

Politechnika Świętokrzyska w Kielcach Wydział Elektrotechniki, Automatyki i Informatyki Katedra Informatyki, Elektroniki i Elektrotechniki		
Kierunek: Informatyka	Przedmiot: Programowanie Obiektowe 2- Projekt	
Grupa dziekańska: 2ID13B Rok: 2023/2024r.	Temat projektu: Strona z grami	Grupa: Dawid Wójcikowski Piotr Zieliński Mateusz Wrenk

Opis Projektu:

Projekt „Strona z grami” został wykonany w języku programowania Java 11, w środowisku NetBeans. Projekt łączy ze sobą język Java, oraz SQL (do obsługi bazy danych), oraz technologie JavaServer Pages (JSP), która umożliwia osadzanie kodu Java bezpośrednio w kodzie HTML, co pozwala na generowanie dynamicznych stron internetowych.

Ponadto, język JavaScript jest używany do obsługi komunikacji z serwerem przy użyciu WebSocket.

Skład Zespołu:

Dawid Wójcikowski- Gra karciana “Mniejsze-Większe”, obsługa bazy danych, wygląd strony, obsługa połączenia serwer-klient.

Piotr Zieliński- Gra “Kości” na dwóch graczy, grafiki do gier (kości i karty), testy.

Mateusz Wrenk- Gra karciana „BlackJack”, opis projektu, dokumentacja.

Funkcjonalność Projektu:

Projekt jest stroną z grami, gdzie gracze mogą założyć konto, logować i prowadzić rywalizację punktacyjną w dostępnych grach.

Projekt posiada trzy gry:

-„**Kości**”; gdzie dwójka graczy losuje na zmianę 5 kostek sześciościennych oraz zapisuje wyniki do swoich dwóch tabel, według określonych zasad, gracz z największą ilością punktów- wygrywa a jego wynik jest zapisywany do tablicy wyników.

-„**Mniejsze-Większe**”; gdzie gracz widzi wylosowaną z talii kartę i ma za zadanie zgadnięcie czy następna karta jest większa- czy mniejsza od pokazanej karty, największy wynik gracza jest zapisywany do tabeli wyników. Gra kończy się gdy gracz wyjdzie z gry, lub skończy całą talie kart.

-„**BlackJack**”; gdzie gracz gra przeciwko krupierowi (komputer, który tylko dobiera losowo karty i trzyma się nałożonych mu zasad) i próbuje uzyskać jak najbliżej, ale nie więcej niż, 21 punktów. Na początku każdej rundy gracz i krupier dobierają dwie karty, obie karty gracza są odkryte, natomiast tylko jedna karta krupiera jest odkryta. Gracz może dobierać karty bądź nie, aby uzyskać najbardziej sprzyjającą wartość punktów, kiedy gracz zaprzestanie dobierania kart, krupier odkrywa swoją kartę i według zasad, których musi grać, dobiera karty lub nie brać kart. Żeby wygrać gracz musi osiągnąć większą ilość punktów niż krupier, ale jeśli osiągnie więcej niż 21 punktów, lub ma mniej punktów niż krupier, przegrywa. Gra nie ma limitu kart. Gracz ma możliwość ustawienia zakładu, który po wygranej jest pomnażany o jakiś procent, a po przegranej ten zakład jest w całości mu odbierany. Ilość zebranej stawki jest zapisywany w tabeli wyników.

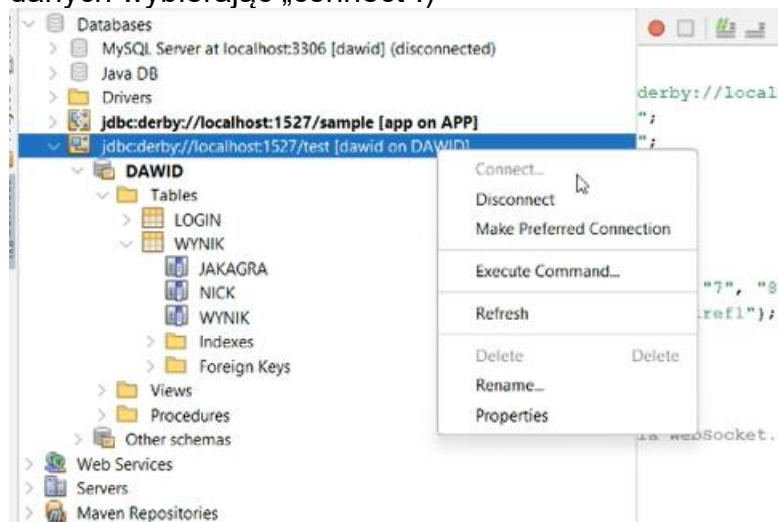
Uruchomienie Oraz Obsługa Projektu:

-Serwer projektu jest uruchamiany w środowisku NetBeans, a klient ma dostęp do programu poprzez adres URL, pod warunkiem że klient ma taki sam adres IP jak serwer. (Polecane jest uruchomienie serwera i klienta z dostępem do tej samej sieci WIFI).

-Obsługa Projektu:

Po wejściu na stronę:

(Należy zmienić adres ip na swój w pliku „plik.jsp”, „wieksze.jsp”, „black.jsp”, aby można było się połączyć ze stroną) (Przy każdym uruchomieniu projektu należy włączyć baze danych wybierając „connect”):



<http://???.???.???8080/projekwebapplication/> powita nas widok ekranu logowania:

Logowanie

Login:

Hasło:

Rejestracja

Login:

Hasło:

Nick:

Gdzie możemy utworzyć nowe konto:

Logowanie

Login:

Hasło:

Rejestracja

Login:

Hasło:

Nick:

Lub zalogować się na już istniejące:

Logowanie

Login:

Hasło:

Rejestracja

Login:

Hasło:

Nick:

Po zalogowaniu, przechodzimy na stronę główną strony, gdzie możemy zagrać w trzy gry.
Zobaczyć tablicę wyników pod przyciskiem „Zobacz topke”
I przycisk „Wyloguj”, który cofa nas do ekranu logowania:

Witaj, Test12!

[Przejdź do Gry Większe mniejsze](#)

[Przejdź do gry kosci](#)

[Przejdź do Strony 3](#)

[Zobacz topke](#)

[Wyloguj](#)

Po kliknięciu przycisku „Zobacz topke” przenosi nas do tabel wyników dla poszczególnych gier:

TOP10 Gry Większe mniejsze

Pozycja	Nick	Wynik
1	dwa	21
2	admin	3
3	Fartuszekxxx	1
4	tak	0

TOP10 Gry BlackJack

TOP10 Gry Kości

[Powrot](#)

Skąd możemy wrócić do strony głównej przyciskiem „Powrót”

Przycisk „Przejdź do Gry Większe mniejsze” przenosi nas do gry karcianej „Mniejsze-Większe”



[Higher](#)

[Lower](#)

Liczba poprawnych zgadnięć: 0








Twój Rekord: 0




Pozostała ilość kart w tali: 51

[Powrot](#)














Przyciski „Higher” oraz „Lower” służą do odgadywania następnej karty, a przycisk „powrót” przenosi nas do strony głównej.

Przycisk „Przejdź do gry kosci” przenosi nas do gry w „kości”:






















Player 1

Kategoria	Punkty	Akcja
1		
2		
3		
4		
5		
6		
3x		
4x		
Full		
Mały Straight		
Duży Straight		
General		
Szansa		
TOTAL		

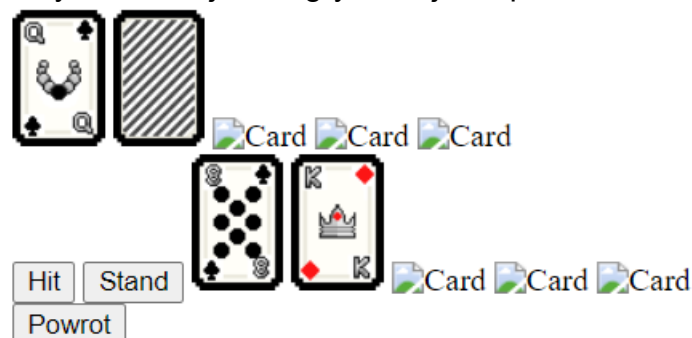
Player 2

Kategoria	Punkty	Akcja
1		
2		
3		
4		
5		
6		
3x		
4x		
Full		
Mały Straight		
Duży Straight		
General		
Szansa		
TOTAL		



Przycisk „powrót” przenosi nas do strony głównej, a przyciski „przypisz gracza 1” i „przypisz gracza 2” pozwalają na przypisanie aktualnego gracza do konkretnej tabeli. Przycisk „start” rozpoczyna grę. Gracze mogą użyć przycisku reroll, aby jeszcze raz losować kośćmi, przyciski „Akcja” służą do wpisywania punktacji do tabeli.

