

Содержание

[Введение 5](#_Toc137557439)

[1 Постановка задачи 6](#_Toc137557440)

[2 Выбор решения 7](#_Toc137557441)

[3 Описание программы 8](#_Toc137557442)

[3.1 Серверная часть 8](#_Toc137557443)

[3.2 Клиентская часть 9](#_Toc137557444)

[4 Описание способа организации пользовательского интерфейса 10](#_Toc137557445)

[5 Описание результатов работы программы 11](#_Toc137557446)

[5.1 Запуск программы и тестовая игра 11](#_Toc137557447)

[Заключение 15](#_Toc137557448)

[Список используемой литературы 16](#_Toc137557449)

[Приложение А. Исходный код 17](#_Toc137557450)

[Приложение А.1 – Исходный код Сервера 17](#_Toc137557451)

[Приложение А.2 – Исходный код Клиента 20](#_Toc137557452)

[Приложение В. UML-диаграммы приложения 28](#_Toc137557453)

[Приложение В.1 – UML-диаграмма вариантов использования 28](#_Toc137557454)

[Приложение В.2 – UML-диаграмма классов клиента 29](#_Toc137557455)

[Приложение В.3 – UML-диаграмма классов сервера 30](#_Toc137557456)

[Приложение В.4 – UML-диаграмма деятельности 30](#_Toc137557457)

[Приложение В.5 – UML-диаграмма развёртывания 31](#_Toc137557458)

[Приложение В.6 – UML-диаграмма последовательности 31](#_Toc137557459)

Введение

Java - один из наиболее популярных и мощных языков программирования, широко применяемый в различных сферах разработки программного обеспечения. Он известен своей простотой, надежностью, масштабируемостью и платформенной независимостью, что делает его идеальным выбором для создания разнообразных приложений, включая веб-приложения, мобильные приложения, настольные приложения и многое другое.

Представленная в данном курсовом проекте программа является клиент-серверным приложением, которое реализует игру “Угадай мелодию”.

1. Постановка задачи

Основной задачей данного курсового проекта является закрепление знаний и приобретение практических навыков разработки клиент-серверных приложений на языке Java. Для этого необходимо выполнить программную реализацию игры «Угадай мелодию» на языке Java в среде разработки NetBeans.

Программа работает таким образом, что серверная часть занимается приемом запросов от пользователя,тем самым реализуя основную логику игры,а клиентская часть отображает интерфейс и устанавливает взаимодействие с сервером. В проекте предусмотрен графический интерфейс и используются следующие технологии:

1. Java Collections Framework

2. Механизм обработки исключительных ситуаций

3. Java Stream API

4. Java Multithreading

5. Java Socket и Java Server Socket

1. Выбор решения

Для реализации проекта используются:

* Объектно-ориентированный язык Java;
* Библиотека JavaFX;
* Пакеты, содержащие файлы ресурсов, пользовательский интерфейс;
* Java Stream API;
* IDE NetBeans 8.2;

Основной задачей курсового проекта являлась реализация взаимодействия клиента и сервера. Для выполнения данной задачи использовались TCP сокеты и I/O операции.

Взаимодействие между клиентом и сервером должно происходить по принципу “Запрос-Ответ”. С помощью TCP сокетов осуществляется подключение клиента к серверу,а для непосредственной отправки/получения информации использовались I/O операции.

Интерфейс пользователя реализован с помощью библиотеки JavaFX и содержит в себе стандартный функционал игры “Угадай мелодию”:

* Старт игры;
* Переключение на следующую песню;
* Варианты ответа;
* Отображение количества очков;
* Отображение оставшегося времени;
* Отображение результата проверки ответа;

1. Описание программы

Программа, состоит из клиентской и серверной части. Сначала запускается сервер, который входит в состояние ожидания запроса клиента.

После чего устройствах запускается клиентская часть, которая по нажатию определенных кнопок, посылает определённый запросы к серверу.

* 1. Серверная часть

Server.java – главный файл серверного процесса, котором находится точка входа в программу, функция Main.

Метод start() запускает сервер, который регистрирует сокеты и встаёт в бесконечное ожидание подключения клиентов.

После того,как клиент подключился,обработчик клиента добавляется в список clients и создается отдельный поток для его обработки.При получении запроса сервер создаёт нить для его обработки. Метод run() инициализирует потоки ввода-вывода для обмена данными с клиентом.Затем происходит чтение данных от клиента в цикле,пока клиент не отключился.Каждая строка данных от клиента передается в метод processClientData() для обработки.

Также в серверной части есть еще метод chooseRandomSong(),который занимается выбором случайной песни для текущего раунда игры.

* 1. Клиентская часть

Client.java – главный файл пользовательского процесса, содержащий функции работы программы. Также реализует интерфейс программы.

