**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 6](#_Toc153930429)

[**ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ** 8](#_Toc153930430)

[*2.1 Постановка задачи* 8](#_Toc153930431)

[*2.2 Общие требования* 9](#_Toc153930432)

[**3 Реализация программного обеспечения** 18](#_Toc153930433)

[**РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ** 23](#_Toc153930434)

[**ЗАКЛЮЧЕНИЕ** 26](#_Toc153930435)

[**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ** 27](#_Toc153930436)

[**Приложение A** 28](#_Toc153930437)

# **ВВЕДЕНИЕ**

Игра, которую мы рассматриваем, представляет собой феноменальное творение, основанное на компьютерной разработке и программировании. Это увлекательное приложение, созданное с использованием кода и технологий, переносящее игрока в мир виртуальной тюрьмы в стиле тамагочи. Рассмотрим подробнее, как этот проект объединяет в себе элементы программирования и игрового дизайна для создания захватывающего игрового опыта, уникального в своем роде.

Игры - это удивительный формат развлечений, способный объединить людей разных возрастов и культур. Они предлагают не только возможность отдохнуть и развлечься, но и стать платформой для обучения, социализации и развития различных навыков.

Игровая индустрия постоянно эволюционирует, предлагая разнообразные жанры от стратегий и приключений до симуляторов и головоломок. Важной частью игрового процесса является их способность к инновациям и прорывам в технологиях, что позволяет создавать увлекательные миры с потрясающими графикой и физикой.

Одним из заметных аспектов современных игр является их обучающий потенциал. Они способствуют развитию логического мышления, принятию решений в условиях ограниченного времени, а также развивают социальные навыки через мультиплеерные режимы игр.

Игры также стали площадкой для культурного выражения, позволяя разработчикам передавать определенные сообщения или создавать виртуальные миры, отражающие различные аспекты реальной жизни.

В целом, игры имеют огромное влияние на нашу культуру и способны оказывать положительное воздействие на наше мышление, восприятие и способности, предлагая уникальный способ взаимодействия и развлечения.

**ОБЗОР И ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ**

В игре "Тюрьма в стиле Тамагочи" вы становитесь участником уникального симулятора тюремной жизни, где каждое ваше действие и решение влияют на параметры вашего заключенного.

Параметры вашего заключенного:

1. Счастье: Отражает уровень удовлетворенности вашего персонажа.

2. Уровень здоровья: Определяет физическое состояние заключенного.

3. Деньги: Используются для покупки продуктов, влияющих на состояние.

Продукты и их влияние на параметры:

1. Брокколи: Повышает уровень здоровья, но уменьшает счастье вашего персонажа.

2. Пицца: Повышает оба параметра — счастье и уровень здоровья.

3. Сигарета: Увеличивает счастье, но ухудшает уровень здоровья.

Дополнительные функции:

1. Игровой кликер: Позволяет зарабатывать деньги для покупки продуктов.

2. Радио: Увеличивает уровень счастья вашего персонажа.

3. Сохранения: Возможность сохранения данных о заключенном в формате Excel.

4. Изменение языка: Пользователи могут выбирать язык игры по своему усмотрению.

5. Логирование: Ведется контроль работы программы и проверка на ошибки

Эта захватывающая игра предоставляет вам уникальную возможность исследовать динамику тюремной жизни, принимать стратегические решения и управлять параметрами вашего персонажа, создавая захватывающий и интерактивный игровой опыт, который вдохновляет на размышления о влиянии наших действий на наше благополучие и окружающих нас людей.

# **ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

Целью данной курсовой работы является разработка программного продукта для автоматизации работы ресторана. Программа должна предоставлять функционал для учёта продукции и разбиение продукции на категории, обеспечивать систему авторизации сотрудников и учёт заказов клиента ресторана.

## *2.1 Постановка задачи*

Тема Курсового проекта - Тюрьма

Ответвления темы для реализации:  
Архитектура приложения; Учёт здоровья и уровня счастья заключенных;

*Основные сущности предметной области:*

**Еда и источники счастья:** Это основные объекты поднятия здоровья и уровня счастья. Каждый продукт имеет свои характеристики, такие как тип, цена и характеристики.

**Заключенный:** Ваш игровой персонаж за характеристиками которого вы должны следить.

**Игровой кликер:** Позволяет зарабатывать деньги для поднятия уровня счастья и здоровья заключенного, методом покупки продуктов.

Поведение программы:

* Возможность выбора сохранения: Система должна сохранять прогресс в бинарные файлы.
* Отслеживание состояния персонажа: Приложение должно позволять обрабатывать и отслеживать состояние персонажа.
* Управление состоянием персонажа: Приложение должно позволять манипулировать состоянием персонажа
* Возможность ведения контроля: Ведется контроль работы программы и проверка на ошибки

## *2.2 Общие требования*

1. Разработать интуитивно понятный и легко используемый интерфейс программы.
2. Обеспечить надежность и безопасность хранения данных о персонаже
3. Обеспечить поддержку различных видов изменений характеристик персонажа
4. Провести тестирование программного продукта на предмет выявления и устранения возможных ошибок.

# 

**1. Требования к игре:**

* Реализовать удобный интерфейс.
* Реализовать возможность добавления новых персонажей.
* Реализовать функционал редактирования состояния персонажа.

