|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Politechnika Bydgoska im. Jana i Jędrzeja Śniadeckich  Wydział Telekomunikacji, Informatyki i Elektrotechniki  al. prof. S. Kaliskiego 7, 85-796 Bydgoszcz | | | |  | |
| Przedmiot | **Programowanie urządzeń mobilnych** | | | Kierunek/Tryb |  |
| Nr laboratorium | 3 | Data wykonania |  | Grupa |  |
| Ocena |  | Data oddania |  | Imię Nazwisko |  |
| Nazwa ćwiczenia | Interfejs użytkownika i obsługa zdarzeń | | |

**Cel ćwiczenia laboratoryjnego**

Celem ćwiczenia jest zapoznanie studentów z tworzeniem prostej gry mobilnej na platformie Android przy użyciu interfejsu użytkownika **EditText,** **Button,** **TextView, EditText**. Studenci mają nauczyć się:

1. Projektować interfejs użytkownika, który obejmuje wprowadzanie danych, przycisk do sprawdzania odpowiedzi oraz wyświetlanie wyników i komunikatów.
2. Programować obsługę zdarzeń związaną z rozgrywką.
3. Wykonywać proste operacje na danych wejściowych.
4. Wyświetlać wynik i komunikaty na ekranie w czytelny sposób.
5. Obsługiwać potencjalne błędy podczas wprowadzania danych.

**Zadanie**

Studenci zostaną poproszeni o stworzenie aplikacji, która umożliwia grę w wisielca. Program ma wyświetlać ukryte słowo zastępując litery np. znakami "\_". Gracz będzie próbował odgadnąć to słowo, zgadując pojedyncze litery. Jeśli gracz zgadnie słowo, wygrywa grę. W przeciwnym razie ma ograniczoną liczbę prób na odgadnięcie słowa, zanim przegra.

**Reguły gry**

1. Program ma wybrać ukryte słowo, które gracz będzie próbował odgadnąć. Słowo to powinno być wcześniej zdefiniowane w kodzie programu.
2. Gracz ma maksymalnie 6 prób na odgadnięcie słowa. Każda nieudana próba zwiększa licznik prób.
3. Na początku gry wyświetla się ukryte słowo jako ciąg znaków "\_" odpowiadający liczbie liter w słowie. Gracz nie zna zawartości słowa.
4. Program wyświetla aktualny stan słowa z zamaskowanymi literami oraz informuje o liczbie pozostałych prób.
5. Gracz próbuje odgadnąć literę, wprowadzając ją z klawiatury.
6. Jeśli litera jest w słowie, program aktualizuje wyświetlane słowo, odkrywając odgadnięte litery.
7. Jeśli litera nie jest w słowie, program zwiększa licznik prób i wyświetla informację, że litera jest nieprawidłowa.
8. Gra trwa do momentu, aż gracz odgadnie całe słowo lub przekroczy limit prób.
9. Jeśli gracz odgadnie całe słowo przed wykorzystaniem wszystkich prób, wygrywa i program wyświetla odpowiednie gratulacje.
10. Jeśli gracz przekroczy limit prób, program informuje go o przegranej i pokazuje całe słowo.
11. Program powinien być napisany w sposób czytelny i komentowany, aby inni mogli łatwo zrozumieć, jak działa.

**Opcjonalne rozszerzenia**

* Możesz dodać bardziej rozbudowany interfejs użytkownika, który pozwoli graczowi wybierać słowa lub odczytywać je z pliku.
* Możesz rozszerzyć listę słów, które gracz może odgadywać, i losowo wybierać jedno z tych słów.
* Możesz dodać zapis wyników gry i wyników graczy.