В методе start() создается графический интерфейс и происходит установка соединения с сервером с помощью метода connectToServer().После этого создается отдельный поток для чтения данных от сервера,чтобы не блокировать главный поток интерфейса.

В методе readServerData() происходит чтение данных от сервера и обновление интерфейса на основе этих данных.Данные,полученные от сервера,могут представлять различные типы информации,например:счет,

время,результаты угадывания и список доступных песен.В зависимости от

типа данных,интерфейс обновляется соответствующим образом.

Метод playSong() отвечает за воспроизведение музыки,используя класс

MediaPlayer.

1. Описание способа организации пользовательского интерфейса

В качестве среды разработки была выбрана программа NetBeans. Программа предоставляет все средства, необходимые при разработке графического приложения.

Для реализации пользовательского интерфейса была использована библиотека JavaFX. Эта библиотека содержит более богатый и удобный набор элементов пользовательского интерфейса, обеспечивает одинаковое восприятие конечными пользователями приложений на разных платформах.

Все кнопки приложения были созданы с помощью класса Button.С помощью метода setStyle() были установлены их цвет и размер.С помощью этого же метода был установлен фон всего приложения.

С помощью класса ImageView было осуществлено отображение обложки текущей песни.

Поле таймера,очков и результата проверки ответа было реализовано с помощью класса Label.

Надпись “Угадай мелодию” была реализована с помощью метода animateGradientLabel().

Для создания синей рамки вокруг обложки песни используются классы HBox и GridPane.

1. Описание результатов работы программы

Среда разработки NetBeans предоставляет все средства, необходимые при разработке и отладке многомодульной программы.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования было выявлено и исправлено множество проблем, связанных с вводом данных, изменением дизайна клиентской части, алгоритмом программы.

Ниже продемонстрирован результат тестирования функционала программы.

* 1. Запуск программы и тестовая игра

При запуске программы открывается главное меню (рис. 1).



Рисунок – 1 Главное меню приложения

При нажатии на кнопку «Start Game» начинает проигрывается случайно выбранная песня,появляются варианты ответов и включается таймер(рис.2).



Рисунок – 2 Начало игры

После того,как пользователь ответил меняются поля «Result», «Score» и останавливается таймер(рис.3).

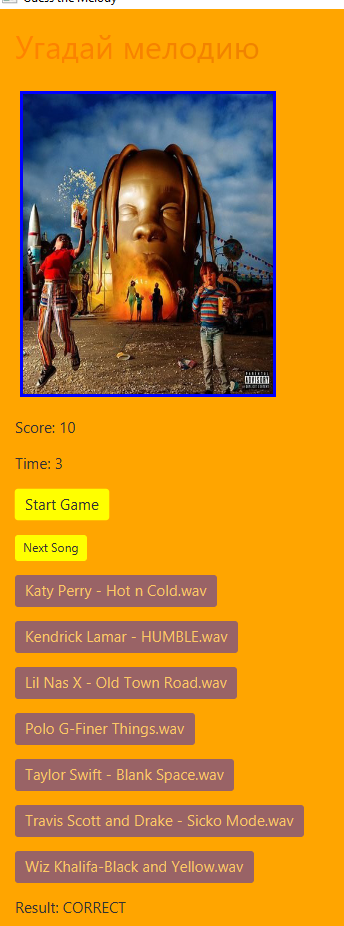


Рисунок – 3 Ответ

После того,как произошел ответ,пользователь должен нажать кнопку «Score»,после чего поменяется обложка песни,текущая проигрываемая песня и заново включится таймер(рис.4).

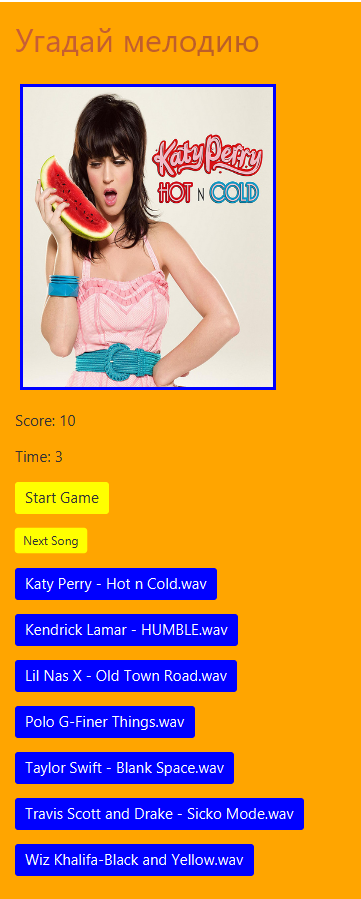


Рисунок – 4 Следующая песня

Заключение

Были реализованы следующие функции:

1.Воспроизведение текущей песни.

