**Politechnika Wrocławska**
**Wydział Informatyki i Telekomunikacji**

Kierunek: **INA**
Specjalność: **-**

# Praca Dyplomowa Inżynierska

## Serwis do przeprowadzania rozgrywek Rummikub

Adam Bednarz

Opiekun pracy
**dr inż. Jakub Lemiesz**

Wrocław 2022

## Streszczenie

Tutaj tekst streszczenia po polsku.

## Abstract

Tutaj treść streszczenia po angielsku.

# Spis treści

# Wstęp

Celem pracy jest zaprojektowanie i implementacja serwisu informatycznego, który umożliwia przeprowadzanie gier Rummikub.

Założenia funkcjonalne serwisu:

- autoryzacja użytkowników,
- zarządzanie instancjami gier,
- korzystanie z serwisu za pomocą strony internetowej lub aplikacji na telefon,
- rozgrywka z botem.

Obecnie istnieją już takie rozwiązania, które dostarczają nam podobne funkcjonalności. Jednakże motywacją tej pracy jest bliższe zapoznanie się z procesem tworzenia takiego serwisu, jak również analiza złożoności gry Rummikub. Czynnikiem wyróżniającym się stworzonego serwisu jest umożliwienie graczom uczestniczenie w danej rozgrywce niezależnie od tego czy korzystają z aplikacji na telefon (iOS, Android), czy też ze strony internetowej w przeglądarce.

Praca składa się z czterech rozdziałów. W rozdziale pierwszym omówiono grę Rummikub, jej zasady oraz przebieg rozgrywki. W rozdziale drugim zobrazowano szczegółowy projekt serwisu. Do graficzego przedstawienia wykorzystano notacje UML. W rozdziale trzecim przedstawiono implementację serwisu. Opisane zostały użyte technologie oraz języki programowania, które zostały użyte w projekcie informatycznym. Przedstawiono dokumentację techniczną modułów systemu oraz schemat bazy danych.

# Rozdział 1

# Opis gry Rummikub

## 1.1 Gra

Została stworzona przez Ephraima Hertzano i po raz pierwszy wydano ją w 1950 roku. W rozgrywce może uczestniczyć od dwóch do czterech graczy. Gra składa się z kości, które są numerowane od 1 do 13. Występują one w czterech kolorach (pomarańczowy, czerwony, niebieski, czarny). Każda kość o danym kolorze i liczbie występuje dwa razy. Ponadto występują dwie specjalne kości jako jokery. Łącznie gra zawiera 106 kości.

Gra polega na wykładaniu grup, bądź serii. Grupa to trzy lub cztery kości o różnych barwach, ale z tą samą liczbą, natomiast seria to co najmniej trzy kolejne kości o tym samym kolorze.

W przypadku braku odpowiednich kości do gry można posłużyć się dwoma zestawami kart po 52 karty i 2 jokery. Gdzie 1, 11, 12, 13 można zastąpić odpowednio asem, waletem, damą i królem.

## 1.2 Rozgrywka

Zbiór wszystkich wymieszanych kości nazywany jest bankiem. Z niego na początku gry każdy gracz otrzymuje 14 kości. W przypadku pierwszego ruchu należy wyłożyć własne kości o sumie numerów tych kości co najmniej 30, bez możliwości modyfikowania kości leżących na planszy.

Po wyłożeniu pierwszego ruchu gracz może modyfikować inne wyłożone wcześniej układy. Dozwolone jest przebudowywanie układów kości (rozbijanie lub rozbudowywanie). Jednak w każdym ruchu należy wyłożyć przynajmniej jedną własną kość.

Na każdy ruch przypada ustalony wcześniej limit czasowy. Po jego upływie w przypadku braku wyłożenia kości przez gracza, lub gdy stan planszy nie spełnia zasad gry, gracz pobiera z banku jedną kość i cofa wprowadzone zmiany układów kości.

Joker w grze Rummikub symbolizuje dowolną kość. Można nim zastąpić brakujący element do utworzenia układu kości. Joker ma taką samą wartość jak kość, którą zastępuje. Można zabrać wyłożonego jokera zastępując go odpowiednią kością i wykorzystać go w innym miejscu na planszy.

Gra kończy się w momencie, gdy któryś z graczy wyłoży wszystkie swoje kości lub gdy zabraknie kości w banku. W przypadku drugim wygrywa ten gracz, który ma najmniejszą sumę liczb znajdujących się na kościach.