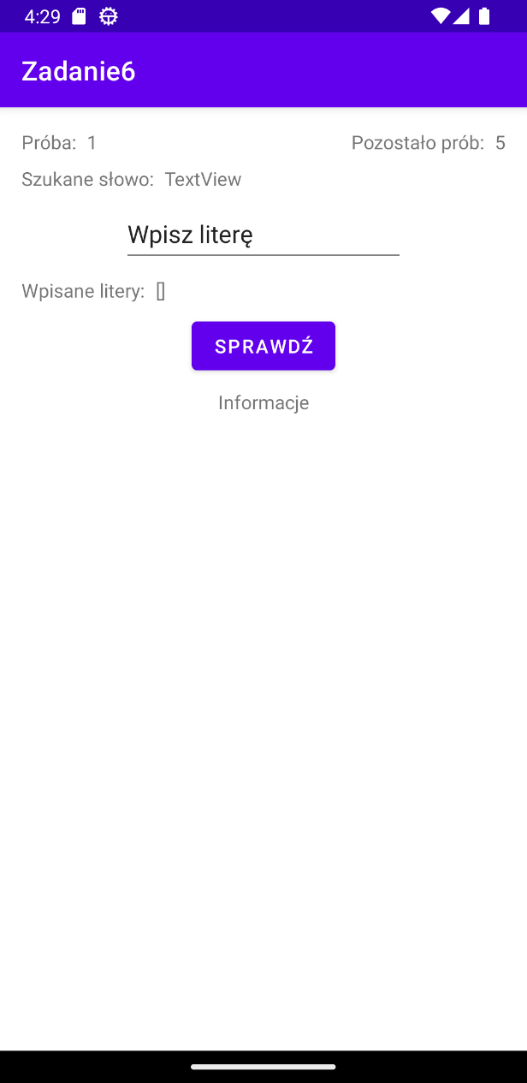
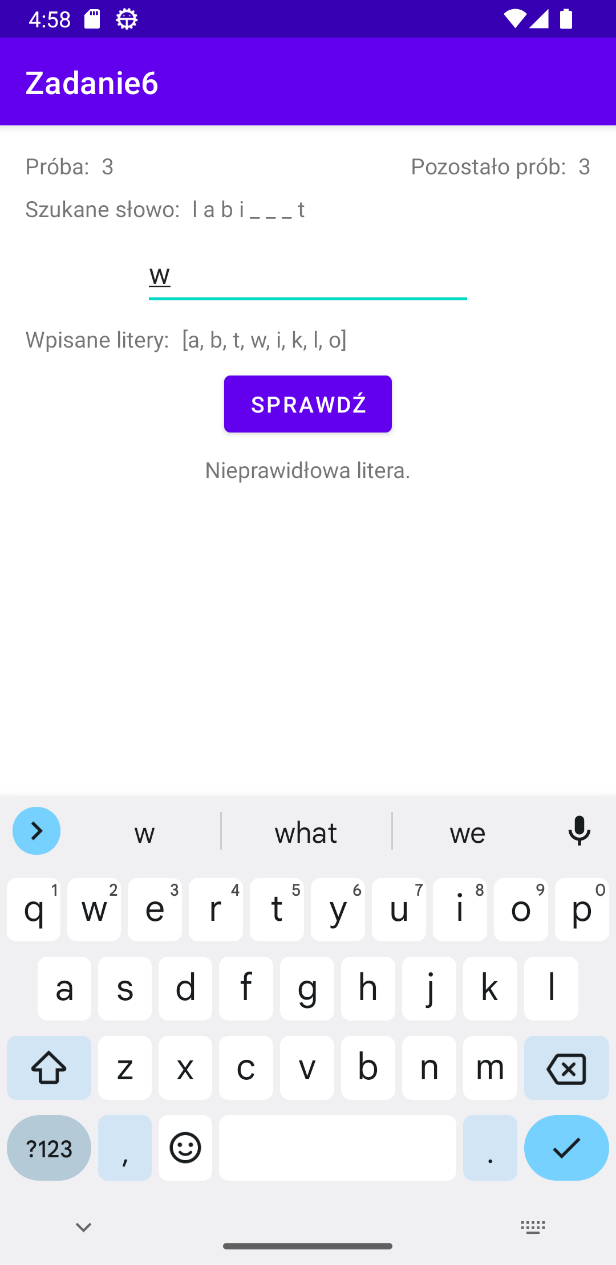
**Oczekiwane wyniki**

