Technologie Obiektowe 2
Projekt
Gra Mastermind

Zespol1

Dokumentacja

Michał Dygas
Ignacy Grudziński
Grzegorz Niedziela
Wiktor Pawłowski

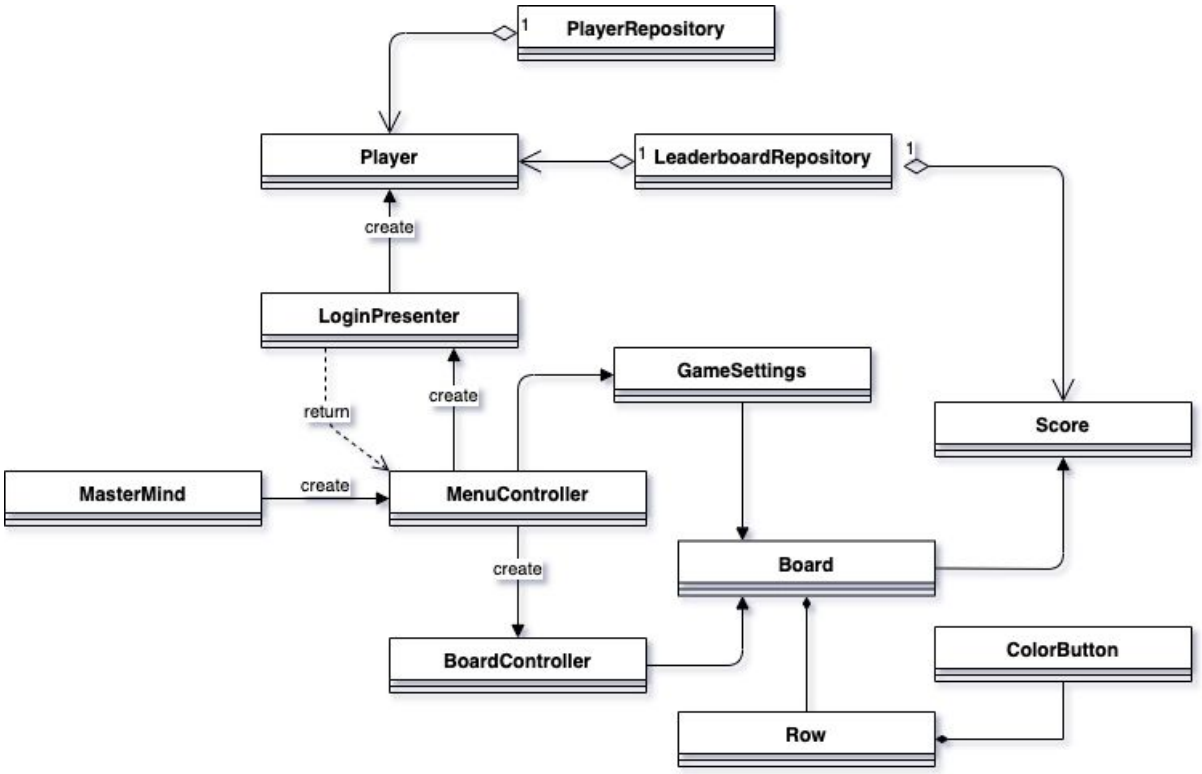
Model - diagram klas (planowanych) **3**

Instrukcja uruchomienia **5**

M1 - początkowy wygląd i funkcjonalności **5**

Używane technologie **6**

Model - diagram klas (planowanych)



MasterMind - klasa odpowiada za inicjalizację gry. Będzie ona tworzyć nową instancję klasy *MenuController*.
MenuController - zadaniem klasy jest prezentacja graczowi menu, z którego będzie mógł rozpocząć grę, zalogować się lub zmienić ustawienia.
BoardController - kontroler planszy.
Board - klasa odpowiadająca za wyświetlanie planszy graczowi.
Row - instancja klasy reprezentuje jeden rząd dziurek na planszy. Jej odpowiedzialnością jest dbanie o poprawne jego wyświetlanie.
ColorButton - zadaniem klasy jest zmienianie koloru przycisku.
GameSettings - odpowiedzialnością klasy będzie danie graczowi możliwości zmiany ustawień rozgrywki.
Score - reprezentacja wyniku rozgrywki. Klasa odpowiada za przekazanie go do *LeaderboardRepository*.

