UNIWERSYTET RZESZOWSKI
WYDZIAŁ NAUK ŚCISŁYCH I TECHNICZNYCH
INSTYTUT INFORMATYKI

Tomasz Wiśniewski
134986
Informatyka

System zarządzania turniejami e-sportowymi

Praca projektowa

Praca wykonana pod kierunkiem
mgr inż Ewa Żesławska

Rzeszów 2025

Spis treści

Spis rysunków

Spis tabel

Streszczenie

Niniejszy projekt opisuje proces tworzenia oraz implementacji aplikacji desktopowej "System zarządzania turniejami e-sportowymi". Aplikacja została napisana w języku Java z wykorzystaniem biblioteki Swing do budowy graficznego interfejsu użytkownika. Głównym celem systemu jest automatyzacja i uproszczenie procesu zarządzania rozgrywkami turniejowymi. Projekt wykorzystuje architekturę warstwową, w tym wzorzec DAO do komunikacji z relacyjną bazą danych SQLite za pośrednictwem JDBC, a kluczowym elementem implementacji jest zastosowanie hierarchii dziedziczenia do modelowania różnych typów drużyn.

Abstract

This project describes the development and implementation process of a desktop application, "Esports Tournament Management System". The application was written in the Java language using the Swing library for the graphical user interface. The main objective of the system is to automate and simplify the management of tournament gameplay. The project utilizes a layered architecture, including the DAO pattern for communication with a relational SQLite database via JDBC. A key element of the implementation is the use of an inheritance hierarchy to model different types of teams participating in the tournament.

1 Opis założeń projektu

Celem projektu było stworzenie w pełni funkcjonalnego systemu informatycznego do zarządzania turniejami e-sportowymi. Głównym problemem, który aplikacja ma rozwiązywać, jest brak prostych i dostępnych narzędzi dla organizatorów amatorskich lub lokalnych rozgrywek. Obecnie osoby te często polegają na skomplikowanych arkuszach kalkulacyjnych lub ręcznych zapiskach, co jest nieefektywne i podatne na błędy. Problem ten jest ważny, ponieważ scena amatorskiego e-sportu dynamicznie się rozwija, a bariery organizacyjne ograniczają jej potencjał.

Aby rozwiązać ten problem, niezbędne było stworzenie aplikacji, która automatyzuje kluczowe procesy. Realizacja projektu przebiegała krok po kroku: od zaprojektowania architektury i bazy danych, przez implementację logiki zarządzającej turniejem, aż po stworzenie intuicyjnego interfejsu graficznego. Wynikiem prac jest samodzielna aplikacja desktopowa, która minimalizuje nakład pracy organizatora i zapewnia integralność oraz poprawność przebiegu rozgrywek.

Aplikacja została zaprojektowana z myślą o turniejach w popularnej grze e-sportowej Valorant. W standardowym meczu rywalizują dwie pięcioosobowe drużyny, a celem jest wygranie 13 rund. Pojedynczą rundę można wygrać poprzez wyeliminowanie wszystkich przeciwników lub przez podłożenie i zdetonowanie bomby (zwanej "Spike") przez stronę atakującą. Mecz kończy się, gdy jedna z drużyn jako pierwsza osiągnie 13 punktów, co prowadzi do wyników takich jak 13-10 czy 13-5. Zrozumienie tej prostej zasady punktacji jest kluczowe dla kontekstu działania aplikacji, która pozwala na zapisywanie i przetwarzanie właśnie takich wyników.