2.Отображение обложки текущей песни.

3.Подсчет очков.

4.Возможность ответа только в течение определенного времени.

5.Переключение на следующую песню.

6.Выбор ответа из предоставленных.

В процессе разработки были использованы следующие технологии и инструменты:

1. Язык программирования Java.

2. Библиотека JavaFXдля создания графического интерфейса.

3. Среда разработки NetBeans.

Таким образом, выполнение данной курсовой работы позволило углубиться в изучение языка программирования Java и его возможностей, а также получить практические навыки разработки клиент-серверных приложений.

Список используемой литературы

1. Портянкин, Java Swing: Эффектные пользовательские интерфейсы - Издание второе, 2011, 200 с.
2. Берд, Барри Java для чайников / Барри Берд. - М.: Диалектика / Вильямс, 2013. - 521 c.
3. Дубаков А.А. Сетевое программирование: учебное пособие / А.А. Дубаков – СП: НИУ ИТМО, 2013. – 248 с.

Приложение А. Исходный код

Приложение А.1 – Исходный код Сервера

public class Server {

private static final int SERVER\_PORT = 8888; // Порт сервера

private ServerSocket serverSocket;

private List<ClientHandler> clients;

private List<String> songAnswers;

private int number\_variants = 7;

private Integer[] songs\_variants = new Integer[number\_variants];

public static void main(String[] args) {

Server server = new Server();

server.start();

}

public Server() {

clients = new ArrayList<>();

songAnswers = new ArrayList<>();

initializeSongsAndAnswers();

}

public void start() {

try {

serverSocket = new ServerSocket(SERVER\_PORT);

System.out.println("Server started on port " + SERVER\_PORT);

while (true) {

Socket clientSocket = serverSocket.accept();

System.out.println("Client connected: " + clientSocket.getInetAddress());

ClientHandler clientHandler = new ClientHandler(clientSocket);

clients.add(clientHandler);

Thread clientThread = new Thread(clientHandler);

clientThread.start();

}

} catch (IOException e) {

e.printStackTrace();

}

}

private void initializeSongsAndAnswers() {

songAnswers.add("Katy Perry - Hot n Cold.wav");

songAnswers.add("Kendrick Lamar - HUMBLE.wav");

songAnswers.add("Lil Nas X - Old Town Road.wav");

songAnswers.add("Polo G-Finer Things.wav");

songAnswers.add("Taylor Swift - Blank Space.wav");

songAnswers.add("Travis Scott and Drake - Sicko Mode.wav");

songAnswers.add("Wiz Khalifa-Black and Yellow.wav");

}

private class ClientHandler implements Runnable {

private Socket clientSocket;

private BufferedReader reader;

private PrintWriter writer;

private int score;

private String right\_answer;

private int index\_right\_answer;

public ClientHandler(Socket clientSocket) {

this.clientSocket = clientSocket;

score = 0;

}

@Override

public void run() {

try {

reader = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

writer = new PrintWriter(clientSocket.getOutputStream(), true);

String clientData;

while ((clientData = reader.readLine()) != null) {

// Обработка данных от клиента

processClientData(clientData);

}

} catch (IOException e) {

e.printStackTrace();

} finally {

// Закрытие сокета и потоков при отключении клиента

try {

reader.close();

writer.close();

clientSocket.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

private void chooseRandomSong() {

Random random = new Random();

//Случайный выбор позиции правильного ответа

for (int i = 0; i < number\_variants; i++)

{

songs\_variants[i] = i;

}

for (int i = number\_variants; i < songAnswers.size(); i++)

{

int j = random.nextInt(i);

if (j < number\_variants)

songs\_variants[j] = i;

}

index\_right\_answer = random.nextInt(7);

right\_answer = (String) songAnswers.toArray()[songs\_variants[index\_right\_answer]];

}

private void processClientData(String clientData) {

Random random = new Random();

// Логика обработки данных от клиента и отправки данных другим клиентам

// Здесь вы можете проверять время ответа, обновлять статистику и т.д.

