DOKU\MENTACJA PROGRAMU PUZZLE GAME

Spis treści

[Opis projektu 1](#_Toc43816589)

[Opis klas 1](#_Toc43816590)

[Źródła grafiki 2](#_Toc43816591)

# Opis projektu

Gra powstała jako projekt, wykonana przez Piotra Piesiaka. Puzzle Game polega na ułożeniu puzzli na planszy w kolejności 1 – 15, licząc od lewego górnego rogu. Dozwolone jest tylko przesuwanie elementu na „puste” pole. Nie można „wyjąć” i przełożyć puzzli.

Aby uruchomić grę, należy wypakować wszystkie pliki do jednego folderu skompilować program za pomocą komendy „javac puzzle\_game.java” i następnie uruchomić poleceniem „java puzzle\_game”. Działanie programu opiera się na bazowej klasie Game\_widnow oraz Board.

# Opis klas

Klasa bazowa Game\_window dziedziczy po klasie z pakietu javax.swing JFrame. Można ją znaleźć w pliku Game\_window.java. Zawiera ona pola HEIGHT i WIDTH oznaczające wymiary monitora użytkownika w celu możliwości skalowania okna programu. W konstruktorze tej klasy zostają sczytane parametry rozdzielczości monitora oraz ustawienie tła głównego okna wykorzystując klasę ImagePanel, która dziedziczy i nadpisuje metodę paintComponent z klasy bazowej JPanel skalując w ten sposób tło. Metoda display wyświetla okno programu na środku ekranu, z ustawionym tłem oraz z brakiem możliwości zmiany wielkości.

Klasa Menu dziedziczy po klasie Game\_window i interfejsie ActionListener. Można ją znaleźć w pliku Game\_window.java. Wyświetla ona w oknie przyciski nawigacyjne gry używając siatki GridLayout. W metodzie z interfejsu actionPerformed zostają wykonywane polecenia w zależności od naciśniętego przycisku : „PLAY GAME” – zamyka okno Menu i uruchamia nowe okno gry, „EXIT” wyświetla nowe okno, które pyta użytkownika czy na pewno chce opuścić grę.

Klasa Play\_game dziedziczy po klasie Game\_window i interfejsie ActionListner. Można ją znaleźć w pliku Play\_game.java Posiada ona pola Board przechowujące aktualny stan na planszy, buttons[] – tablicę przechowującą przyciski na planszy oraz zmienne zapisujące przebieg gry. W konstruktorze inicjalizujemy nowy obiekt planszy i zapisujemy początkowe parametry gry . Metodą display\_mod tworzymy planszę składającą się z 16 przycisków, informacje o grze oraz przyciski nawigacyjne. Każdy przycisk jest obiektem klasy Button i posiada swoją nazwę i dwie ikony – żółty na podświetleniu i normalny niebieski. Dodatkowo zapisuje on swoją pozycję (x,y). W metodzie actionPerformed w przypadku naciśnięcia elementu na planszy sprawdzane jest czy można go przesunąć. Jeśli tak to sczytujemy z obiektu Board element na pozycji x,y i zamieniamy go z pustym przyciskiem (stwarzając w ten sposób wrażenie przesunięcia). W przypadku naciśnięcia HINT uruchamiamy algorytm znajdujący najlepszy ruch w danej sytuacji (pod warunkiem, że ruchów jest mniej niż 34 – ograniczenie wpływające na szybkość rozgrywki). W przypadku Menu wracamy do okna Menu.

Klasa Board jest klasą opisującą stan danej planszy. Posiada pola – dwie tablice dwuwymiarowe mówiące o położeniu danego elementu w stanie „początkowym” oraz aktualnym. Pomocnicze pola : neighbours mówi o możliwych ruchach (góra dół lewo prawo) oraz positions przechowujące początkowe pozycje danych elementów (0…15). W konstruktorze inicjalizujemy pola obiektu. Metoda generate\_board generuje losową planszę – metodą random\_move robimy czasami losowe ruchy używając prostej heurystyki mówiącej jak dany ruch spowoduje oddalenie danego elementu od miejsca docelowego. Metoda find\_moves zwraca nam wszystkie możliwe ruchy dla danego stanu – są to sąsiednie elementy pozycji „0”. Metoda make\_move próbuje wykonać ruch danym klockiem i uaktualnia stan planszy. Metoda draw służy do wyświetlenia planszy w terminalu, metoda terminal sprawdza czy plansza jest rozwiązana. Za pomocą metody BFS sprawdzamy najlepszy możliwy ruch do wykonania. Jest to algorytm A\* z heurystyką manhattanską. W kolejce priorytetowej przechowuje stany, które zawierają pola: value – czyli minimalna odległość klocków do ułożenia końcowego, move – czyli ruch, dzięki któremu osiągnęliśmy ten stan, cnt\_moves mówiące ile ruchów zrobiliśmy dotychczas, before czyli Stan poprzedni oraz obiekt klasy board przechowujący aktualne dane na planszy. Korzystam z setu, żeby usprawnić algorytm. Przechowuje w nim informacje o stanie na planszy, który już odwiedził algorytm. Klasa Board\_info zawiera akutalne położenie puzzli oraz nadpisuje metody hashCode i equals, aby móc umieścić ją w secie.

Obiekt klasy End wywoływany jest w razie „wygranej” użytkownika i i informuje o końcu rozgrywki. Również zawiera pola (jak Game\_window) HEIGHT i WIDTH aby umożliwić skalowanie.

Obiekt klasy Exit wywoływany jest w momencie wychodzenia z gry i pyta użytkownika czy na pewno chce wyjść. Tak jak End i Gam\_window zawiera pola przechowujące wymiary monitora w celu skalowania.

# Źródła grafiki

Grafikę tzn. tło i styl przycisków pozyskałem kolejno z Adobe Stock oraz z picpng.com