1.1 Wymagania funkcjonalne

- **Zarządzanie danymi turnieju:** System pozwala na dodawanie i usuwanie drużyn różnych typów (Profesjonalna, Amatorska, etc.) oraz przypisanych do nich graczy.
- **Automatyzacja rozgrywki:** System automatycznie generuje pary meczowe w poszczególnych etapach turnieju, uwzględniając losowość i obsługując przypadki nieparzystej liczby drużyn poprzez mechanizm "wolnego losu".
- **Zarządzanie wynikami:** Interfejs umożliwia wprowadzanie wyników zakończonych meczów.
- **Przebieg turnieju:** Aplikacja automatycznie zarządza przechodzeniem do kolejnych etapów po zakończeniu rundy, włączając w to logikę meczu o 3. miejsce.
- **Funkcje administracyjne:** System posiada funkcję całkowitego resetu turnieju, która usuwa wszystkie dane z bazy.
- **Generowanie raportów:** Aplikacja umożliwia wgląd w historię wszystkich rozegranych meczów oraz prezentuje finalny ranking po zakończeniu turnieju.

1.2 Wymagania niefunkcjonalne

- **Wydajność:** Operacje interfejsu oraz podstawowe zapytania do bazy danych powinny być wykonywane w czasie poniżej 2 sekund, zapewniając płynność działania.
- **Użyteczność:** Interfejs użytkownika został zaprojektowany w sposób prosty i intuicyjny, aby był zrozumiały dla użytkownika bez wiedzy technicznej.

- **Niezawodność:** Aplikacja jest odporna na podstawowe błędy użytkownika (np. próba startu turnieju bez drużyn) i obsługuje je, wyświetlając odpowiednie komunikaty.
- **Przenośność:** Program napisany w Javie jest przenośny i działa na każdym systemie operacyjnym (Windows, macOS, Linux), na którym zainstalowano środowisko JRE.
- **Utrzymywalność:** Kod źródłowy został podzielony na logiczne warstwy (model, DAO, core, ui), co ułatwia jego zrozumienie, konserwację i ewentualny dalszy rozwój.

2 Opis struktury projektu

Projekt został zrealizowany w języku Java z wykorzystaniem biblioteki Swing do stworzenia graficznego interfejsu użytkownika. Warstwa persystencji danych opiera się na plikowej bazie SQLite, z którą aplikacja komunikuje się poprzez API JDBC.

2.1 Architektura aplikacji

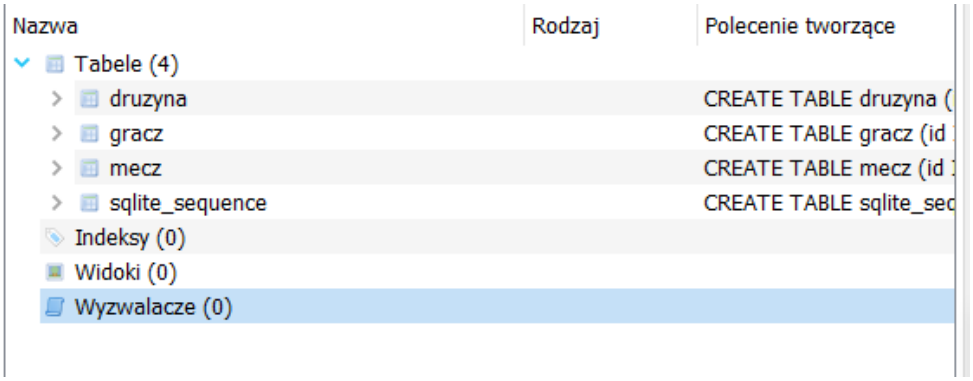
Struktura projektu została oparta na architekturze warstwowej w celu separacji poszczególnych logik:

- **Pakiet `model`** - zawiera klasy danych, w tym abstrakcyjną klasę `Druzyna` oraz jej cztery konkretne podklasy (`DruzynaProfesjonalna`, `DruzynaAmatorska`, etc.), które tworzą wymaganą hierarchię dziedziczenia.
- **Pakiet `dao`** - (Data Access Object) - Zestaw klas odpowiedzialnych za całą komunikację z bazą danych, hermetyzujących zapytania SQL.
- **Pakiet `core`** - Rdzeń aplikacji, zawierający główną logikę biznesową w klasie `TournamentManager`
- **Pakiet `ui`** - Warstwa prezentacji, odpowiedzialna za graficzny interfejs użytkownika (`TournamentGUI`).