1. **Внутренние функции игры:**

* Разработать систему, позволяющую следить за состоянием персонажа.
* Реализовать систему повышения уровня здоровья и уровня счастья.
* Реализовать функционал заработка денег для покупки внутриигровых предметов.

**3. Функции Логирования и сохранения;**

* Разработать систему сохранения персонажа.
* Обеспечить возможность контроля работы программы и проверки на ошибки.
* Реализовать возможность изменения языка.

**РЕАЛИЗАЦИЯ**

***Выбранные технологии: Java, JavaAwt***

**Java** - Java – широко используемый язык программирования для написания интернет-приложений. Язык Java широко использовался на протяжении более двух десятилетий. Миллионы приложений Java используются и сегодня. Java – это многоплатформенный, объектно-ориентированный и сетецентрический язык, который сам по себе может использоваться как платформа. Это быстрый, безопасный и надежный язык программирования для всего: от мобильных приложений и корпоративного ПО до приложений для работы с большими данными и серверных технологий.

**Java.awt** - Abstract Window Toolkit (AWT) — это исходная платформо-независимая оконная библиотека графического интерфейса (Widget toolkit) языка Java. Сейчас AWT является частью Java Foundation Classes (JFC) — стандартного API для реализации графического интерфейса в Java-программе.

**Git** - это система контроля версий (Version Control System, VCS), которая используется для отслеживания изменений в исходном коде программного обеспечения и управления совместной работой нескольких разработчиков. Разработан Linus Torvalds в 2005 году, Git стал одним из самых популярных инструментов в области разработки программного обеспечения.

***Основные черты Git:***

**Децентрализованная система:** Каждый участник проекта имеет полную копию репозитория. Это позволяет работать независимо, не завися от постоянного подключения к центральному серверу.

**Эффективность и быстродействие:** Git предоставляет эффективные механизмы для слияния изменений, а также хранит данные в специально оптимизированной структуре, что обеспечивает высокую скорость работы.

История изменений: Git сохраняет полную историю изменений в проекте. Это позволяет в любой момент вернуться к предыдущей версии кода или проанализировать, кто и когда внес какие изменения.

**Ветвление и слияние (Branching and Merging):** Работа в ветках позволяет разработчикам изолировать свою работу и затем сливать её обратно в основную ветку.

**Отслеживание изменений:** Git автоматически обнаруживает и отслеживает изменения в файлах, а не просто хранит копии файлов на разных этапах времени.

**Поддержка небинарных файлов:** Git не только хорошо работает с исходным кодом, но и способен управлять изменениями в любых файлах.

На момент написания отчета по курсовой работе, в нашем репозитории GitHub мы имеем 14 веток, и в общей сложности 36 коммитов. Так же было выполнено 15 pull requests.

**Общее обоснование:**

Выбор технологий и языка проекта был основан на их плюсах

Java:

Переносимость: Язык Java предлагает возможность запуска программ на различных платформах без изменения кода. Это обеспечивает универсальность проекта и доступность для широкой аудитории.

Большое сообщество разработчиков: Java имеет огромное активное сообщество разработчиков, что обеспечивает доступ к различным ресурсам, библиотекам и поддержке, упрощая процесс разработки.

Безопасность: Встроенные механизмы безопасности Java обеспечивают защиту от различных угроз, что критично для приложений, работающих с данными пользователя.

Java AWT и JPanel:

Простота и быстрота разработки: Использование AWT и JPanel для построения пользовательского интерфейса обеспечивает базовые инструменты без необходимости привлечения дополнительных библиотек, что ускоряет процесс разработки.

Модульность и организация: JPanel позволяет структурировать и управлять компонентами интерфейса, что обеспечивает гибкость в организации элементов игры, таких как меню, управляющие панели и другие.

Легкость использования: Для простых игровых интерфейсов, где не требуется сложная графика или анимация, AWT и JPanel предоставляют достаточные инструменты для быстрой реализации и отображения элементов интерфейса.

Выбор Java, Java AWT и JPanel обоснован их универсальностью, простотой использования и доступностью. AWT и JPanel предоставляют базовые, но эффективные инструменты для создания интерфейса, что может быть вполне достаточно для простой игры с упором на игровой процесс и логику, без излишней сложности графики и анимаций.

Из основных классов что можно представить на данном этапе можно предложить:

**GameLogic(основной класс с полной реализацией логики программы)**

package com.example;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Paths;

import java.util.ArrayList;

import java.util.Timer;

import java.util.TimerTask;

import org.json.JSONObject;

import javax.swing.JOptionPane;

import java.text.SimpleDateFormat;

import java.util.Date;

import org.apache.poi.ss.usermodel.\*;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

import java.io.FileOutputStream;

import java.io.FileInputStream;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

import java.util.Scanner;

import javax.sound.sampled.\*;

public class GameLogic {

private static final Scanner scanner = new Scanner(System.in);

private static final Logger logger = LogManager.getLogger(GameLogic.class);

public static void main(String[] args) {

configureLogger();

logger.info("Application started");

Food cigarette = new Food();

cigarette.cigarette();

cigarette.cigaretteParameter();

Food soup = new Food();

soup.soup();

soup.soupParameter();

Food chifir = new Food();

chifir.chifir();

chifir.chifirParameter();

Games clicker = new Games();

clicker.clicker();

clicker.clickerParameter();

Games radio = new Games();

radio.radio();

radio.radioParameter();

MenuButtons homeB = new MenuButtons();

homeB.homeParameter();