// Пример:

if (clientData.equals("START")) {

// Логика начала игры

// Отправить данные клиенту

score = 0;

chooseRandomSong();

//writer.println("SONG\_RIGHT: " + right\_answer);

writer.println("SONGS: " + (String) songAnswers.toArray()[songs\_variants[0]] + "," + (String) songAnswers.toArray()[songs\_variants[1]]

+ "," + (String) songAnswers.toArray()[songs\_variants[2]] + "," + (String) songAnswers.toArray()[songs\_variants[3]] + "," + (String) songAnswers.toArray()[songs\_variants[4]] + "," + (String) songAnswers.toArray()[songs\_variants[5]] + "," + (String) songAnswers.toArray()[songs\_variants[6]]);

writer.println("SCORE: " + Integer. toString(score));

writer.println("PLAY\_SONG: " + (String) songAnswers.toArray()[songs\_variants[index\_right\_answer]]);

} else if (clientData.equals("NEXT")) {

// Логика перехода к следующей песне

// Отправить данные клиенту

chooseRandomSong();

writer.println("SONGS: " + (String) songAnswers.toArray()[songs\_variants[0]] + "," + (String) songAnswers.toArray()[songs\_variants[1]]

+ "," + (String) songAnswers.toArray()[songs\_variants[2]] + "," + (String) songAnswers.toArray()[songs\_variants[3]] + "," + (String) songAnswers.toArray()[songs\_variants[4]] + "," + (String) songAnswers.toArray()[songs\_variants[5]] + "," + (String) songAnswers.toArray()[songs\_variants[6]]);

writer.println("SCORE: " + Integer. toString(score));

writer.println("PLAY\_SONG: " + (String) songAnswers.toArray()[songs\_variants[index\_right\_answer]]);

} else if (clientData.startsWith("ANSWER: ")) {

// Логика проверки ответа клиента

String answer = clientData.substring(8);

if (answer.equals(right\_answer)){

score = score + 10;

writer.println("CORRECT");

writer.println("SCORE: " + Integer. toString(score));

}

else{

writer.println("ERROR");

if (score > 0){

score = score - 10;

}

writer.println("SCORE: " + Integer. toString(score));

}

// Проверить ответ и отправить данные клиенту

}

else if (clientData.startsWith("TIMEOUT")) {

if (score > 0){

score = score - 10;

}

writer.println("TIMEOUT");

writer.println("SCORE: " + Integer. toString(score));

}

}

Приложение А.2 – Исходный код Клиента

public class Client extends Application {

private static final String SERVER\_IP = "127.0.0.1"; // IP-адрес сервера

private static final int SERVER\_PORT = 8888; // Порт сервера

private static String SONG\_PATH = "songs\\";

ImageView imageView = new ImageView();

String cssDefault = "-fx-border-color: blue;\n" + "-fx-border-insets: 5;\n"

+ "-fx-border-width: 3;\n";

private Socket socket;

private BufferedReader reader;

private PrintWriter writer;

private Label guessLabel;

private Label scoreLabel;

private Label timeLabel;

private Label resultLabel;

private int time\_sec;

private String lastPlayedSong;

private Button song1Button;

private Button song2Button;

private Button song3Button;

private Button song4Button;

private Button song5Button;

private Button song6Button;

private Button song7Button;

private Button startButton;

private Button nextButton;

// private Button submitButton;

private Timer myTimer;

private MediaPlayer mediaPlayer;

private static final Color[] COLORS = {

Color.RED, Color.BLUE, Color.YELLOW, Color.GREEN,

Color.BLUE, Color.PURPLE, Color.YELLOW

};

private static final int TRANSITION\_DURATION = 2000;

public static void main(String[] args) {

launch(args);

}

@Override

public void start(Stage primaryStage) {

Image image = new Image("file:K:/kursjava67567567/Kurs/songs/Hello.jpg");

final HBox pictureRegion = new HBox();

GridPane gridpane = new GridPane();

pictureRegion.setStyle(cssDefault);

pictureRegion.getChildren().add(imageView);

gridpane.add(pictureRegion, 1, 1);

imageView.setImage(image);

imageView.setFitWidth(250);

imageView.setFitHeight(300);

imageView.setStyle(cssDefault);

guessLabel = new Label("Угадай мелодию");

guessLabel.setStyle("-fx-font-size: 32px;");

scoreLabel = new Label("Score: 0");

scoreLabel.setStyle("-fx-font-size: 15px;");

scoreLabel.setLayoutY(200);

scoreLabel.setLayoutX(200);

timeLabel = new Label("Time: 0");

timeLabel.setStyle("-fx-font-size: 15px;");

resultLabel = new Label("Result: ");

resultLabel.setStyle("-fx-font-size: 15px;");

startButton = new Button("Start Game");

startButton.setOnAction(event -> handleStartGame());

startButton.setStyle("-fx-background-color: yellow;-fx-font-size: 15px;");

nextButton = new Button("Next Song");

nextButton.setOnAction(event -> handleNextSong());

nextButton.setStyle("-fx-background-color: yellow;");

// submitButton = new Button("Submit Answer");

// submitButton.setOnAction(event -> handleSubmitAnswer());

song1Button = new Button("song1");

song1Button.setStyle("-fx-background-color: blue; -fx-text-fill: white;-fx-font-size: 15px;");