2.2 Schemat bazy danych

System wykorzystuje plikową bazę danych SQLite do przechowywania danych. Składa się ona z trzech głównych tabel:

1. **druzyna**: Przechowuje informacje o drużynach (id, nazwa, typ, sponsor, miasto, itd.).
2. **gracz**: Przechowuje informacje o graczach (id, nick, druzyna_id).
3. **mecz**: Przechowuje historię rozegranych meczów (id, id drużyn, wyniki, zwycięzca).



Rysunek 1: Struktura tabel w bazie danych `turniej.db`.

2.3 Wymagania sprzętowe i oprogramowanie

- **System operacyjny:** Windows, macOS lub Linux.
- **Oprogramowanie:** Zainstalowane środowisko Java Development Kit (JDK) w wersji 11 lub nowszej.
- **Biblioteki zewnętrzne:** Do poprawnego działania aplikacji wymagane jest dołączenie do struktury projektu sterownika JDBC dla SQLite. W projekcie wykorzystano plik **`sqlite-jdbc-3.40`** który jest dołączony do plików źródłowych.
- **Pamięć RAM:** Minimum 256 MB.
- **Miejsce na dysku:** Około 50 MB.

3 Harmonogram realizacji projektu

3.1 Przebieg prac

Realizacja projektu została podzielona na cztery główne fazy, rozłożone w czasie. Poniższa tabela przedstawia uproszczony harmonogram prac, który stanowi podstawę do stworzenia diagramu Gantta.

Faza projektu	**Główne zadania**
1. Analiza i Projekt	Analiza wymagań, projekt architektury, schemat bazy danych.
2. Implementacja Rdzenia	Kodowanie modelu, DAO, logiki w `TournamentManager`.
3. Implementacja GUI	Budowa interfejsu w Swing, połączenie z logiką.
4. Testowanie i Dokumentacja	Testowanie funkcjonalności, poprawa błędów, pisanie dokumentacji.

Tabela 1: Uproszczony harmonogram realizacji projektu.

3.2 Problemy napotkane podczas realizacji

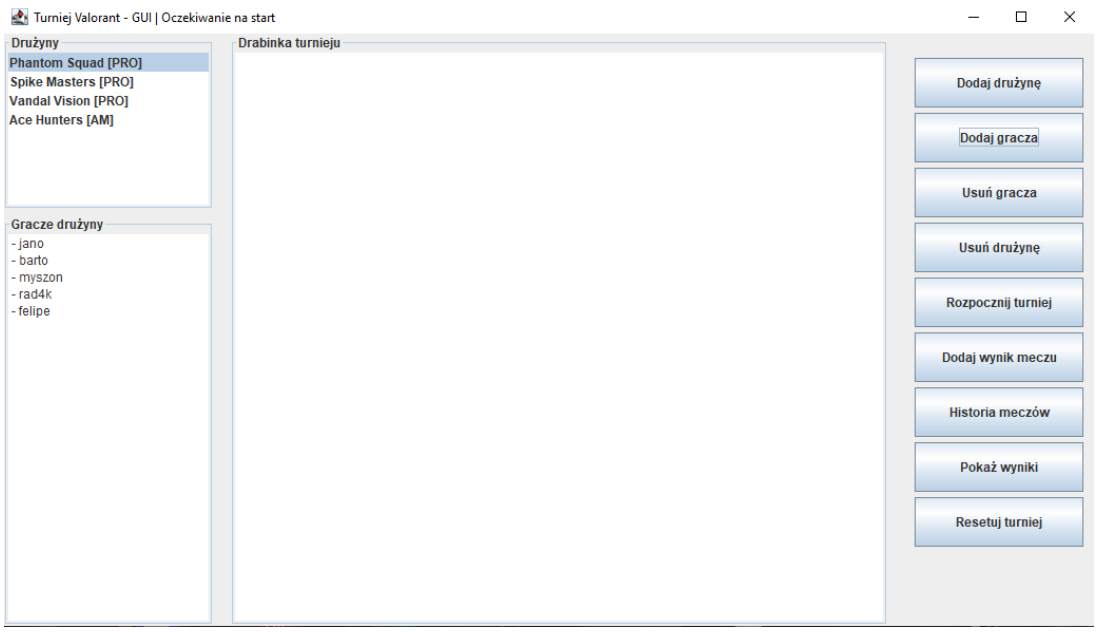
W trakcie implementacji napotkano kilka wyzwań logicznych. Największym z nich była poprawna obsługa przypadków brzegowych w logice turnieju, takich jak start z nieparzystą liczbą drużyn. Początkowa wersja algorytmu błędnie obsługiwała "wolny los", co wymagało gruntownej przebudowy metody zarządzającej przejściem między etapami. Kolejnym problemem była konfiguracja połączenia z bazą danych, co objawiało się błędem "No suitable driver found" i zostało rozwiązane poprzez poprawne dodanie sterownika JDBC do zależności projektu.