MenuButtons cutlB = new MenuButtons();

cutlB.cutleryParameter();

MenuButtons gameB = new MenuButtons();

gameB.gameParameter();

MenuButtons restaB = new MenuButtons();

restaB.restartParameter();

ArrayList<Food> foodCanvas = new ArrayList<>();

foodCanvas.add(cigarette);

foodCanvas.add(soup);

foodCanvas.add(chifir);

ArrayList<Games> gamesCanvas = new ArrayList<>();

gamesCanvas.add(clicker);

gamesCanvas.add(radio);

Pet pet = new Pet();

String filePath = "src/main/resources/Gio.wav";

playBackgroundMusic(filePath);

Runtime.getRuntime().addShutdownHook(new Thread(() -> {

// Код, выполняемый при закрытии программы

writeToExcelOnExit(pet.getName());

}));

int choice = JOptionPane.showOptionDialog(

null,

"Choose or create a save:",

"Start Game",

JOptionPane.YES\_NO\_OPTION,

JOptionPane.QUESTION\_MESSAGE,

null,

new Object[]{"Load Save", "New Save"},

"Load Save"

);

String selectedSave;

if (choice == JOptionPane.YES\_OPTION) {

// Load an existing save

selectedSave = chooseSave();

if (selectedSave != null) {

Var.petSave = new File("saves", selectedSave + ".json");

reloadSave(pet);

} else {

// Handle the case where no save is selected or created

JOptionPane.showMessageDialog(null, "No save selected or created. Exiting.");

return;

}

} else {

// Create a new save

selectedSave = createNewSave();

if (selectedSave != null) {

Var.petSave = new File("saves", selectedSave + ".json");

// Call createNewCharacter to initialize the pet

createNewCharacter(pet);

} else {

// Handle the case where no new save is created

JOptionPane.showMessageDialog(null, "No new save created. Exiting.");

return;

}

}

if (selectedSave != null) {

Var.petSave = new File("saves", selectedSave + ".json");

reloadSave(pet);

} else {

// Handle the case where no save is selected or created

JOptionPane.showMessageDialog(null, "No save selected or created. Exiting.");

return;

}

// TODO: Bedingung anpassen

/\* if(pet.name == ""){

pet.startValues();

}

else {

reloadSave(pet);

}

\*/

CanvasLayer c = new CanvasLayer(foodCanvas, gamesCanvas, pet);

c.canvas.addMouseListener(new MouseAdapter() {

public void mouseClicked(MouseEvent e) {

if (Var.switchScreen == Display.FOOD\_SCREEN){

if (foodCollide(cigarette, e)) {

if(pet.lessMoney(cigarette)==true)

{

pet.moreHappiness(cigarette);

pet.moreHealth(cigarette);

logger.info("pressed cigarette");

Client.sendMessage(pet.name+": покурил");

}

}

if (foodCollide(soup, e)) {

if(pet.lessMoney(soup)==true)

{

pet.moreHappiness(soup);

pet.moreHealth(soup);

logger.info("pressed soup");

Client.sendMessage(pet.name+": поел пиццы");

}

}

if (foodCollide(chifir, e)) {

if(pet.lessMoney(chifir)==true)

{

pet.moreHappiness(chifir);

pet.moreHealth(chifir);

logger.info("pressed chifir");

Client.sendMessage(pet.name+": поел брокколи");

}

}

}

if (Var.switchScreen == Display.GAME\_SCREEN){

if(gameCollide(clicker, e)){

pet.moreHappinessGame(clicker);

pet.moreMoney(clicker);

logger.info("pressed clicker");

Client.sendMessage(pet.name+": заработал десяточку");

}

if(gameCollide(radio, e)){

pet.moreHappinessGame(radio);

logger.info("pressed radio");

}

}

if (imageCollide(homeB, e)){

Var.switchScreen = Display.HOME\_SCREEN;

}

if (imageCollide(cutlB, e)){

Var.switchScreen = Display.FOOD\_SCREEN;;

}

if (imageCollide(gameB, e)){

Var.switchScreen = Display.GAME\_SCREEN;

}

if (imageCollide(restaB, e) && Var.switchScreen == Display.DEAD){

try {

Var.petSave.createNewFile();

} catch (IOException ex) {

ex.printStackTrace();

}

pet.startValues();

c.buffer();

}

}

});

Var.healthTimer = Var.init();

Timer timer = new Timer();

timer.schedule(new TimerTask() {

@Override

public void run() {

Var.currentTime = System.currentTimeMillis();

c.basicLayer();

if (Var.switchScreen != Display.DEAD){

c.petStats();

logger.info("switch screen");

}

if (Var.switchScreen == Display.FOOD\_SCREEN){

c.feeding();

logger.info("switch screen feeding");

}

if (Var.switchScreen == Display.HOME\_SCREEN){

c.petAnimation();

}

if (Var.switchScreen == Display.GAME\_SCREEN){

c.games();

logger.info("switch screen games");

}

if (Var.switchScreen == Display.DEAD){

c.dead();

logger.info("switch screen dead");

}

if(pet.money == 0){

c.noMoney();

}

if (pet.happinessLvl == 0 && pet.illTimer == null){

pet.illness();

}

if(pet.aPetTimer == null){

pet.petA();

}

c.buffer();

pet.updatePet();

if (pet.moreLove && pet.lveTimer == null) {

pet.petLove();