# Rozdział 2

# Projekt serwisu

W tym rozdziale przedstawiono szczegółowy projekt systemu w notacji UML uwzględniający wymagania funkcjonalne opisane w rozdziale 1. Do opisu relacji pomiędzy składowymi systemu wykorzystano diagramy ....

## 2.1 Założenia

W serwisie wykorzystano architekturę klient-serwer. Istnieje jeden serwer, do którego może podłączać się wiele klientów i odpowiada on za komunikację i zarządzanie danymi. W komponencie klienta wykorzystano architekturę trójwarstwową. Architektura ta dzieli komponent na trzy osobne części:

- warstwa prezentacji,
- warstwa biznesowa,
- warstwa danych.

Warstwa prezentacji jest to interfejs graficzny użytkownika. Jest odpowiedzialna za interakcję z użytkownikiem (wyświetlanie i wprowadzanie danych).

Warstwa biznesowa odpowiada za przetwarzanie komunikatów od użytkownika lub ze strony serwera. Tutaj zawarta jest wszelka logika aplikacji. Przetworzone dane są przekazywane do warstwy prezentacji i/lub warstwy danych. Warstwa ta jest łącznikiem pomiędzy warstwą prezentacji, a warstwą danych.

Warstwa danych jest dostępem do danych. Obsługuje połączenie aplikacji z zewnętrznym obiektem dostarczającym dane (baza danych, serwer).

## 2.2 Przypadki użycia i scenariusze

W tej sekcji należy przedstawić przypadki użycia oraz odpowiadające im scenariusze dla poszczególnych grup użytkowników ....

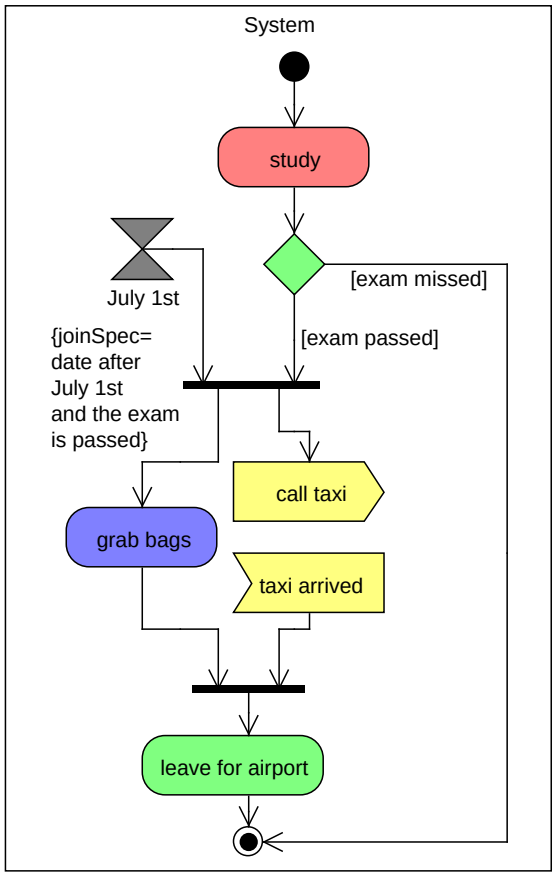
## 2.3 Diagramy klas

W tej sekcji należy przedstawić diagramy klas dla odpowiednich elementów systemu zidentyfikowane na podstawie wcześniejszych rozważań

## 2.4 Diagramy aktywności

W tej sekcji należy przedstawić diagramy aktywności dla elementów systemu i odpowiednich procesów wynikające z wcześniejszej analizy.

W niniejszym rozdziale przedstawiono diagramy aktywności .... Diagram na rysunku 2.1 przedstawia ....




Rysunek 2.1: Diagram aktywności związany z procesem rejestracji dokumentu.

## 2.5 Diagramy sekwencji

W tej sekcji należy przedstawić diagramy sekwencji dla obiektów systemu zidentyfikowanych na podstawie wcześniejszych rozważań. Należy wykorzystać nazewnictwo wprowadzone w poprzednich rozdziałach, w szczególności odpowiadające definicjom wprowadzonych klas.

## 2.6 Diagramy stanów

W tej sekcji należy przedstawić diagramy stanów w których może znaleźć się system. Diagramy te są szczególnie istotne przy projektowaniu systemów czasu rzeczywistego.