3.3 System kontroli wersji

Kod źródłowy projektu był zarządzany przy użyciu systemu kontroli wersji Git. Repozytorium publiczne projektu znajduje się pod adresem: https://github.com/Wisniewsky/programowanie

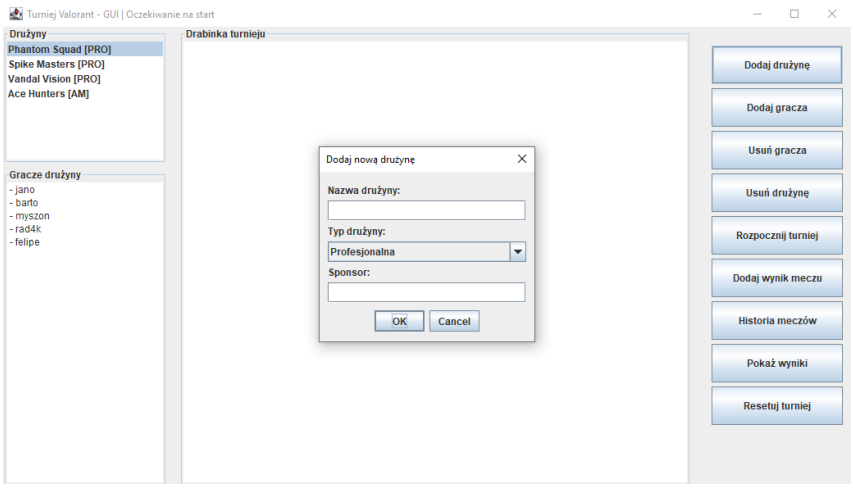
4 Prezentacja warstwy użytkowej projektu

Aplikacja została zaprojektowana z myślą o prostocie obsługi. Główny interfejs podzielony jest na trzy części: panel zarządzania po lewej, drabinkę turnieju na środku i panel akcji po prawej.



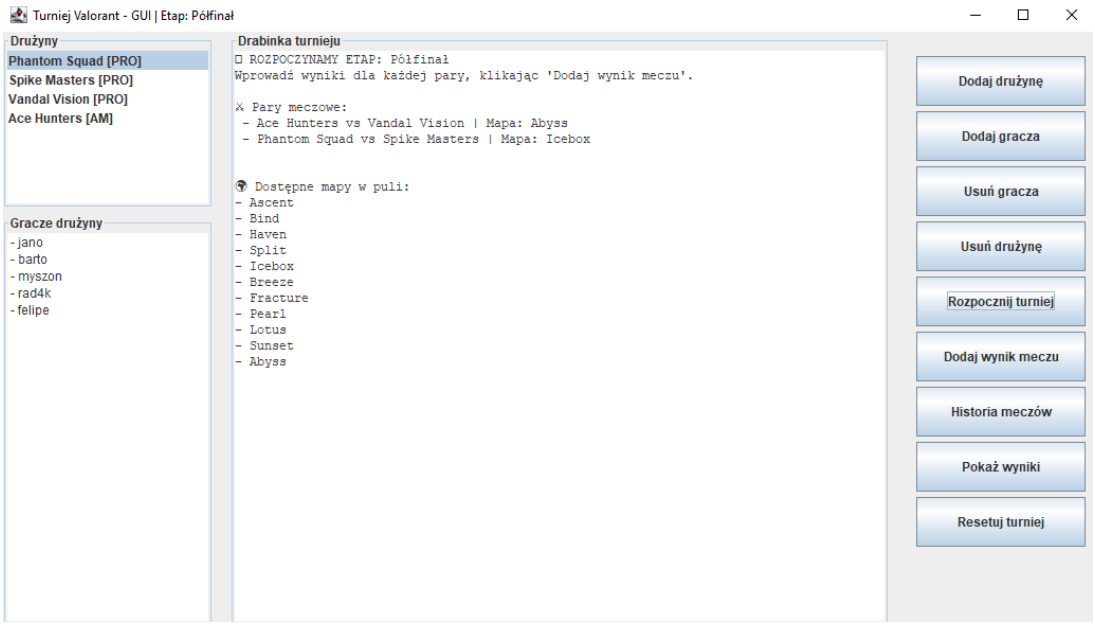
Rysunek 2: Główne okno aplikacji po uruchomieniu.