}

if ((Var.currentTime - Var.healthTimer) >= 500) {

pet.petHealth();

pet.petHappiness();

Var.healthTimer = System.currentTimeMillis();

}

//TODO: нужно сделать норм сохранение питомца, чтобы не было 2 сохранения(одно которое сохраняется прям при создании) и вот это которое сохраняется по хакрыют игры(нужно оставить только это)

// и нужно сделать так, чтобы имя сохранения было такое же как и имя персонажа(т.е. если есть какие то запрещенные символы в имене(все кроме букв) они заменяются звездочками

//нужно реализовать тут так же сохранение, чтобы оно было со звездочками

save(pet);

}

}, 0, 60);

logger.info("Application exiting");

}

private static String chooseSave() {

File savesDirectory = new File("saves");

if (savesDirectory.exists() && savesDirectory.isDirectory()) {

File[] saveFiles = savesDirectory.listFiles();

if (saveFiles != null && saveFiles.length > 0) {

File selectedFile = (File) JOptionPane.showInputDialog(

null,

"Choose a save to load:",

"Load Save",

JOptionPane.QUESTION\_MESSAGE,

null,

saveFiles,

saveFiles[0]

);

if (selectedFile != null) {

Var.petSave = selectedFile;

return selectedFile.getName().replaceFirst("[.][^.]+$", "");

}

}

}

logger.info("Inside chooseSave");

return null; // No existing saves found

}

private static String createNewSave() {

String saveName;

boolean validInput = false;

do {

saveName = JOptionPane.showInputDialog("Enter a name for the new save:");

if (saveName != null && !saveName.isEmpty() && saveName.matches("^[\\p{Alnum}]+$")) {

validInput = true;

} else {

JOptionPane.showMessageDialog(null, "Please enter a valid name (only letters and digits).");

}

} while (!validInput);

try {

File savesDirectory = new File("saves");

if (!savesDirectory.exists()) {

savesDirectory.mkdir();

}

Var.petSave = new File(savesDirectory, saveName + ".json");

Var.petSave.createNewFile();

return saveName;

} catch (IOException e) {

e.printStackTrace();

}

return null;

}

private static void createNewCharacter(Pet pet) { String petSaveName = JOptionPane.showInputDialog("Enter a name for the new save:");

if (petSaveName != null && !petSaveName.isEmpty()) {

// Обработка текста с использованием TextProcessor

String processedName = TextProcessor.processText(petSaveName);

Var.petSave = new File("saves", processedName + ".json");

pet.name = processedName; // Обновляем имя персонажа

} else {

// Handle the case where the user cancels or enters an empty name

JOptionPane.showMessageDialog(null, "Please enter a valid name. Exiting.");

return;

}

pet.startValues();

save(pet);

logger.info("Inside createNewCharacter");

}

public static void save(Pet p) {

try {

JSONObject json = new JSONObject();

json.put("name", p.name);

json.put("loveLvl", p.loveLvl);

json.put("health", p.healthLvl);

json.put("happiness", p.happinessLvl);

json.put("money", p.money);

// Convert JSONObject to a formatted string

String jsonString = json.toString(2);

// Write the string to the file

FileWriter fw = new FileWriter(Var.petSave);

fw.write(jsonString);

fw.flush();

fw.close();

} catch (Exception e) {

e.printStackTrace();

}

}

public static void reloadSave(Pet p) {

try {

String content = Files.readString(Paths.get(Var.petSave.toURI()));

if (!content.isEmpty()) {

JSONObject json = new JSONObject(content);

p.name = json.optString("name", "");

p.loveLvl = json.optInt("loveLvl", 0);

p.healthLvl = json.optInt("health", 0);

p.happinessLvl = json.optInt("happiness", 0);

p.money = json.optInt("money", 0);

}

} catch (Exception e) {

e.printStackTrace();

}

logger.info("Inside reloadSave");

}

private static void writeToExcelOnExit(String petName) {

try {

String folderName = "excel\_saves";

File folder = new File(folderName);

if (!folder.exists()) {

folder.mkdir();

}

String fileName = petName.replaceAll("\\s", "\_") + "\_GameLog.xlsx";

File excelFile = new File(folder, fileName);

XSSFWorkbook workbook;

if (excelFile.exists()) {

FileInputStream fis = new FileInputStream(excelFile);

workbook = new XSSFWorkbook(fis);

fis.close();

} else {

workbook = new XSSFWorkbook();

}

Sheet sheet = workbook.getSheet("PetData");

if (sheet == null) {

sheet = workbook.createSheet("PetData");

}

SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

String currentTime = dateFormat.format(new Date());

Row row = sheet.createRow(sheet.getLastRowNum() + 1);

row.createCell(0).setCellValue(currentTime);

row.createCell(1).setCellValue(petName);

try (FileOutputStream fileOut = new FileOutputStream(excelFile)) {

workbook.write(fileOut);

}

workbook.close();

logger.info("Inside writeToExcelOnExit");

} catch (Exception e) {

e.printStackTrace();

logger.error("Error in writeToExcelOnExit", e);

}

}

public static void startNewGame(Pet pet, CanvasLayer canvasLayer) {

String petName = JOptionPane.showInputDialog("Enter your pet's name:");

if (petName != null && !petName.isEmpty()) {

pet.name = petName;

} else {

// Handle the case where the user cancels or enters an empty name

// You can add additional logic here as needed

return;

