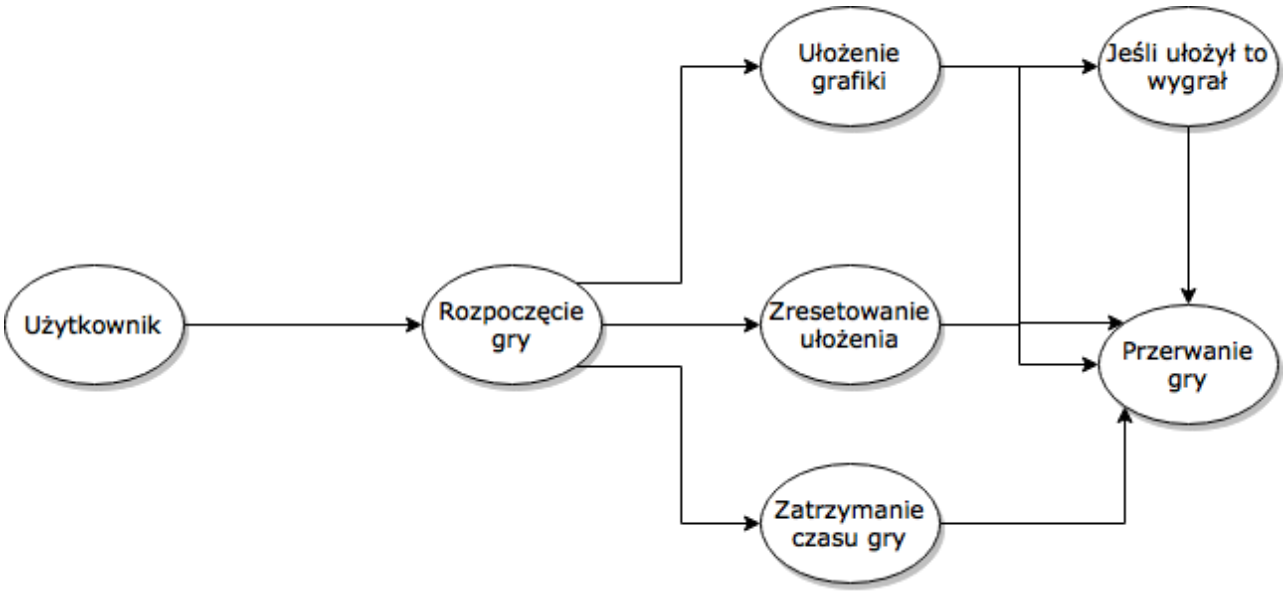
# Puzzle 4x4

## Opis funkcjonalności aplikacji i jej założenia

Aplikacja jest grą polegającą na ułożeniu pojedynczych kafelków tak aby wspólnie ułożyły obrazek w całość w jak najkrótszym czasem. Grafika jest podzielona na 16 części z jedną pustą umożliwiając przesuwanie sąsiednich. Gra kończy się gdy wszystkie kafelki zostaną ułożone poprawne. Każde rozpoczęcie nowej gry powoduje ponownie pomieszanie kolejności kafelek i zresetowanie zegara. Aplikacja została napisana obiektowo w Javie i jest przystosowana do współpracy z JRE 1.9. Można ją zbudować używając MVN i posiadając jego strukturę katalogów(łącznie z dokumentacją).