Aby dodać drużynę, należy kliknąć przycisk "Dodaj drużynę" i wypełnić formularz, który pojawi się w nowym oknie.



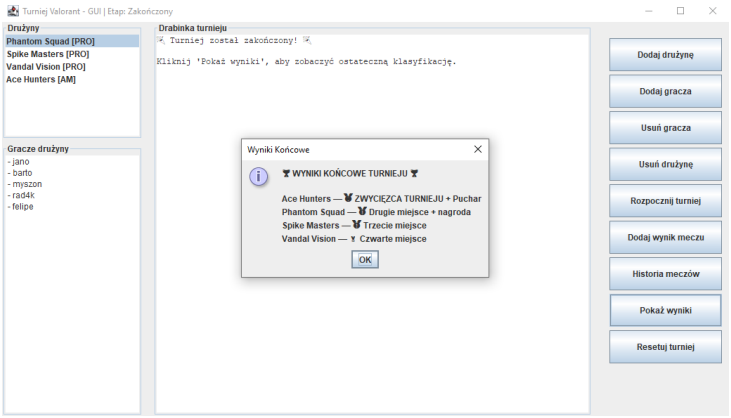
Rysunek 3: Formularz dodawania nowej drużyny typu "Profesjonalna".

Po dodaniu co najmniej trzech drużyn, można rozpocząć turniej przyciskiem "Rozpocznij turniej", co spowoduje wygenerowanie pierwszego etapu, wylosowanie par i puli map.



Rysunek 4: Widok drabinki turniejowej po wylosowaniu par i puli map.

Wprowadzanie wyników odbywa się za pomocą przycisku "Dodaj wynik meczu". Po zakończeniu rundy i całego turnieju, możliwe jest wyświetlenie okna z podsumowaniem wyników.



Rysunek 5: Okno prezentujące końcowe wyniki turnieju.