} // Add logic to reset the game state for a new game

pet.startValues();

canvasLayer.buffer();

logger.info("Inside startnewgame");

}

public static void restartGame(Pet pet, CanvasLayer canvasLayer) {

// Add logic to restart the game state

try {

Var.petSave.createNewFile();

} catch (IOException ex) {

ex.printStackTrace();

}

pet.startValues();

canvasLayer.buffer();

logger.info("Inside restartgame");

}

public static void playBackgroundMusic(String fileName) {

try {

File soundFile = new File(fileName);

AudioInputStream audioInputStream = AudioSystem.getAudioInputStream(soundFile);

Clip clip = AudioSystem.getClip();

clip.open(audioInputStream);

clip.loop(Clip.LOOP\_CONTINUOUSLY);

clip.start();

} catch (Exception e) {

e.printStackTrace();

}

}

public static boolean foodCollide(Food food, MouseEvent mouse) {

return mouse.getX() > food.x && mouse.getX() < (food.x + food.ovalWidth) &&

mouse.getY() > food.y && mouse.getY() < (food.y + food.ovalHeight);

}

public static boolean gameCollide(Games games, MouseEvent mouse){

return mouse.getX() > games.x && mouse.getX() < (games.x + games.ovalWidth) &&

mouse.getY() > games.y && mouse.getY() < (games.y + games.ovalHeight);

}

public static boolean imageCollide(MenuButtons b,MouseEvent mouse){

return mouse.getX() > b.x && mouse.getX() < (b.x + b.width) &&

mouse.getY() > b.y && mouse.getY() < (b.y + b.height);

}

private static void configureLogger() {

// Можете добавить свои настройки логгера здесь

// Например, LogManager.setFactory() или установить конфигурацию программно

// Но в данном случае, логгер будет настроен согласно файлу log4j2.xml

}

}

# **3 Реализация программного обеспечения**

Изображение выглядит как текст, снимок экрана, Мультфильм, Анимация

Автоматически созданное описание  
Рисунок – Интерфейс “Главное окно”

Код функций используемых в окне

package com.example;

import javax.imageio.ImageIO;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.awt.event.ActionListener;

import java.awt.image.\*;

import java.io.File;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Locale;

import java.text.MessageFormat;

import java.awt.MenuItem;

import java.util.ResourceBundle;

public class CanvasLayer implements ActionListener {

public Pet name;

GameLogic gl = new GameLogic();

JFrame mainFrame = new JFrame("Prison Tamagotchu");

Canvas canvas = new Canvas();

ArrayList<Food> foodCanvas;

ArrayList<Games> gamesCanvas;

Graphics g;

int width = 615;

int height = 635;

BufferStrategy bufferStrategy;

private MenuItem newGame;

private MenuItem continueGame;

private MenuItem engLang;

private MenuItem rusLang;

private MenuItem exit;

private MenuItem restart;

BufferedImage cupcake, pizza, broccoli, bg, cutlery, home, games, cursor, radio;

private ResourceBundle messages;

@Override

public void actionPerformed(ActionEvent e) {

String command = e.getActionCommand();

switch (command) {

case "New":

GameLogic.startNewGame(name, this);

break;

case "Continue":

// Handle Continue action

break;

case "Restart":

GameLogic.restartGame(name, this);

break;

case "Exit":

// Handle Exit action

System.exit(0);

break;

case "eng":

messages = ResourceBundle.getBundle("Messages", Locale.ENGLISH);

updateUI();

break;

case "rus":

messages = ResourceBundle.getBundle("Messages", new Locale("ru", "RU"));

updateUI();

break;

// Add more cases as needed

}

}

public CanvasLayer(ArrayList<Food> foodCanvas, ArrayList<Games> gamesCanvas, Pet name) {

this.foodCanvas = foodCanvas;

this.gamesCanvas = gamesCanvas;

this.name = name;

MenuBar mb = new MenuBar();

mainFrame.setMenuBar(mb);

messages = ResourceBundle.getBundle("Messages", Locale.getDefault());

Menu menu = new Menu(messages.getString("fileMenuLabel"));

newGame = new MenuItem(messages.getString("newGameMenuItemLabel"));

newGame.setActionCommand("New");

newGame.addActionListener(this);

continueGame = new MenuItem(messages.getString("continueMenuItemLabel"));

continueGame.setActionCommand("Continue");

continueGame.addActionListener(this);

restart = new MenuItem(messages.getString("restartMenuItemLabel"));

restart.setActionCommand("Restart");

restart.addActionListener(this);

exit = new MenuItem(messages.getString("exitMenuItemLabel"));

exit.setActionCommand("Exit");

exit.addActionListener(this);

Menu language = new Menu(messages.getString("languageMenuLabel"));

engLang = new MenuItem(messages.getString("englishMenuItemLabel"));

engLang.setActionCommand("eng");

engLang.addActionListener(this);

rusLang = new MenuItem(messages.getString("russianMenuItemLabel"));

rusLang.setActionCommand("rus");

rusLang.addActionListener(this);

language.add(engLang);

language.add(rusLang);

menu.add(newGame);

menu.add(continueGame);

menu.add(restart);

menu.addSeparator();

menu.add(language);

menu.addSeparator();

menu.add(exit);

mb.add(menu);