## 2.7 Projekt bazy danych

W tej sekcji należy przedstawić projekt bazy danych. Należy omówić wycinek rzeczywistości i odpowiadające mu zidentyfikowane elementy systemu, których wartości będą podlegać utrwalaniu. Należy przedyskutować wybór typów danych dla atrybutów poszczególnych obiektów. Należy uzasadnić wybór platformy DBMS. Dla relacyjnych baz danych należy przedyskutować jej normalizację.

## 2.8 Opis protokołów

W tej sekcji należy omówić protokoły wykorzystywane przez komponenty systemu. Omówić formaty komunikatów i zilustrować je przykładami.

## 2.9 Opis algorytmów

W tej sekcji należy wymienić i przedyskutować algorytmy wykorzystywane w systemie. Algorytmy należy przedstawić w pseudokodzie (wykorzystać pakiet `algorithm2e`). Omówienia poszczególnych kroków algorytmów powinny zawierać odwołania do odpowiednich linii pseudokodu. Dla zaproponowanych autorskich algorytmów należy przeprowadzić analizę ich złożoności czasowej i pamięciowej.

Algorytm bąblowania jest przedstawiony w Pseudokodzie 2.1.

**Pseudokod 2.1:** Wyporność przez bąblowanie

```
   Input: Zbiór bąbli B
   Output: Wyporność W
1  foreach b ∈ B do
2      Process(b);
3      for i ← 1 to |B| do
4          if Calculate(EW(i,b)) ⩽ 0 then
5              b ← 2 ∗ b;
6  while B ≠ ∅ do
7      for j ← 1 to |B| do
8          if Calculate(FT(j,b̂)) ⩽ 0 then
9              w ← 2 ∗ b̂;
10             W ← W ∪ {w};
11             B ← B \ {b};
```

# Rozdział 3

# Implementacja serwisu

## 3.1 Opis technologii

Należy tutaj zamieścić krótki opis (z referencjami) do technologii użytych przy implementacji systemu.

Do implementacji systemu użyto języka JAVA w wersji ..., szczegółowy opis można znaleźć w [1]. Interfejs zaprojektowano w oparciu o HTML5 i CSS3 [3].

## 3.2 Omówienie kodów źródłowych

Kod źródłowy 3.1 przedstawia opisy poszczególnych metod interfejsu: `WSPodmiotRejestracjaIF`. Kompletne kody źródłowe znajdują się na płycie CD dołączonej do niniejszej pracy w katalogu `Kody` (patrz Dodatek A).

Kod źródłowy 3.1: Interfejs usługi Web Service: `WSPodmiotRejestracjaIF`.

```
package erejestracja.podmiot;
import java.rmi.RemoteException;
// Interfejs web serwisu dotyczącego obsługi podmiotów i rejestracji.
public interface WSPodmiotRejestracjaIF extends java.rmi.Remote{
// Pokazuje informacje o danym podmiocie.
// parametr: nrPeselRegon – numer PESEL podmiotu lub numer REGON firmy.
// return: Podmiot – obiekt transportowy: informacje o danym podmiocie.
public Podmiot pokazPodmiot(long nrPeselRegon) throws RemoteException;
// Dodaje nowy podmiot.
// parametr: nowyPodmiot – obiekt transportowy: informacje o nowym podmiocie.
// return: true – jeśli podmiot dodano, false – jeśli nie dodano.
public boolean dodajPodmiot(Podmiot nowyPodmiot) throws RemoteException;
// Usuwa dany podmiot.
// parametr: nrPeselRegon – numer PESEL osoby fizycznej lub numer REGON firmy.
// return: true – jeśli podmiot usunięto, false – jeśli nie usunięto.
public boolean usunPodmiot(long nrPeselRegon) throws RemoteException;
// Modyfikuje dany podmiot.
// parametr: podmiot – obiekt transportowy: informacje o modyfikowanym podmiocie.
// return: true – jeśli podmiot zmodyfikowano, false – jeśli nie zmodyfikowano.
public boolean modyfikujPodmiot(Podmiot podmiot) throws RemoteException;
// Pokazuje zarejestrowane podmioty na dany dowód rejestracyjny.
// parametr: nrDowoduRejestracyjnego – numer dowodu rejestracyjnego.
// return: PodmiotRejestracja[] – tablica obiektów transportowych: informacje o
// wszystkich zarejestrowanych podmiotach.
public PodmiotRejestracja[] pokazZarejestrowanePodmioty(
String nrDowoduRejestracyjnego) throws RemoteException;
// Nowa rejestracja podmiotu na dany dowód rejestracyjny.
// parametr: nrDowoduRejestracyjnego – numer dowodu rejestracyjnego.
```

```
// parametr: nrPeselRegon - numer PESEL podmiotu lub numer REGON firmy.
// parametr: czyWlasciciel - czy dany podmiot jest właścicielem pojazdu.
// return: true - jeśli zarejestrowano podmiot, false - jeśli nie zarejestrowano.
public boolean zarejestrujNowyPodmiot(String nrDowoduRejestracyjnego,
long nrPeselRegon, boolean czyWlasciciel) throws RemoteException;
// Usuwa wiązanie pomiędzy danym podmiotem, a dowodem rejestracyjnym.
// parametr: nrDowoduRejestracyjnego - numer dowodu rejestracyjnego.
// parametr: nrPeselRegon - numer PESEL podmiotu lub numer REGON firmy.
// return: true - jeśli podmiot wyrejestrowano, false - jeśli nie wyrejestrowano.
public boolean wyrejestrujPodmiot(String nrDowoduRejestracyjnego,
long nrPeselRegon) throws RemoteException;
```