//song1Button.setBackground(Color.BLUE);

song1Button.setOnAction(event -> handleSong1Answer());

song1Button.setDisable(true);

song2Button = new Button("song2");

song2Button.setStyle("-fx-background-color: blue; -fx-text-fill: white;-fx-font-size: 15px;");

song2Button.setOnAction(event -> handleSong2Answer());

song2Button.setDisable(true);

song3Button = new Button("song3");

song3Button.setStyle("-fx-background-color: blue; -fx-text-fill: white;-fx-font-size: 15px;");

song3Button.setOnAction(event -> handleSong3Answer());

song3Button.setDisable(true);

song4Button = new Button("song4");

song4Button.setStyle("-fx-background-color: blue; -fx-text-fill: white;-fx-font-size: 15px;");

song4Button.setOnAction(event -> handleSong4Answer());

song4Button.setDisable(true);

song5Button = new Button("song5");

song5Button.setStyle("-fx-background-color: blue; -fx-text-fill: white;-fx-font-size: 15px;");

song5Button.setOnAction(event -> handleSong5Answer());

song5Button.setDisable(true);

song6Button = new Button("song6");

song6Button.setStyle("-fx-background-color: blue; -fx-text-fill: white;-fx-font-size: 15px;");

song6Button.setOnAction(event -> handleSong6Answer());

song6Button.setDisable(true);

song7Button = new Button("song7");

song7Button.setStyle("-fx-background-color: blue; -fx-text-fill: white;-fx-font-size: 15px;");

song7Button.setOnAction(event -> handleSong7Answer());

song7Button.setDisable(true);

VBox root = new VBox(15);

root.getChildren().add(guessLabel);

root.getChildren().add(gridpane);

root.setPadding(new Insets(15));

root.getChildren().addAll(scoreLabel, timeLabel, startButton, nextButton, song1Button, song2Button, song3Button,song4Button,song5Button,song6Button,song7Button, resultLabel);

root.setStyle("-fx-background-color: orange;");

Scene scene = new Scene(root, 285, 800);

// scene.setBackgroundColor(Color.parseColor("#ffffff"));

primaryStage.setScene(scene);

primaryStage.setTitle("Guess the Melody");

primaryStage.show();

animateGradientLabel(guessLabel);

connectToServer();

}

private void animateGradientLabel(Label label) {

// Создание FadeTransition для плавного перехода между цветами

FadeTransition fadeTransition = new FadeTransition(Duration.millis(TRANSITION\_DURATION), label);

fadeTransition.setFromValue(0.0);

fadeTransition.setToValue(1.0);

fadeTransition.setCycleCount(Animation.INDEFINITE);

fadeTransition.setAutoReverse(true);

// Создание Timeline для изменения цвета надписи

Timeline colorTimeline = new Timeline();

KeyFrame[] keyFrames = new KeyFrame[COLORS.length];

for (int i = 0; i < COLORS.length; i++) {

Color color = COLORS[i];

Stop[] stops = {

new Stop(0, color),

new Stop(1, color.darker()) // Темнее значение цвета для создания градиента

};

LinearGradient gradient = new LinearGradient(0, 0, 1, 0, true, CycleMethod.NO\_CYCLE, stops);

keyFrames[i] = new KeyFrame(Duration.millis(i \* TRANSITION\_DURATION), event -> {

Paint paint = gradient;

label.setTextFill(paint);

});

}

colorTimeline.getKeyFrames().addAll(keyFrames);

colorTimeline.setCycleCount(Animation.INDEFINITE);

// Запуск анимации

fadeTransition.play();

colorTimeline.play();

}

private void connectToServer() {

try {

socket = new Socket(SERVER\_IP, SERVER\_PORT);

reader = new BufferedReader(new InputStreamReader(socket.getInputStream()));

writer = new PrintWriter(socket.getOutputStream(), true);

Thread readerThread = new Thread(this::readServerData);

readerThread.start();

} catch (IOException e) {

e.printStackTrace();

Platform.exit();

}

}

private void readServerData() {

try {

String serverData;

while ((serverData = reader.readLine()) != null) {

final String data = serverData; // Создаем финальную копию serverData

Platform.runLater(() -> handleServerData(data));

}

} catch (IOException e) {

e.printStackTrace();

}

}

private String getSongFile(String songName) {

// Получение пути к файлу песни по ее имени из вашей локальной директории

// Здесь нужно реализовать логику поиска пути к файлу песни на вашей системе

return (SONG\_PATH + songName);

}

private void playSong(String songFile) {

if (mediaPlayer != null) {

mediaPlayer.stop();

}

Media media = new Media(new File(songFile).toURI().toString());

mediaPlayer = new MediaPlayer(media);

mediaPlayer.play();

}

private void handleServerData(String serverData) {

// Обработка данных от сервера и обновление интерфейса

// Пример:

if ((serverData.startsWith("CORRECT")) || (serverData.startsWith("ERROR")) || (serverData.startsWith("TIMEOUT"))) {