// filepath to non animated images (food & background)

try {

bg = ImageIO.read(new File("src/images/bg.png"));

home = ImageIO.read(new File("src/images/home.png"));

games = ImageIO.read(new File("src/images/gamepad.png"));

cursor = ImageIO.read(new File("src/images/gamepad.png"));

radio = ImageIO.read(new File("src/images/gamepad.png"));

cutlery = ImageIO.read(new File("src/images/cutlery.png"));

cupcake = ImageIO.read(new File("src/images/sigarette.png"));

pizza = ImageIO.read(new File("src/images/pizza.png"));

broccoli = ImageIO.read(new File("src/images/broccoli.png"));

//restart = ImageIO.read(new File("src/images/restart.png"));

} catch (IOException e) {

e.printStackTrace();

}

// setting up the frame and canvas

mainFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

mainFrame.setSize(width, height);

mainFrame.setVisible(true);

mainFrame.setLocationRelativeTo(null);

mainFrame.setResizable(false);

canvas.setSize(width, height);

//canvas.setBackground(new Color(126, 146, 203, 255));

canvas.setVisible(true);

mainFrame.add(canvas);

canvas.createBufferStrategy(3);

bufferStrategy = canvas.getBufferStrategy();

}

public void basicLayer() {

//button dummies

// MenuButtons homeB = new MenuButtons();

// homeB.homeParameter();

// MenuButtons cutlB = new MenuButtons();

// cutlB.restartParameter();

g = bufferStrategy.getDrawGraphics();

g.clearRect(0, 0, width, height);

g.drawImage(bg, 0, 0, null);

g.drawImage(home, 0, 0, null);

g.drawImage(games, 150,435, null);

g.drawImage(cutlery, 0, 0, null);

g.setColor(new Color(0, 0, 0));

// g.drawRect(homeB.x, homeB.y, homeB.width, homeB.height);

// g.drawRect(cutlB.x, cutlB.y, cutlB.width, cutlB.height);

}

public void petStats() {

g.setColor(new Color(255, 255, 255));

g.setFont(new Font("Arial", Font.BOLD, 16));

g.drawString(messages.getString("nameLabel") + " " + name.name, 290, 100);

g.setColor(new Color(0, 0, 0, 119));

g.drawString("Healths", 291, 121);

for (int i = 1; i <= name.healthLvl / 10; i++) {

int rectPos = 351 + i \* 11;

g.drawRect(rectPos, 111, 8, 10);

}

g.drawString("Happiness", 286, 141);

for (int i = 1; i <= name.happinessLvl / 10; i++) {

int rectPos = 351 + i \* 11;

g.drawRect(rectPos, 131, 8, 10);

}

g.setColor(new Color(0, 255, 69, 255));

g.drawString(messages.getString("healthLabel"), 290, 120);

for (int i = 1; i <= name.healthLvl / 10; i++) {

int rectPos = 350 + i \* 11;

g.drawRect(rectPos, 110, 8, 10);

}

g.setColor(new Color(255, 221, 0));

g.drawString(messages.getString("happinessLabel"), 285, 140);

for (int i = 1; i <= name.happinessLvl / 10; i++) {

int rectPos = 350 + i \* 11;

g.drawRect(rectPos, 130, 8, 10);

}

g.setColor(new Color(255,0,0));

g.drawString(messages.getString("moneyLabel"), 290, 160);

g.drawString(String.valueOf(name.money), 370, 160);

try {

if (name.loveLvl < 100) {

g.drawImage(ImageIO.read(new File(Heart.risingHeart((name.loveLvl) / 5))), 102, 90, null);

} else {

g.drawImage(ImageIO.read(new File(Heart.risingHeart(19))), 102, 90, null);

}

} catch (IOException e) {

e.printStackTrace();

}

}

public void petAnimation() {

try {

g.drawImage(ImageIO.read(new File(name.filePet)), 0, 80, null);

} catch (IOException e) {

e.printStackTrace();

}

} public void mainMenu(){

g.setColor(new Color(255,255,0));

g.drawString("Main Menu", width / 2, height / 2 - 80);

}

public void feeding() {

g.setColor(new Color(255, 255, 0));

g.drawString(messages.getString("feedPrompt"), width / 2 - 70, height / 2 - 80);

for (Food i : foodCanvas) {

if (i == foodCanvas.get(0)) {

g.drawString(messages.getString("cigarette"), i.x + 30, i.y + 100);

g.drawImage(cupcake, i.x, i.y, null);

g.drawString("10 $", i.x + 40, i.y + 120);

}

if (i == foodCanvas.get(1)) {

g.drawString(messages.getString("pizza"), i.x + 40, i.y + 100);

g.drawImage(pizza, i.x, i.y, null);

g.drawString("30 $", i.x + 40, i.y + 120);

}

if (i == foodCanvas.get(2)) {

g.drawString(messages.getString("broccoli"), i.x + 30, i.y + 100);

g.drawImage(broccoli, i.x, i.y, null);

g.drawString("30 $", i.x + 40, i.y + 120);

}

}

}

public void games() {

g.setColor(new Color(255, 255, 0));

g.drawString(messages.getString("gamesPrompt"), width / 2 - 70, height / 2 - 80);

for (Games i : gamesCanvas) {

if (i == gamesCanvas.get(0)) {

g.drawString(messages.getString("clickerGame"), i.x + 30, i.y + 100);

g.drawImage(cursor, i.x, i.y, null);