Kod źródłowy 3.2 przedstawia procedurę przetwarzającą żądanie. Hasz utrwalany `%granulacja` wykorzystywany jest do komunikacji międzyprocesowej.

Kod źródłowy 3.2: Przetwarzanie żądania - procedura `process_req()`.

```
sub process_req(){
  my($r) = @_;
  $wyn = "";
  if ($r=~/get/i) {
        @reqest = split("_",$r);
        $zad = $reqest[0];
        $ts1 = $reqest[1];
        $ts2 = $reqest[2];
        @date1 = split(/\D/,$ts1);
        @date2 = split(/\D/,$ts2);
        print "odebralem:_$r";
        $wyn = $wyn."zadanie:_$zad\n";
        $wyn = $wyn."czas_od:_"."$date1[0]"."-"."$date1[1]"."-"."$date1[2]"."_"."$date1[3]".":"."$
        $wyn = $wyn."czas_do:_"."$date2[0]"."-"."$date2[1]"."-"."$date2[2]"."_"."$date2[3]".":"."$
        $wyn = $wyn.&sym_sens($ts1,$ts2);
        return $wyn;
  }
  if ($r=~/set gt/i) {
        @reqest = split("_",$r);
        $zad = $reqest[0];
        $ts1 = $reqest[1];
        $ts2 = $reqest[2];
        $gt = $reqest[2];
        dbmopen(%granulacja,"granulacja_baza",0644);
        $granulacja{"gt"}=$gt;
        dbmclose(%granulacja);
        $wyn = "\'GT\'_zmienione_na:_$gt";
  }
}
```

# Rozdział 4

# Bot

W tym rozdziale należy omówić zawartość pakietu instalacyjnego oraz założenia co do środowiska, w którym realizowany system będzie instalowany. Należy przedstawić procedurę instalacji i wdrożenia systemu. Czynności instalacyjne powinny być szczegółowo rozpisane na kroki. Procedura wdrożenia powinna obejmować konfigurację platformy sprzętowej, OS (np. konfiguracje niezbędnych sterowników) oraz konfigurację wdrażanego systemu, m.in. tworzenia niezbędnych kont użytkowników. Procedura instalacji powinna prowadzić od stanu, w którym nie są zainstalowane żadne składniki systemu, do stanu w którym system jest gotowy do pracy i oczekuje na akcje typowego użytkownika.

# Podsumowanie

W podsumowaniu należy określić stan zakończonych prac projektowych i implementacyjnych. Zaznaczyć, które z zakładanych funkcjonalności systemu udało się zrealizować. Omówić aspekty pielęgnacji systemu w środowisku wdrożeniowym. Wskazać dalsze możliwe kierunki rozwoju systemu, np. dodawanie nowych komponentów realizujących nowe funkcje.

W podsumowaniu należy podkreślić nowatorskie rozwiązania zastosowane w projekcie i implementacji (niebanalne algorytmy, nowe technologie, itp.).

# Bibliografia

[1] Java technology. Web pages: http://www.oracle.com/technetwork/java/.

[2] J. Cichoń, M. Klonowski, Łukasz Krzywiecki, B. Różański, P. Zieliński. On the number of nodes and their distribution in chord. 2006.

[3] B. Frain. *Responsive Web Design with HTML5 and CSS3*. Packt Publishing, 2012.

# Załącznik A

# Zawartość płyty CD

W tym rozdziale należy krótko omówić zawartość dołączonej płyty CD.