Przycisk „Przejdź do gry blackjack” przenosi nas do gry w „BlackJacka”



Przycisk „powrót” przenosi nas do strony głównej. Przyciski „Hit” i „Stand” są wykorzystywane do gry.

Informacje Na Temat Stworzonych Klas, Metod, Funkcji:

- „Kości”;

Klasa: GameController

@ServerEndpoint("/GameController"): Ta adnotacja określa punkt końcowy WebSocket dla tej klasy.

private static SharedData sharedData = new SharedData();: Instancja klasy SharedData, zawierająca współdzielone dane między różnymi sesjami.

private int[] diceValues;: Tablica do przechowywania wartości pięciu kości.

private final int[][] scores = new int[2][13];: Dwuwymiarowa tablica do przechowywania wyników dla dwóch graczy w 13 różnych kategoriach.

private int player1Total = 0;: Łączny wynik dla gracza 1.

private int player2Total = 0;: Łączny wynik dla gracza 2.

private int currentPlayer = 1;: Zmienna do śledzenia aktualnego gracza (1 lub 2).

private final String[] categories = { ... };: Tablica zawierająca nazwy różnych kategorii punktacji.

private boolean[] selectedDice;: Tablica do śledzenia, które kości są wybrane.

private boolean hasRolled = false;: Wskazuje, czy kości zostały rzucone.

private boolean hasRerolled = false;: Wskazuje, czy kości zostały ponownie rzucone.

Metoda calculateSumOfAllDice: Oblicza sumę wartości wszystkich kości.

Metoda calculateSumOfSpecificValue: Oblicza sumę kości o określonej wartości.

Metoda countDiceValues: Liczy wystąpienia każdej wartości kostki.

Metoda checkConditionsForScore: Sprawdza warunki punktacji w zależności od wybranej kategorii.

Metoda calculateScore: Oblicza punkty dla konkretnej kategorii.

Metoda isThreeOfAKind, isFourOfAKind, isFullHouse, isGeneral, isSmallStraight,

isLargeStraight: Metody do sprawdzania, czy są spełnione określone warunki punktacji.

Metoda hasSpecificValue: Sprawdza, czy istnieje przynajmniej jedna kostka o określonej wartości.

Metoda updateScore: Aktualizuje wynik dla konkretnej kategorii.

Metoda calculateTotalScore: Oblicza łączny wynik dla gracza.

Metoda updatePlayerTotal: Aktualizuje łączny wynik gracza i aktualizuje odpowiadający mu wiersz w tabeli.

Metoda initializeGame: Inicjalizuje grę ustawiając wartości kostek, wysyłając odpowiedzi i ogłaszając turę aktualnego gracza.

Metoda sendResponse: Wysyła odpowiedź do konkretnej sesji.

Metoda rerollDice: Ponownie rzuca kośćmi, które nie zostały wybrane.

Metoda playerRollDice: Rzuca kośćmi dla gracza.

Metoda endTurn: Kończy aktualną turę, przełącza się na następnego gracza i rozpoczyna nową turę.

Metoda startNewTurn: Resetuje flagi i rozpoczyna nową turę, rzucając kośćmi i wysyłając odpowiedzi.

Klasa SharedData:

Metoda public String getgracz1(): Metoda zwracająca nazwę gracza 1 z dodatkowym tekstem "Przypisany1".

Metoda public String getgracz2(): Metoda zwracająca nazwę gracza 2 z dodatkowym tekstem "Przypisany2".

Metoda public void Przypisz1(String msg): Metoda przyjmująca jako argument nazwę gracza i przypisująca ją do pola Gracz1.

Metoda public void Przypisz2(String msg): Metoda przyjmująca jako argument nazwę gracza i przypisująca ją do pola Gracz2, pod warunkiem, że Gracz2 wcześniej był ustawiony na "bb".

Metoda public void wygrana(int nickk): Metoda obsługująca zapisywanie wyników graczy w bazie danych. Przyjmuje jako argument numer gracza (nickk). Sprawdza, czy nick gracza istnieje w tabeli wyników. Jeśli tak, zaktualizuje wynik; jeśli nie, dodaje nowy rekord.