}

if (i == gamesCanvas.get(1)) {

g.drawString(messages.getString("radioGame"), i.x + 40, i.y + 100);

g.drawImage(radio, i.x, i.y, null);

}

}

}

public void noMoney() {

g.setColor(new Color(255, 255, 0));

g.drawString(messages.getString("noMoney"), width / 2 - 70, height / 2 - 70);

}

public void dead() {

g.setColor(new Color(255, 255, 0));

g.drawString(MessageFormat.format(messages.getString("petDied"), name.name), 250, 200);

//g.drawImage(restart, 0, 0, null);

}

private void updateUI() {

// Update UI elements with localized text

mainFrame.setTitle(messages.getString("mainMenu"));

MenuBar mb = mainFrame.getMenuBar();

Menu menu = mb.getMenu(0);

menu.setLabel(messages.getString("fileMenuLabel"));

MenuItem newGame = menu.getItem(0);

newGame.setLabel(messages.getString("newGameMenuItemLabel"));

MenuItem continueGame = menu.getItem(1);

continueGame.setLabel(messages.getString("continueMenuItemLabel"));

MenuItem restart = menu.getItem(2);

restart.setLabel(messages.getString("restartMenuItemLabel"));

MenuItem exit = menu.getItem(3);

exit.setLabel(messages.getString("exitMenuItemLabel"));

Menu language = new Menu(messages.getString("languageMenuLabel"));

engLang = new MenuItem(messages.getString("englishMenuItemLabel"));

engLang.setActionCommand("eng");

engLang.addActionListener(this);

rusLang = new MenuItem(messages.getString("russianMenuItemLabel"));

rusLang.setActionCommand("rus");

rusLang.addActionListener(this);

}

public void buffer() {

bufferStrategy.show();

g.dispose();

}

} public void mainMenu(){

g.setColor(new Color(255,255,0));

g.drawString("Main Menu", width / 2, height / 2 - 80);

}

public void feeding() {

g.setColor(new Color(255, 255, 0));

g.drawString(messages.getString("feedPrompt"), width / 2 - 70, height / 2 - 80);

for (Food i : foodCanvas) {

if (i == foodCanvas.get(0)) {

g.drawString(messages.getString("cigarette"), i.x + 30, i.y + 100);

g.drawImage(cupcake, i.x, i.y, null);

g.drawString("10 $", i.x + 40, i.y + 120);

}

if (i == foodCanvas.get(1)) {

g.drawString(messages.getString("pizza"), i.x + 40, i.y + 100);

g.drawImage(pizza, i.x, i.y, null);

g.drawString("30 $", i.x + 40, i.y + 120);

}

if (i == foodCanvas.get(2)) {

g.drawString(messages.getString("broccoli"), i.x + 30, i.y + 100);

g.drawImage(broccoli, i.x, i.y, null);

g.drawString("30 $", i.x + 40, i.y + 120);

}

}

}

public void games() {

g.setColor(new Color(255, 255, 0));

g.drawString(messages.getString("gamesPrompt"), width / 2 - 70, height / 2 - 80);

for (Games i : gamesCanvas) {

if (i == gamesCanvas.get(0)) {

g.drawString(messages.getString("clickerGame"), i.x + 30, i.y + 100);

g.drawImage(cursor, i.x, i.y, null);

}

if (i == gamesCanvas.get(1)) {

g.drawString(messages.getString("radioGame"), i.x + 40, i.y + 100);

g.drawImage(radio, i.x, i.y, null);

}

}

}

public void noMoney() {

g.setColor(new Color(255, 255, 0));

g.drawString(messages.getString("noMoney"), width / 2 - 70, height / 2 - 70);

}

public void dead() {

g.setColor(new Color(255, 255, 0));

g.drawString(MessageFormat.format(messages.getString("petDied"), name.name), 250, 200);

//g.drawImage(restart, 0, 0, null);

}

private void updateUI() {

// Update UI elements with localized text

mainFrame.setTitle(messages.getString("mainMenu"));

MenuBar mb = mainFrame.getMenuBar();

Menu menu = mb.getMenu(0);

menu.setLabel(messages.getString("fileMenuLabel"));

MenuItem newGame = menu.getItem(0);

newGame.setLabel(messages.getString("newGameMenuItemLabel"));

MenuItem continueGame = menu.getItem(1);

continueGame.setLabel(messages.getString("continueMenuItemLabel"));

MenuItem restart = menu.getItem(2);

restart.setLabel(messages.getString("restartMenuItemLabel"));

MenuItem exit = menu.getItem(3);

exit.setLabel(messages.getString("exitMenuItemLabel"));

Menu language = new Menu(messages.getString("languageMenuLabel"));

engLang = new MenuItem(messages.getString("englishMenuItemLabel"));

engLang.setActionCommand("eng");

engLang.addActionListener(this);

rusLang = new MenuItem(messages.getString("russianMenuItemLabel"));

rusLang.setActionCommand("rus");

rusLang.addActionListener(this);

}

public void buffer() {

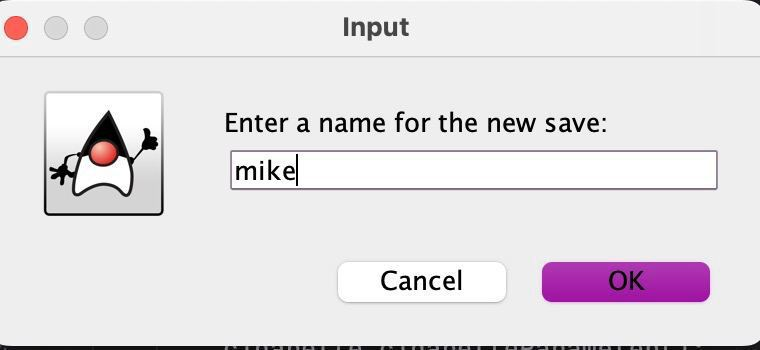
bufferStrategy.show();

g.dispose();