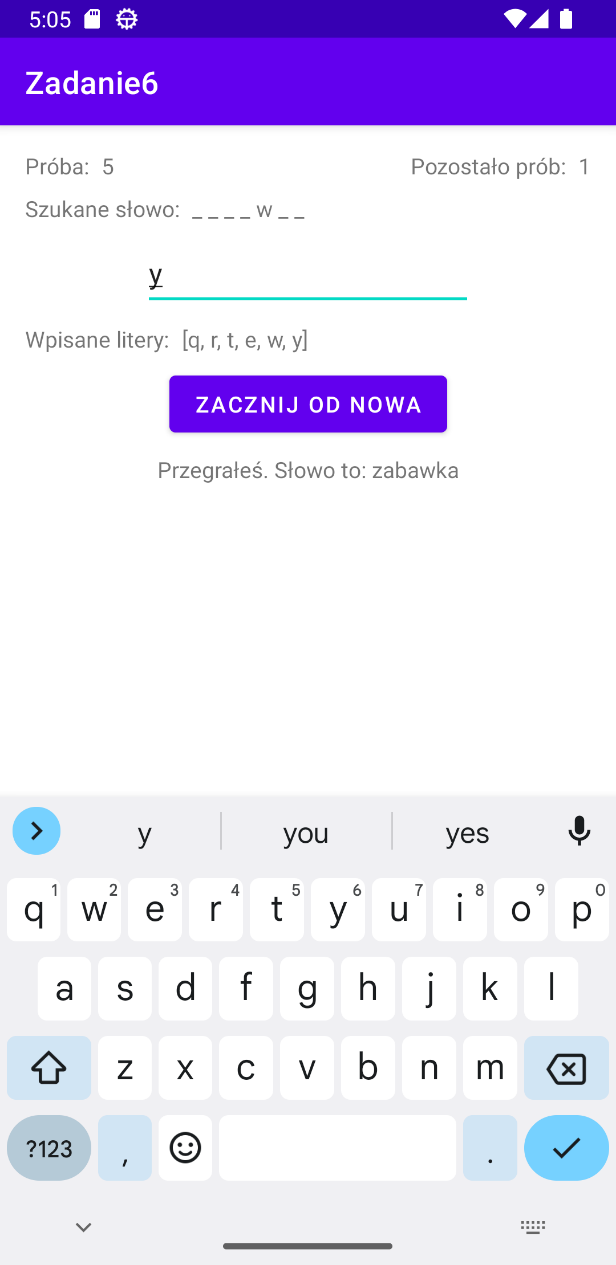
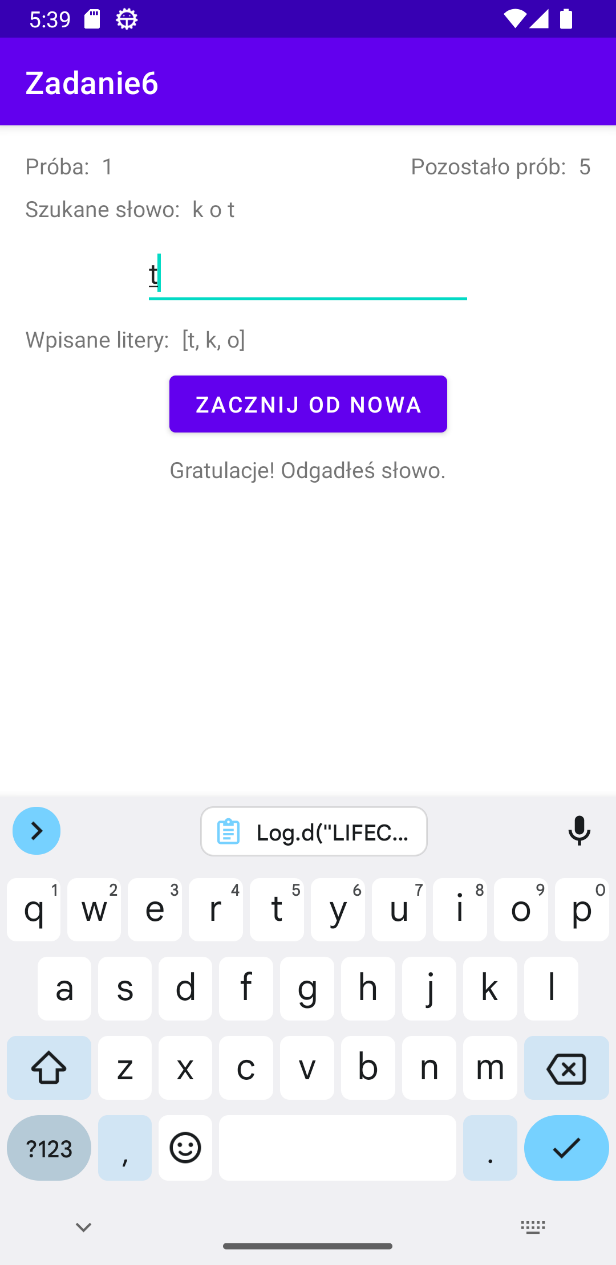
Po zakończeniu tego ćwiczenia studenci powinni być w stanie:

1. Tworzyć interfejsy użytkownika w aplikacji mobilnej.
2. Programować obsługę zdarzeń związaną z rozgrywką.
3. Przeprowadzać logikę gry w wisielca.
4. Wyświetlać stan gry na ekranie w formie tekstu.
5. Zrozumieć i obsługiwać błędy w grze.
6. Testować i debugować grę.
7. Projektować interaktywne rozwiązania gier.
8. Rozwiązywać problemy związane z logiką gry.
9. Organizować kod i dokumentować go.
10. Tworzyć aplikacje mobilne w Android Studio.

**Zadanie**

Należy utworzyć aplikację, która umożliwia grę w wisielca. Program ma wyświetlać ukryte słowo zastępując litery np. znakami "\_". Gracz będzie próbował odgadnąć to słowo, zgadując pojedyncze litery. Jeśli gracz zgadnie słowo, wygrywa grę. W przeciwnym razie ma ograniczoną liczbę prób na odgadnięcie słowa, zanim przegra. Na poniższych rysunkach przedstawiono przykładowe rozwiązanie ilustrujące wygląd interfejsu aplikacji.

**Sprawozdanie**

Sprawozdanie z ćwiczenia w ramach nauki programowania aplikacji mobilnych powinno zawierać istotne informacje i dokumentację dotyczącą zadania oraz jego realizacji. Oto kilka kluczowych elementów, które powinny być uwzględnione w sprawozdaniu:

**1. Tytuł i informacje ogólne**

* Tytuł ćwiczenia.
* Imię i nazwisko studenta.
* Data realizacji ćwiczenia.

**2. Cel ćwiczenia**

* Krótka informacja o celu i znaczeniu ćwiczenia, jak również o tym, czego studenci mieli się nauczyć.

**3. Opis projektu**

* Opis funkcjonalności gry Wisielec w ramach projektu.
* Charakteryzacja interfejsu użytkownika, zawierająca informacje o wykorzystanych widokach i elementach interfejsu.

**4. Implementacja**

* Opis procesu tworzenia projektu, wraz z krokami realizacji gry Wisielec.
* Omówienie wykorzystanych narzędzi i technologii, w tym Android Studio, język Java/Kotlin.
* Przedstawienie kodu źródłowego aplikacji, zarówno XML (layout) jak i kodu Java/Kotlin.

**5. Funkcje kluczowe**

* Omówienie kluczowych funkcji aplikacji, takich jak obsługa zgadywania liter, zarządzanie stanem gry oraz sposób prezentacji wyników.

**6. Testowanie**

* Opis testowania gry Wisielec, włączając w to przykłady testów przeprowadzonych w trakcie implementacji.
* Przykładowe przypadki testowe i raport z wynikami testów.

**7. Wyniki**

* Przedstawienie wyników działania aplikacji, w tym zrzuty ekranu demonstrujące działanie gry.

**8. Podsumowanie**

* Krótka ocena projektu i osiągnięć w kontekście zrealizowanego zadania.
* Wnioski wynikające z ćwiczenia, jakie umiejętności i doświadczenie zdobyli studenci.

**9. Trudności i błędy**

* Informacje na temat ewentualnych problemów napotkanych podczas implementacji gry i jak zostały one rozwiązane.

**10. Źródła i odniesienia**

* Jeśli korzystano z materiałów lub źródeł zewnętrznych, uwzględnij je w tekście.

**11. Dodatkowe materiały**

* Ewentualne dodatkowe materiały, takie jak kody źródłowe, zrzuty ekranu lub inne dokumentacje, które uzupełniają sprawozdanie.