5 Podsumowanie

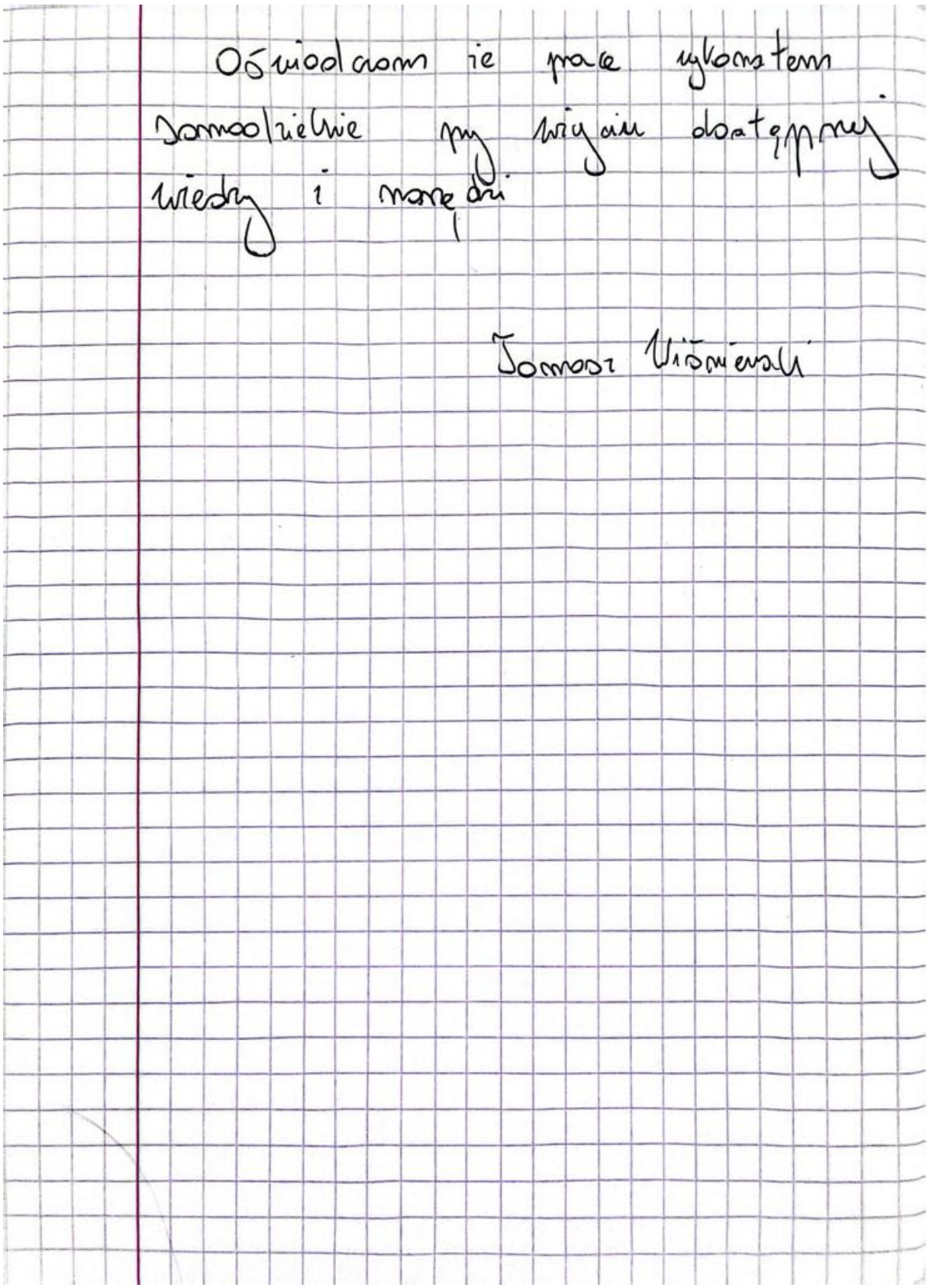
Zrealizowany projekt z powodzeniem implementuje system do zarządzania turniejami e-sportowymi. Aplikacja spełnia wszystkie założone wymagania funkcjonalne, oferując stabilne i logicznie działające środowisko do prowadzenia rozgrywek. Najważniejszym osiągnięciem projektowym jest implementacja elastycznej logiki turniejowej oraz zbudowanie obiektowej architektury opartej na hierarchii dziedziczenia.

Aplikacja stanowi solidną bazę do dalszego rozwoju. W przyszłości można by ją rozbudować o takie funkcjonalności jak: pełna implementacja operacji CRUD (w tym edycja danych), import i eksport danych z/do plików CSV, czy też bardziej zaawansowane statystyki graczy i drużyn.

Bibliografia

1. Materiały dydaktyczne z zajęć "Programowanie Obiektowe Java", mgr inż Ewa Żesławska, Uniwersytet Rzeszowski, 2025.

A Oświadczenie o samodzielności pracy



Rysunek 6: Oświadczenie o samodzielności wykonania pracy.