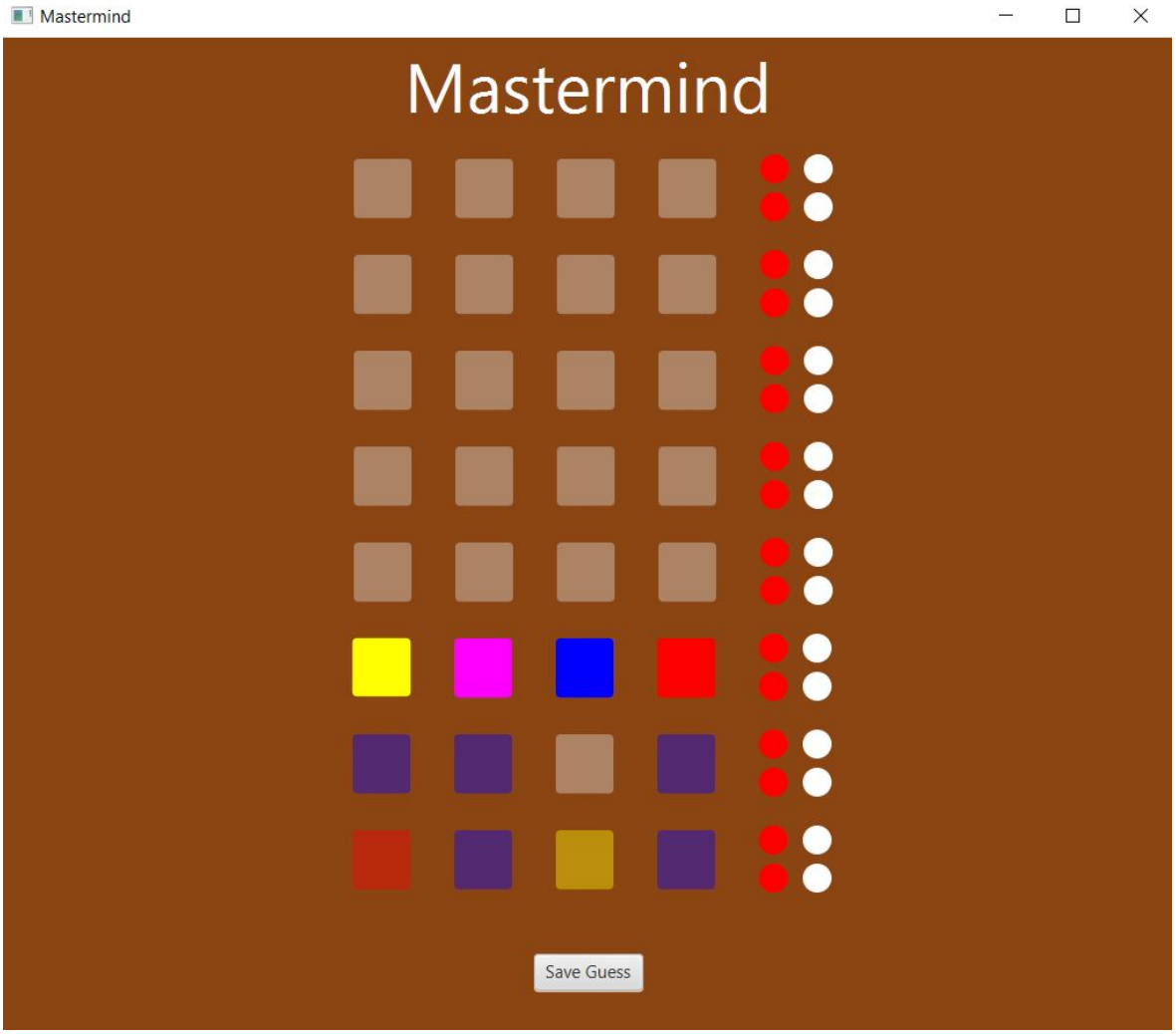
LeaderboardRepository - zadaniem klasy jest przechowywanie wyników rozgrywek graczy.
LoginPresenter - zadaniem klasy jest prezentacja użytkownikowi okna logowania.
Player - klasa reprezentująca gracza.
PlayerRepository - klasa odpowiada za przechowywanie danych graczy.

Instrukcja uruchomienia

1. Otwieramy projekt w IDE (IntellIJ)
2. Klikamy PPM na plik build.gradle i klikamy “Import Gradle Project”
3. Budujemy projekt Gradle -> zespol1 -> Tasks -> build -> build
4. Wybieramy SDK (Java 11)
5. Aby uruchomić aplikację, w menu gradle wybieramy Tasks->application -> Run

M1 - początkowy wygląd i funkcjonalności

Plansza wygląda następująco:



Aby zmienić kolor danego pola, klikamy na niego.

Używane technologie i wykorzystane wzorce projektowe

Do budowania projektu używamy Gradle.
Do Gui będziemy wykorzystywać JavaFX.
Jako baza posłuży nam Sqlite/Mongo.

- MVC - typowa implementacja tego wzorca w oparciu o jeden widok (Board) z generowaną dynamicznie zawartością (Rows), która jest kontrolowana przez klasę BoardController.

- Iterator - w klasie BoardController wykorzystaliśmy ten wzorzec do określania który obecnie rząd powinien być aktywny - takie podejście dodatkowo powinno spełnić zasadę otwarty/zamknięty ponieważ tak napisany kod jest otwarty na rozszerzenie (dodatkowe poziomy trudności) bez większego reformatu, ponieważ z góry zakłada dynamiczną zawartość GUI.

Podział prac

1. M1
   W tym kamieniu milowym prace przypominały bardziej grupową burzę mózgów - każda z osób wtrącała coś od siebie, a projekt nabrał kształtu. Efektywnie wychodziło nam pair programming - jedna osoba pisze, druga robi szybkie review - w ten sposób udało się nam dość sprawnie zaimplementować podstawowe składowe naszego projektu.

   Udziały warte wspomnienia:
- Wiktor - zaprojektowanie sposobu wprowadzania losów na planszę, dodatkowo nadzorował setup projektu (tak aby działał out of the box)

- Grzegorz - zaprezentowanie odrębnego podejścia przy implementacji MVC, refaktoryzacja kodu oraz dokumentacji
- Ignacy - Diagram klas oraz implementacja widoku rzędu w grze
- Michał - zaprojektowanie View w JavaFX, wykorzystanie wzorca Iterator oraz nadzorowanie repozytorium