Platform.runLater(() -> resultLabel.setText("Result: " + serverData));

}

else if (serverData.startsWith("SCORE: ")) {

String scoreValue = serverData.substring(7);

Platform.runLater(() -> scoreLabel.setText("Score: " + scoreValue));

} else if (serverData.startsWith("TIME: ")) {

String timeValue = serverData.substring(6);

Platform.runLater(() -> timeLabel.setText("Time: " + timeValue));

} else if (serverData.startsWith("SONGS: ")) {

String[] parts = (serverData.substring(7)).split(",");

song1Button.setText(parts[0]);

song2Button.setText(parts[1]);

song3Button.setText(parts[2]);

song4Button.setText(parts[3]);

song5Button.setText(parts[4]);

song6Button.setText(parts[5]);

song7Button.setText(parts[6]);

// Обновить название песни в интерфейсе

} else if (serverData.startsWith("PLAY\_SONG: ")) {

String songName = serverData.substring(11);

if (!songName.equals(lastPlayedSong)) {

lastPlayedSong = songName;

changeImage(songName);

myTimer = new Timer();

time\_sec = 5;

playSong(getSongFile(songName));

myTimer.schedule(new StopTask(), 0, 1000);

song1Button.setDisable(false);

song2Button.setDisable(false);

song3Button.setDisable(false);

song4Button.setDisable(false);

song5Button.setDisable(false);

song6Button.setDisable(false);

song7Button.setDisable(false);

}

else {

// Песня уже была воспроизведена, перейти к следующей

//writer.println("NEXT");

}

}

}

class StopTask extends TimerTask {

public void run() {

time\_sec --;

String timeValue = Integer.toString(time\_sec);

Platform.runLater(() -> timeLabel.setText("Time: " + timeValue));

if (time\_sec == 0)

{

writer.println("TIMEOUT");

song1Button.setDisable(true);

song2Button.setDisable(true);

song3Button.setDisable(true);

song4Button.setDisable(true);

song5Button.setDisable(true);

song6Button.setDisable(true);

song7Button.setDisable(true);

myTimer.cancel();

}

}

}

private void changeImage(String songId) {

Image image;

//Rectangle border = new Rectangle(250, 300);

// border.setFill(Color.BLUE);

// border.setStroke(Color.BLACK);

//border.setStrokeWidth(4);

switch (songId) {

case "Katy Perry - Hot n Cold.wav":

image = new Image("file:K:/kursjava67567567/Kurs/songs/Katy.jpg");

break;

case "Kendrick Lamar - HUMBLE.wav":

image = new Image("file:K:/kursjava67567567/Kurs/songs/Kendrick.jpg");

break;

case "Lil Nas X - Old Town Road.wav":

image = new Image("file:K:/kursjava67567567/Kurs/songs/Lil Nas X.jpg");

break;

case "Polo G-Finer Things.wav":

image = new Image("file:K:/kursjava67567567/Kurs/songs/Polo G.jpg");

break;

case "Taylor Swift - Blank Space.wav":

image = new Image("file:K:/kursjava67567567/Kurs/songs/Taylor Swift.png");

break;

case "Travis Scott and Drake - Sicko Mode.wav":

image = new Image("file:K:/kursjava67567567/Kurs/songs/Travis Scott.jpg");

break;

case "Wiz Khalifa-Black and Yellow.wav":

image = new Image("file:K:/kursjava67567567/Kurs/songs/Wiz Khalifa.jpg");

break;

default:

// Если передан неправильный идентификатор песни, установите

// изображение по умолчанию или выполните другие действия

image = new Image("file:K:/kursjava67567567/Kurs/songs/LeBron.jpg");

break;

}

imageView.setImage(image);

// imageView.setClip(border);

}

private void handleStartGame() {

writer.println("START");

}

private void handleNextSong() {

resultLabel.setText("");

writer.println("NEXT");

}

// private void handleSubmitAnswer() {

// myTimer.cancel();

// String answer = answerField.getText();

// writer.println("ANSWER: " + answer);

// }

private void handleSong1Answer() {

myTimer.cancel();

String answer = song1Button.getText();

writer.println("ANSWER: " + answer);

song1Button.setDisable(true);

song2Button.setDisable(true);

song3Button.setDisable(true);

song4Button.setDisable(true);

song5Button.setDisable(true);

song6Button.setDisable(true);

song7Button.setDisable(true);

}

private void handleSong2Answer() {

myTimer.cancel();

String answer = song2Button.getText();

writer.println("ANSWER: " + answer);

song1Button.setDisable(true);

song2Button.setDisable(true);

song3Button.setDisable(true);

song4Button.setDisable(true);

song5Button.setDisable(true);

song6Button.setDisable(true);

song7Button.setDisable(true);