}

}

# **РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

  
Рисунок 3.1 – Интерфейс окна “Сохранение”

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.2 – Интерфейс окна “Загрузка персонажа”

Изображение выглядит как текст, снимок экрана, Мультфильм, Анимация

Автоматически созданное описание

Рисунок 3.3 – Интерфейс “Главное окно”

Изображение выглядит как текст, снимок экрана, мультфильм, Мультимедийное программное обеспечение

Автоматически созданное описание  
Рисунок 3.4 – Интерфейс “Кормление”

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание  
Рисунок 3.5 – Интерфейс “Смерть”  
  
Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

Рисунок 3.6 – Интерфейс “Локализация”

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсовой работы:

* Были закреплены знания по курсу “Конструирование программного обеспечения”;
* Приобретен опыт при разработке объектно-ориентированных программ;
* Были укреплены знания в конструировании и структурировании приложения;
* Освоены методы передачи данных между приложениями;
* Изучены возможности создания справочной системы.

В ходе выполнения курсовой работы, посвященной созданию игры в стиле "Тамагочи" с атмосферой тюремного быта, были осуществлены значительные усилия по разработке данного проекта на языке программирования Java.

Используя интегрированную среду разработки IntelliJ IDEA и технологии Java AWT и JPanel, удалось создать интерфейс, который отражает атмосферу тюремной жизни и позволяет пользователям взаимодействовать с персонажем, подобно игре "Тамагочи". Эти фреймворки обеспечили необходимый инструментарий для создания пользовательского интерфейса, который визуализирует жизнь в условиях тюрьмы.

Выбор Java в качестве основного языка программирования был обусловлен его переносимостью и удобством в разработке, что сыграло ключевую роль в создании данного приложения.

В результате данного проекта было продемонстрировано успешное применение выбранных технологий для достижения поставленной цели - создания увлекательной игры в жанре "Тамагочи" с уникальной тематикой тюрьмы. Это также подтверждает возможности применения Java и связанных с ней фреймворков для разработки разнообразных игровых приложений.

В целом, выполнение данной курсовой подчеркнуло значимость использования современных технологий для создания разнообразных и захватывающих проектов, каким бы ни был их сюжет и концепция.

**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. "Head First Java" by Kathy Sierra, Bert Bates - Это отличная книга для начинающих, вводящая в мир Java с помощью доступного и занимательного подхода.

2. "Java: The Complete Reference" by Herbert Schildt - Эта книга рассматривает язык Java в деталях, предоставляя информацию о всех его аспектах.

3. "Thinking in Java" by Bruce Eckel - Очень подробная книга, которая поможет вам понять философию и принципы разработки на Java.

4. "Effective Java" by Joshua Bloch - В этой книге автор дает рекомендации по использованию Java в повседневной разработке, предоставляя лучшие практики и советы.

5. "Java Swing" by Marc Loy, Robert Eckstein, Dave Wood, James Elliott - Руководство по библиотеке Swing, которая является частью java.awt и предоставляет множество инструментов для создания графических интерфейсов.

6. "Filthy Rich Clients: Developing Animated and Graphical Effects for Desktop Java Applications" by Chet Haase, Romain Guy - Эта книга предлагает глубокое погружение в разработку графических приложений с использованием Swing и других графических технологий.

7. "Java AWT Reference" by John Zukowski - Справочник по AWT, который содержит информацию о классах, методах и примерах использования для java.awt.

8. "Java AWT (Abstract Window Toolkit) By Example" by John Zukowski - Эта книга предлагает примеры использования java.awt для создания графических пользовательских интерфейсов.

9. "Java Foundations: Introduction to Program Design and Data Structures" by John Lewis, Peter DePasquale, Joe Chase - Книга, которая вводит в основы программирования на Java и включает в себя информацию о java.awt.

10. "Java GUI Development" by Alex Banks - Книга о разработке графических интерфейсов на Java, включая java.awt и Swing.

11. "Java Programming for Beginners: Learn Java in a Week!" by Os Swift - Простое руководство по основам программирования на Java, включая базовые концепции java.awt.

12. "Java 2D Graphics" by Jonathan Knudsen - Книга о создании 2D графики на Java с использованием java.awt.

13. "Java How to Program" by Paul Deitel, Harvey Deitel - Книга, которая покрывает множество тем, включая разработку пользовательских интерфейсов с использованием java.awt

# **Приложение A**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | |
|  |  |  |  |  | КП–10701421–2023-01 | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № документа | Подпись | Дата |
| Разраб. | | Крюков |  |  | *UML-Диаграмма зависимости классов* | Лит | | | Лист | Листов |
| Руковод. | | Станкевич |  |  |  | Д |  | *1* | *4* |
| Консульт. | | Станкевич |  |  | 1-40 01 01 БНТУ  г. Минск | | | | |
| Н.контр. | | Станкевич |  |  |
| Зав.каф. | | Полозков |  |  |