на хостинге Railway
* Среда разработки: PyCharm
* Python версии: 3.10 и выше

3. Структура базы данных

* Пользователи: хранит данные (логин, пароль).
* Заявки: включает информацию о клиенте, описание проблемы, статус и дату создания.

4. Основные сценарии использования

Аутентификация:

* Ввод логина и пароля для проверки доступа;
* Вход в систему при корректных данных.
* Управление заявками:
* Отображение информации о выбранной заявке;
* Изменение статуса заявки;
* Удаление заявки.
* Управление клиентами:
* Отображение информации о выбранном клиенте.
* Управление сотрудниками:
* Отображение информации о выбранном сотруднике.

3 Программирование 1С

Разработка конфигурации для системы технической поддержки

Цель разработки: создать автоматизированную систему для управления службой технической поддержки, обеспечивающую удобное взаимодействие с клиентами и сотрудниками, а также управление заявками и их статусами.

Функциональные возможности:

1. Обработка данных: ввод, редактирование, удаление, хранение информации о заявках, клиентах, сотрудниках и статусах заявок.
2. Модули:
   * Управление заявками: создание, редактирование и отслеживание статусов заявок.
   * Управление клиентами: хранение информации о клиентах, контактных данных и истории обращений.
   * Управление сотрудниками: учёт сотрудников, их должностей, контактных данных и распределения задач.
   * Учёт ресурсов: управление ресурсами, используемыми для выполнения задач (например, техническими средствами, расходными материалами и пр.).

Технические особенности:

1. Конфигурация: 1С: Предприятие.
2. Используемые инструменты разработки:
   * Справочники: для хранения информации о клиентах, сотрудниках, статусах и других справочных данных.
   * Отчёты: для анализа данных по заявкам, сотрудникам, клиентам и эффективности выполнения задач.
   * Обработки и формы: для удобного взаимодействия с данными заявок, клиентов и сотрудников.

Ожидаемые результаты: Создание конфигурации, позволяющей автоматизировать управление заявками, улучшить взаимодействие с клиентами и сотрудниками, а также повысить контроль и эффективность работы службы технической поддержки.

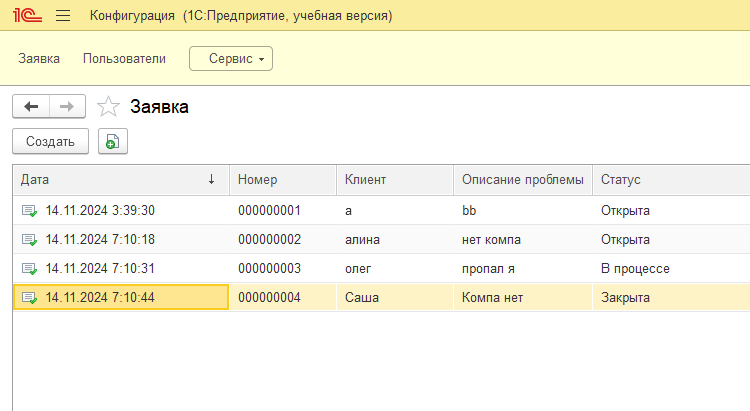


Рисунок 27 – Список заявок

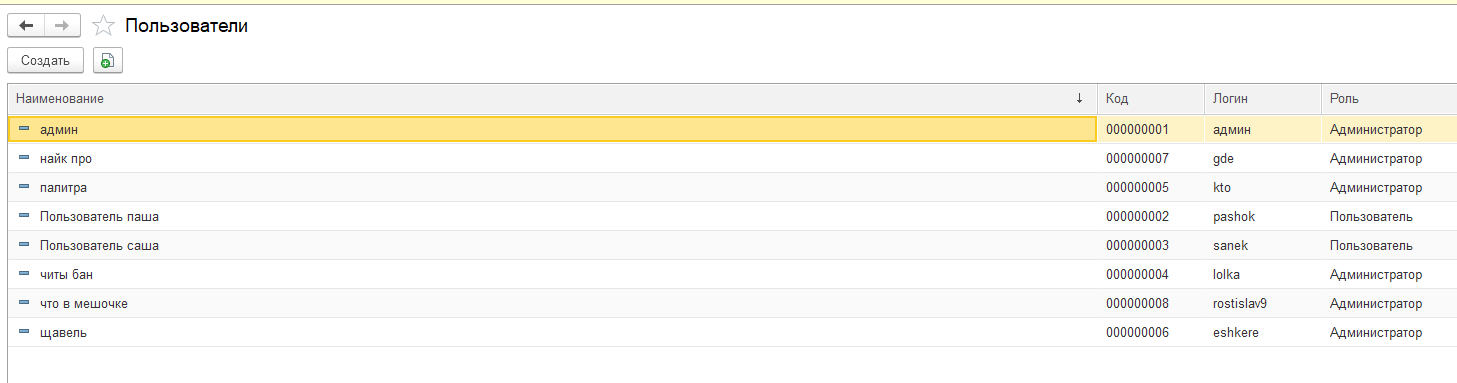


Рисунок 28 – Список пользователей

4 Разработка мобильных приложений

Разработать мобильное приложение на тему АИС.