**Zalecenia ogólne**

* Sprawozdanie powinno być czytelne i przejrzyste, z odpowiednimi nagłówkami i numeracją stron.
* Projektowanie i implementacja powinny być opisane w sposób logiczny i zrozumiały.
* Sprawozdanie powinno być dostatecznie szczegółowe, aby inny programista mógł zrozumieć projekt i ewentualnie go udoskonalić lub wykorzystać w przyszłości.

**Przykład do wykorzystania w ćwiczeniu**

Poniżej przedstawiona jest implementacja gry wisielec w Javie.

**import** java**.**util**.**Scanner**;**

public class Wisielec **{**

public static void main**(**String**[]** args**)** **{**

String secretWord **=** "programowanie"**;** // Słowo, które gracz musi odgadnąć

int maxAttempts **=** 6**;** // Maksymalna liczba prób

char**[]** guessedLetters **=** **new** char**[**secretWord**.**length**()];** // Tablica do przechowywania zgadniętych liter

int attempts **=** 0**;** // Licznik prób

Scanner scanner **=** **new** Scanner**(**System**.**in**);**

**while** **(**attempts **<** maxAttempts**)** **{**

// Wyświetl aktualny stan słowa z zamaskowanymi literami

displayWord**(**secretWord**,** guessedLetters**);**

// Poproś użytkownika o literę

System**.**out**.**print**(**"Podaj literę: "**);**

char guess **=** scanner**.**next**().**charAt**(**0**);**

// Sprawdź, czy litera jest w słowie

**if** **(**secretWord**.**contains**(**String**.**valueOf**(**guess**)))** **{**

System**.**out**.**println**(**"Dobrze! Litera jest w słowie."**);**

// Aktualizuj tablicę zgadniętych liter

updateGuessedLetters**(**secretWord**,** guessedLetters**,** guess**);**

**}** **else** **{**

System**.**out**.**println**(**"Nieprawidłowa litera."**);**

attempts**++;**

**}**

// Sprawdź, czy gracz odgadł całe słowo

**if** **(**isWordGuessed**(**guessedLetters**))** **{**

displayWord**(**secretWord**,** guessedLetters**);**

System**.**out**.**println**(**"Gratulacje! Odgadłeś słowo."**);**

**break;**

**}**

**}**

**if** **(**attempts **>=** maxAttempts**)** **{**

System**.**out**.**println**(**"Przegrałeś. Słowo to: " **+** secretWord**);**

**}**

**}**

// Wyświetlanie aktualnego stanu słowa z zamaskowanymi literami

public static void displayWord**(**String secretWord**,** char**[]** guessedLetters**)** **{**

StringBuilder display **=** **new** StringBuilder**();**

**for** **(**char letter **:** secretWord**.**toCharArray**())** **{**

**if** **(**containsLetter**(**guessedLetters**,** letter**))** **{**

display**.**append**(**letter**);**

**}** **else** **{**

display**.**append**(**"\_"**);**

**}**

display**.**append**(**" "**);**

**}**

System**.**out**.**println**(**display**.**toString**());**

**}**

// Aktualizowanie tablicy zgadniętych liter

public static void updateGuessedLetters**(**String secretWord**,** char**[]** guessedLetters**,** char guess**)** **{**

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

**if** **(**secretWord**.**charAt**(**i**)** **==** guess**)** **{**

guessedLetters**[**i**]** **=** guess**;**

**}**

**}**

**}**

// Sprawdź, czy litera jest w tablicy zgadniętych liter

public static boolean containsLetter**(**char**[]** guessedLetters**,** char letter**)** **{**

**for** **(**char guessedLetter **:** guessedLetters**)** **{**

**if** **(**guessedLetter **==** letter**)** **{**

**return** **true;**

**}**

**}**

**return** **false;**

**}**

// Sprawdź, czy gracz odgadł całe słowo

public static boolean isWordGuessed**(**char**[]** guessedLetters**)** **{**

**for** **(**char letter **:** guessedLetters**)** **{**

**if** **(**letter **==** '\0'**)** **{**

**return** **false;**

**}**

**}**

**return** **true;**

**}**

**}**

Oto przykład prostej aplikacji "Wisielec" napisanej w języku Java z wykorzystaniem Android Studio. Ten projekt zawiera interfejs użytkownika z polem EditText do wprowadzania liter, przyciskiem do sprawdzania litery i TextView do wyświetlania informacji o stanie gry.