}

private void handleSong3Answer() {

myTimer.cancel();

String answer = song3Button.getText();

writer.println("ANSWER: " + answer);

song1Button.setDisable(true);

song2Button.setDisable(true);

song3Button.setDisable(true);

song4Button.setDisable(true);

song5Button.setDisable(true);

song6Button.setDisable(true);

song7Button.setDisable(true);;

}

private void handleSong4Answer() {

myTimer.cancel();

String answer = song4Button.getText();

writer.println("ANSWER: " + answer);

song1Button.setDisable(true);

song2Button.setDisable(true);

song3Button.setDisable(true);

song4Button.setDisable(true);

song5Button.setDisable(true);

song6Button.setDisable(true);

song7Button.setDisable(true);

}

private void handleSong5Answer() {

myTimer.cancel();

String answer = song5Button.getText();

writer.println("ANSWER: " + answer);

song1Button.setDisable(true);

song2Button.setDisable(true);

song3Button.setDisable(true);

song4Button.setDisable(true);

song5Button.setDisable(true);

song6Button.setDisable(true);

song7Button.setDisable(true);

}

private void handleSong6Answer() {

myTimer.cancel();

String answer = song6Button.getText();

writer.println("ANSWER: " + answer);

song1Button.setDisable(true);

song2Button.setDisable(true);

song3Button.setDisable(true);

song4Button.setDisable(true);

song5Button.setDisable(true);

song6Button.setDisable(true);

song7Button.setDisable(true);

}

private void handleSong7Answer() {

myTimer.cancel();

String answer = song7Button.getText();

writer.println("ANSWER: " + answer);

song1Button.setDisable(true);

song2Button.setDisable(true);

song3Button.setDisable(true);

song4Button.setDisable(true);

song5Button.setDisable(true);

song6Button.setDisable(true);

song7Button.setDisable(true);