Платформа: Android

Среда разработки: Android Studio

Подключённая база данных – MySQL на хостинге Railway

Язык программирования: Java

Графические элементы: TextView, EditText, ImageView, Button, CardView, ListView

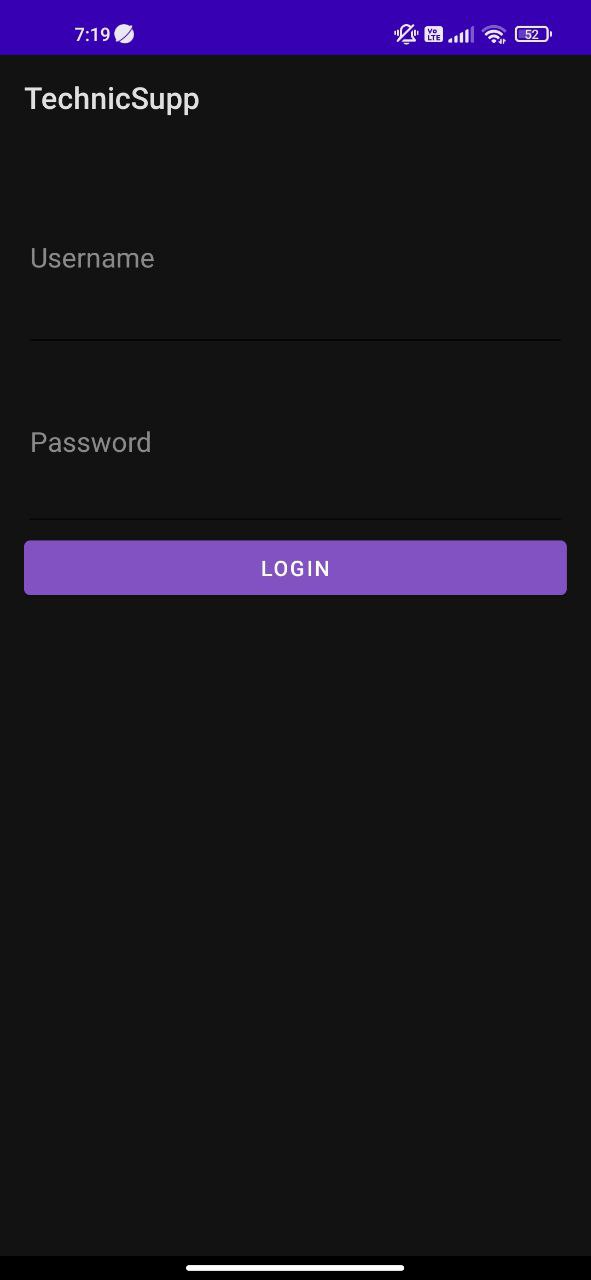


Рисунок 29 – Окно авторизации

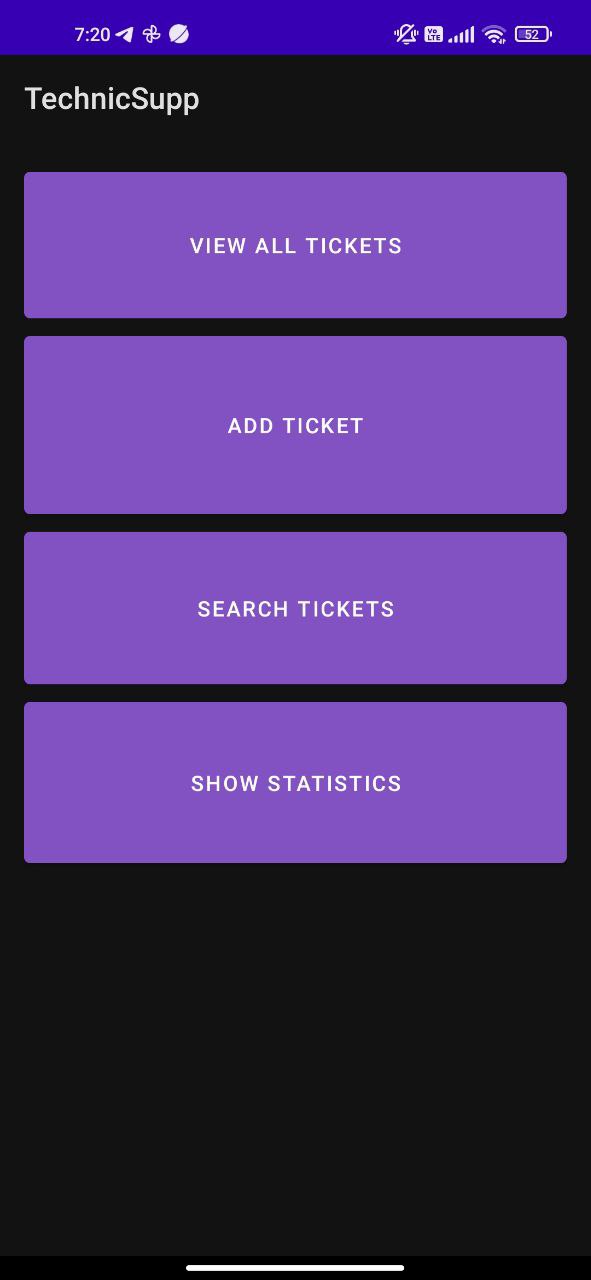


Рисунок 30 – Главное меню (Админ)



Рисунок 31 – Главное меню (Пользователь)

5 Разработка сайта

Назначение: показ итогов учебной практики

Основные функции: просмотр выполненных работ

Технологии:

Среда разработки: Visual Studio Code

Языки программирования: HTML, CSS

Хост: GitHub

Особенности дизайна: минималистичный

[ссылка](https://zamkid8.wordpress.com/%d0%bf%d0%b5%d1%80%d0%b2%d1%8b%d0%b5-9-%d0%b2%d0%b5%d1%81%d1%91%d0%bb%d1%8b%d1%85-%d0%b7%d0%b0%d0%b4%d0%b0%d1%87/)

Заключение

В ходе прохождения учебной практики по программированию удалось достичь всех поставленных целей, что позволило закрепить и расширить знания по ключевым дисциплинам. В рамках практики были успешно разработаны программные модули, создано мобильное приложение, произведено его тестирование и отладка, а также выполнена поддержка кода на всех этапах разработки. Особое внимание было уделено освоению основ программирования на платформе 1С и выполнению задач по системному программированию.

Все задачи выполнены в соответствии с предъявленными требованиями, а полученные результаты отражены на специально созданном сайте, предназначенном для наглядной демонстрации проделанной работы. Практика не только углубила теоретические знания, но и позволила приобрести ценный практический опыт, который станет основой для дальнейшего профессионального роста и успешной карьеры в сфере программирования.