1. Utwórz nowy projekt w Android Studio z pustą aktywnością.
2. Zaktualizuj plik **activity\_main.xml** w katalogu **res/layout** zgodnie z poniższym kodem:

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="16dp"

tools:context=".MainActivity">

<TextView

android:id="@+id/wordTextView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Słowo: "

android:textSize="24sp"

android:textStyle="bold"/>

<TextView

android:id="@+id/infoTextView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text=""

android:textSize="18sp"/>

<EditText

android:id="@+id/letterEditText"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Wprowadź literę"

android:textSize="18sp"/>

<Button

android:id="@+id/checkButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Sprawdź"

android:textSize="18sp"/>

</LinearLayout>

1. Zaktualizuj plik MainActivity.java w katalogu java zgodnie z poniższym kodem:

package com.example.zadanie6;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;

import java.util.HashSet;

import java.util.Random;

import java.util.Set;

public class MainActivity extends AppCompatActivity {

private static final String[] WORDS = {

"komputer", "programowanie", "system", "algorytm", "aplikacja", "internet", "dane", "kod", "sieć", "web",

"serwer", "framework", "grafika", "programista", "strona", "dysk", "RAM", "CPU", "GPU", "baza danych",

"linux", "Windows", "macOS", "sztuczna inteligencja", "machine learning", "cyberbezpieczeństwo", "komponent",

"API", "git", "repository", "debugowanie", "skrypt", "hasło", "router", "firewall", "monitor", "cyfrowy",

"cryptocurrency", "kodowanie", "architektura", "wersjonowanie", "debugowanie", "interfejs", "responsywność",

"frontend", "backend", "hosting", "cyberatak", "synchronizacja", "dokumentacja", "proxy", "wirus", "malware",

"VPN", "sterownik", "intranet", "siec neuronowa", "framework", "kryptografia", "algorithm", "program",

"framework", "obiekt", "serwer", "komponent", "aplikacja mobilna", "rozwiązanie", "komunikacja", "testowanie",

"konfiguracja", "edycja", "plugin", "archiwum", "algorytm", "adres IP", "usługa", "web design", "klient",

"terminal", "zaszyfrowany", "deklaracja", "monitoring", "renderowanie", "API key", "kompilator", "hardware",

"serwer plików", "debugowanie", "rozpoznawanie", "zarządzanie danymi", "zintegrowany", "data mining",

"sieć społecznościowa", "ciasteczko", "wywołanie", "zabezpieczenie", "operating system", "usługa chmurowa",

"framework", "transmisja", "responsive design", "open-source", "hosting", "platforma", "komunikat", "serializacja",

"aplikacja webowa", "cache", "testing", "storage", "asynchroniczny", "rozwiązanie", "algorytmiczny", "rozpoznawanie",

"klucz API", "synchronizacja", "platforma cyfrowa", "zaszyfrowany", "synchronizacja", "open-source", "profil",

"cache", "komunikacja", "algorithm", "kompilator", "zabezpieczenie"

};;

private static final int MAX\_ATTEMPTS = 7;

private String secretWord;

private char[] guessedLetters;

private int attempts;

private Set<Character> uniqueCharacters;

private TextView wordTextView;

private TextView infoTextView;

private EditText letterEditText;

private Button checkButton;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

wordTextView = findViewById(R.id.wordTextView);

infoTextView = findViewById(R.id.infoTextView);

letterEditText = findViewById(R.id.letterEditText);

checkButton = findViewById(R.id.checkButton);

secretWord = chooseRandomWord();

guessedLetters = new char[secretWord.length()];

attempts = 0;

uniqueCharacters = new HashSet<>();

displayWord();

checkButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

if (attempts >= MAX\_ATTEMPTS) {

infoTextView.setText("Przegrałeś. Słowo to: " + secretWord);

} else {

String input = letterEditText.getText().toString().trim();

if (input.length() != 1) {

infoTextView.setText("Podaj jedną literę.");

} else {

char guess = input.charAt(0);

if (uniqueCharacters.contains(guess)) {

infoTextView.setText("Już zgadłeś tę literę.");

} else {

uniqueCharacters.add(guess);

if (checkGuess(guess)) {

infoTextView.setText("Dobrze! Litera jest w słowie.");

} else {

infoTextView.setText("Nieprawidłowa litera.");

attempts++;