}

}

Приложение В. UML-диаграммы приложения

Приложение В.1 – UML-диаграмма вариантов использования

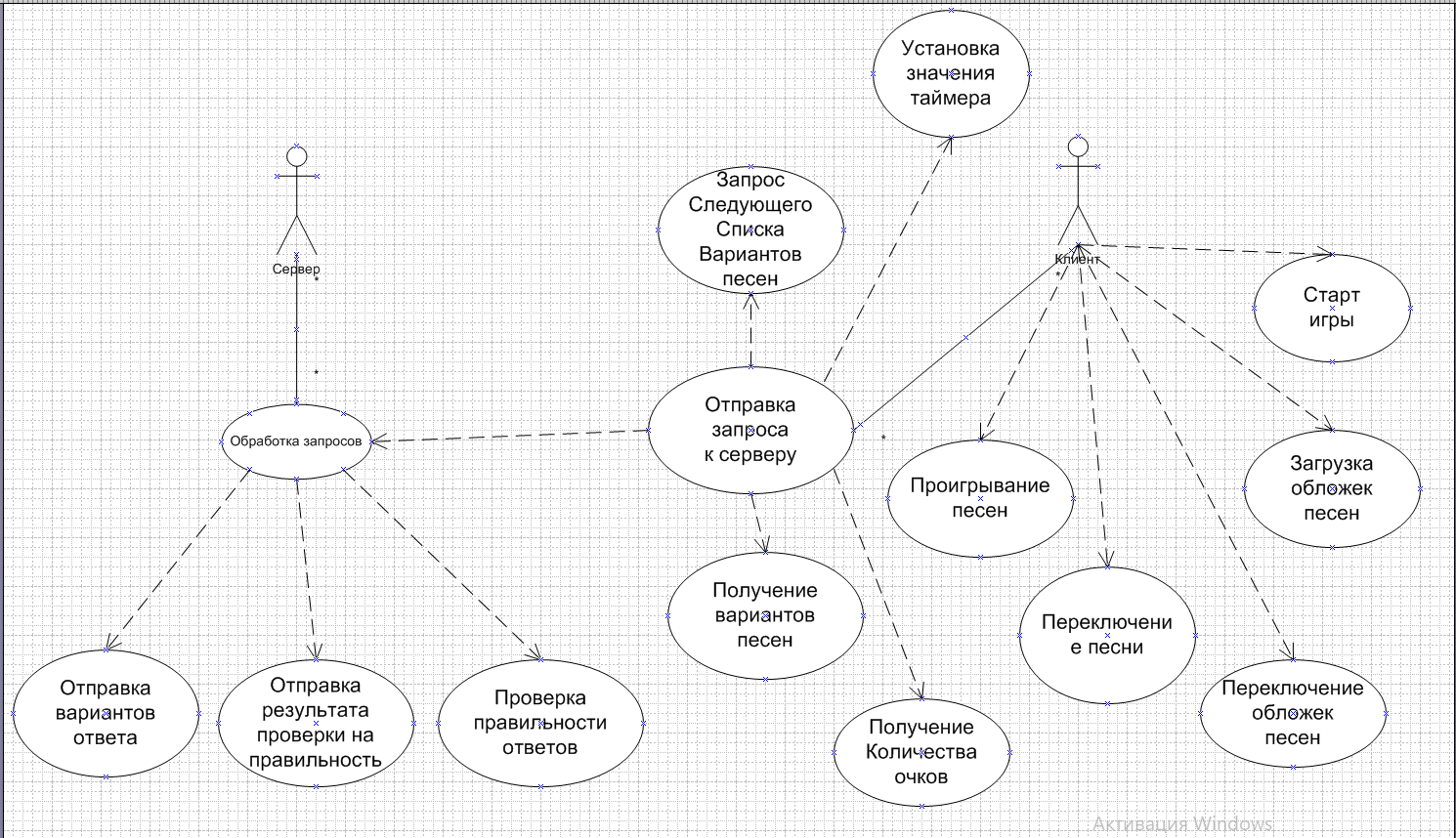


Рисунок 19 – UML-Диаграмма вариантов использования

Приложение В.2 – UML-диаграмма классов клиента



Рисунок 20 – UML-диаграмма классов клиента

Приложение В.3 – UML-диаграмма классов сервера

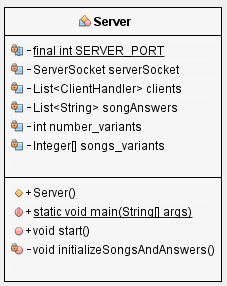


Рисунок 22 – UML-диаграмма классов сервера

Приложение В.4 – UML-диаграмма деятельности

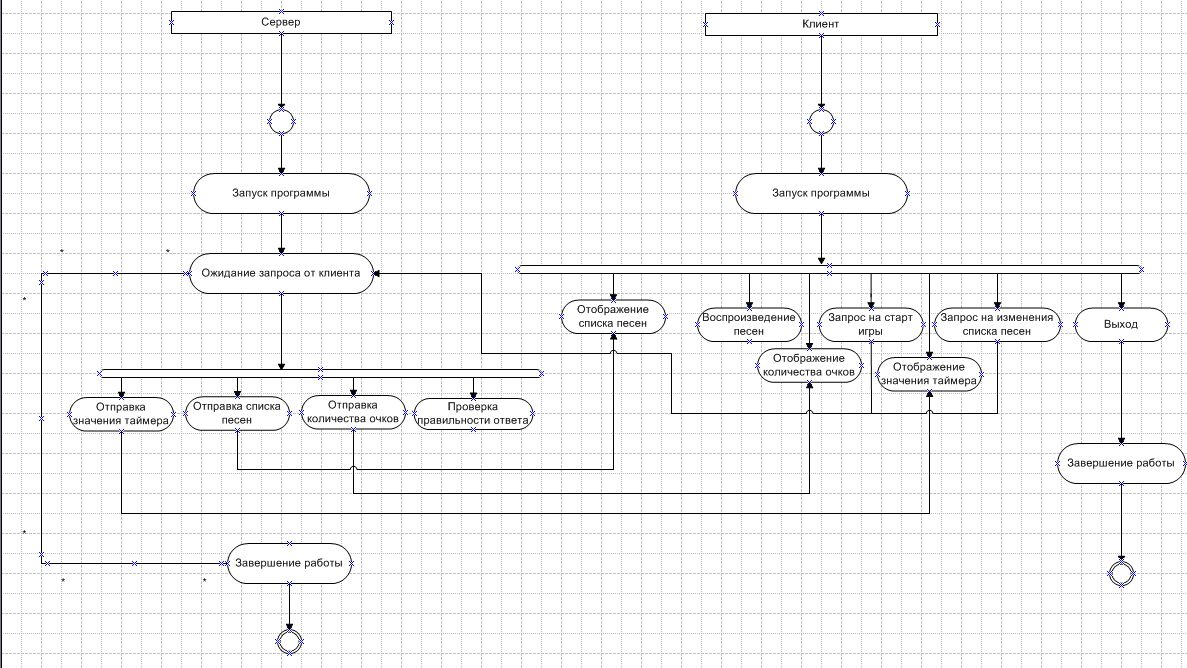


Рисунок 24 – UML-диаграмма деятельности

Приложение В.5 – UML-диаграмма развёртывания



Рисунок 25 – UML-диаграмма развёртывания

Приложение В.6 – UML-диаграмма последовательности



Рисунок 26 – UML-диаграмма последовательности