Список используемой литературы

1. Белугина, С. В. Разработка программных модулей программного обеспечения для компьютерных систем. Прикладное программирование / С. В. Белугина. – 4-е изд., стер. – Санкт-Петербург: Лань, 2023. – 312 с.

2. Крыжановская, Ю. А. Основы JAVA: учебное пособие / Ю. А. Крыжановская, В. Г. Ляликова, М. М. Безрядин. – Воронеж: ВГУ, 2020. – 96 с.

3. Курбатова, И. В. Основы программирования на языке Java: учебное пособие для спо / И. В. Курбатова, А. В. Печкуров. – Санкт-Петербург: Лань, 2024. – 348 с.

4. Курбатова, И. В. Основы программирования на языке Java: учебное пособие для вузов / И. В. Курбатова, А. В. Печкуров. – Санкт-Петербург: Лань, 2024. – 348 с.

5. Куляс, О. Л. Курс программирования на ASSEMBLER: учебное пособие / О. Л. Куляс, К. А. Никитин. – Москва: СОЛОН-Пресс, 2017. – 220 с.

6. Фомичева, С. Г. Основы программирования на языке низкого уровня Assembler: учебное пособие / С. Г. Фомичева, О. С. Варига, А. А. Попкова. – Норильск: ЗГУ им. Н.М. Федоровского, 2022. – 198 с.

7. Новикова, Т. В. Системное проектирование АИС учреждения здравоохранения: учебное пособие / Т. В. Новикова. – Томск: СибГМУ, 2017. – 190 с.

8. Викторова, Н. В. 1С Бухгалтерия: учебное пособие / Н. В. Викторова. – Тюмень: ТюмГУ, 2016. – 40 с.

9. Гантц, И. С. 1С: Предприятие. Программирование для начинающих: Практикум: учебное пособие / И. С. Гантц. – Москва: РТУ МИРЭА, 2023. – 71 с.

10. Рысин, М. Л. Введение в современную Android-разработку на языке Java: учебное пособие / М. Л. Рысин. – Москва: РТУ МИРЭА, 2023 – Часть 1 – 2023. – 132 с.