}

if (isWordGuessed()) {

infoTextView.setText("Gratulacje! Odgadłeś słowo.");

}

}

displayWord();

}

}

}

});

}

private String chooseRandomWord() {

Random random = new Random();

int index = random.nextInt(WORDS.length);

return WORDS[index];

}

private void displayWord() {

StringBuilder display = new StringBuilder();

for (char letter : secretWord.toCharArray()) {

if (containsLetter(letter)) {

display.append(letter);

} else {

display.append("\_");

}

display.append(" ");

}

wordTextView.setText("Słowo: " + display.toString());

}

private boolean containsLetter(char letter) {

for (char guessedLetter : guessedLetters) {

if (guessedLetter == letter) {

return true;

}

}

return false;

}

private boolean checkGuess(char guess) {

boolean correctGuess = false;

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

if (secretWord.charAt(i) == guess) {

guessedLetters[i] = guess;

correctGuess = true;

}

}

return correctGuess;

}

private boolean isWordGuessed() {

for (char letter : guessedLetters) {

if (letter == '\0') {

return false;

}

}

return true;

}

}

Ten przykładowy projekt Android Studio zawiera interfejs użytkownika z polem EditText do wprowadzania liter, przyciskiem "Sprawdź" oraz dwoma TextView do wyświetlania stanu gry. Główna logika gry została zaimplementowana w klasie MainActivity. Gracz próbuje odgadnąć tajne słowo, a program obsługuje logikę gry, wyświetlanie stanu gry oraz obsługę zdarzeń, takie jak zakończenie gry.

**Analiza dziedziny dla kodu przykładu gry w "Wisielca" dla Androida**

1. **Gra Wisielec (Hangman)**
   * Gra typu zgadywanka, w której gracz próbuje odgadnąć tajne słowo, zgadywając litery.
2. **Komponenty interfejsu użytkownika**
   * **wordTextView**: Wyświetla tajne słowo z zamaskowanymi literami.
   * **infoTextView**: Wyświetla komunikaty o stanie gry.
   * **letterEditText**: Pozwala graczowi wprowadzać litery.
   * **checkButton**: Przycisk do sprawdzania litery.
3. **Zmienne i struktury danych**
   * **secretWord**: Zmienna przechowująca tajne słowo do odgadnięcia.
   * **guessedLetters**: Tablica przechowująca odgadnięte litery w tajnym słowie.
   * **attempts**: Licznik prób gracza.
   * **uniqueCharacters**: Zbiór przechowujący unikalne litery, które gracz już odgadł.
   * **String[] WORDS**: Stała tablica zawierająca tajne słowa.
4. **Funkcje i metody**
   * **chooseRandomWord()**: Metoda losująca tajne słowo z tablicy **WORDS**.
   * **displayWord()**: Metoda wyświetlająca aktualny stan tajnego słowa z zamaskowanymi literami.
   * **containsLetter(char letter)**: Metoda sprawdzająca, czy litera jest już odgadnięta.
   * **checkGuess(char guess)**: Metoda sprawdzająca, czy litera odgadnięta przez gracza jest prawidłowa.
   * **isWordGuessed()**: Metoda sprawdzająca, czy gracz odgadł całe tajne słowo.
5. **Główne etapy gry**
   * Inicjalizacja gry: Losowanie tajnego słowa i przygotowanie interfejsu.
   * Wprowadzanie litery: Gracz wprowadza literę w polu **letterEditText** i klika przycisk **checkButton**.
   * Sprawdzenie litery: Program sprawdza, czy litera jest w słowie i aktualizuje stan gry.
   * Wyświetlanie wyników: Wyniki gry, takie jak odgadnięte litery i komunikaty, są wyświetlane w **wordTextView** i **infoTextView**.
   * Zakończenie gry: Gra może zakończyć się sukcesem (odgadnięcie słowa) lub porażką (przekroczenie limitu prób).
6. **Logika gry**
   * Gracz próbuje odgadnąć tajne słowo, wprowadzając litery.
   * Program sprawdza, czy litery są w słowie i aktualizuje stan gry.
   * Gra może zakończyć się sukcesem lub porażką.
7. **Słownik słów**
   * **WORDS** to stała tablica zawierająca tajne słowa, z których wybierane jest jedno losowo na początku gry.