-„Mniejsze-Większe”;

Klasa GraWebSocket:

Jest to klasa reprezentująca endpoint WebSocket dla gry.

Klasa jest oznaczona adnotacją `@ServerEndpoint("/graWebSocket")`, co oznacza, że jest to punkt końcowy serwera WebSocket dostępny pod danym adresem URL `("/graWebSocket")`.

Metoda @OnOpen(): Metoda wywoływana przy nawiązaniu połączenia WebSocket. Inicjalizuje talie kart, uruchamia grę i wysyła komunikaty do klienta.

Metoda @OnClose(): Metoda wywoływana przy zamknięciu połączenia WebSocket. Zapisuje lub aktualizuje wynik gracza w bazie danych.

Metoda @OnMessage(): Metoda wywoływana przy otrzymaniu wiadomości od klienta. W zależności od treści wiadomości, obsługuje zgadywanie, aktualizuje kartę lub ustawia nazwę gracza.

Metoda initializeGame(): Inicjalizuje grę dla danego gracza, ustawia ilość kart w talii i aktualizuje kartę.

Metoda rekord(): Pobiera najlepszy wynik (rekord) gracza z bazy danych.

Metoda handleGuess(): Obsługuje zgadywanie, porównuje wartość aktualnej karty z poprzednią i aktualizuje liczbę poprawnych zgadnięć.

Metoda getCardValue(): Pobiera wartość liczbową karty na podstawie jej nazwy.

Metoda saveOrUpdateResult(): Zapisuje lub aktualizuje wynik gracza w bazie danych w zależności od tego, czy gracz już istnieje.

Metoda createAndShuffleDeck(): Tworzy talie kart, tasuje je i przypisuje do zmiennej "deck".

Metoda updateCard(): Aktualizuje kartę wysyłając informację o kolejnej karcie do klienta.

Metoda sendResponse(): Wysyła wiadomość do klienta poprzez sesję WebSocket.

-„BlackJack”;

Klasa Deck; Reprezentuje talię kart w grze.

Deck(): Konstruktor inicjujący talie kart, tworzący karty dla wszystkich kombinacji kolorów i figur, a następnie tasujący talie.

drawCard(): Metoda zwracająca kartę z wierzchu talii (jeśli talia nie jest pusta).

Klasa Card; Reprezentuje pojedynczą kartę w grze.

Card(): Konstruktor ustawiający kolor i figurę karty.

Metoda toString(): Metoda zwracająca reprezentację tekstową karty.

Metoda getValue(): Metoda zwracająca wartość punktową karty.

Metoda getImagelcon(): Metoda zwracająca ścieżkę do ikony karty.

Metoda getReversedImagelcon(): Metoda zwracająca ścieżkę do ikony odwróconej karty.

Metoda getVisibleImagelcon(): Metoda zwracająca ścieżkę do ikony widocznej karty.

Klasa Hand; Reprezentuje rękę gracza (zarówno gracza jak i krupiera)

Hand(): Konstruktor inicjujący pustą rękę.

Metoda addCard(Card card): Metoda dodająca kartę do ręki.

Metoda getCards(): Metoda zwracająca niemodyfikowalną listę kart w ręce.

Metoda `getScore()`: Metoda obliczająca i zwracająca sumaryczną wartość punktową kart w ręce, uwzględniając elastyczność Asów.

Klasa `blackcontrol`: Główna klasa obsługująca logikę gry.

`blackcontrol()`: Konstruktor inicjujący nową grę.

Metoda `onOpen()`: Metoda wywoływana przy otwarciu sesji, inicjująca grę.

Metoda `onClose()`: Metoda wywoływana przy zamknięciu sesji.

Metoda `endGamenowe()`: Metoda zakończająca grę i rozpoczynająca nową.

Metoda `endGame()`: Metoda sprawdzająca warunki zakończenia gry i ogłaszająca wynik.

Metoda `handleMessage()`: Metoda obsługująca wiadomości od klienta (np. "Hit1", "Hit2", "Stand") i podejmująca odpowiednie akcje w grze.

Metoda `initializeGame()`: Metoda inicjująca nową grę, rozdając karty początkowe.

Metoda `sendResponse()`: Metoda wysyłająca odpowiedź do klienta przez sesję WebSocket.