## Diagram przypadków użycia i jego opis

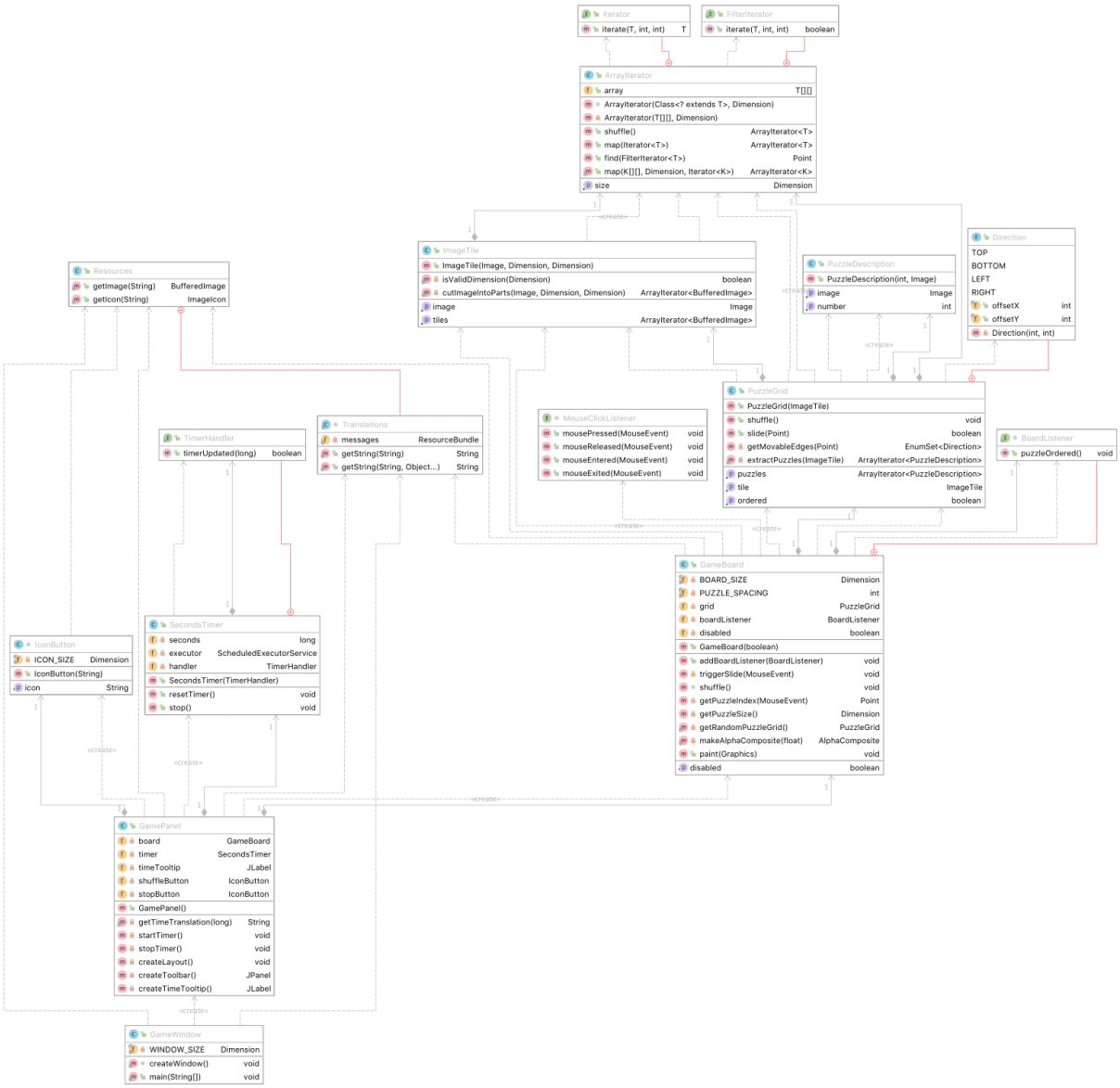



Użytkownik po włączeniu aplikacji rozpoczyna grę przyciskiem z ikonką *Play*. Po rozpoczęciu gry może:

- Ułożyć puzzle, jeśli to zrobił to zostanie wyświetlony mu komunikat o wygranej
- Ułożyć puzzle i przerwać grę zamykając aplikację
- Zatrzymać grę i zegar, który tymczasowo przerywa grę

- Zresetować grę z zegarem

## Diagram klas




Powyższy rysunek przedstawia diagram klas aplikacji. Dziedziczenie klas w tym projekcie zostało zredukowane do minimum i zostało to zastosowane celowo na rzecz kompozycji oraz zasady single responsibility.

## Dokumentacja kodu źródłowego

### Warstwa logiki aplikacji

Głównym założeniem tejże klasy jest operowanie na state aplikacji, niestety w tym przypadku jest on mutable głównie ze względu wielkość wymiaru i specyfikę samego języka.

- Klasa *ArrayIterator*
  Jest to w głównej mierze warstwa abstrakcji między raw array 2D, która dodaje do niej podstawowe operacje, przechowując zarazem wymiary tablicy, które mogą być zmienne podczas runtime programu. Klasa ta tworzy dwuwymiarową tablicę elementów typu `T` wykorzystując mechanizmy refleksji przez wywołanie `Array.newInstance`. Adnotacje `FunctionalInterface` w interfejsach wykorzystywanych przez konstrukcje lambda mają ogromny wpływ maintainability kodu.

  **Metody:**

  - shuffle - miesza elementy tablicy w sposób losowy
  - map - mapuje elementy na wartości zwracane przez funkcję iterującą(najprawdpodobniej lambda)
  - find - wyszukuje element w tablicy i wskazuje wektor dwuwymiarowy pozycji, w której znajduje się obiekt.

- Klasa *ImageTile*
  Klasa odpowiedzialna za cięcie obrazu w tile. Cięcie obrazu następuje w statycznej metodzie `cutImageIntoParts`, która zwraca obiekt `ArrayIterator<BufferedImage>` jako return funkcji. Klasa funkcjonalność `ArrayIterator`

- Klasa *PuzzleDescription*
  Klasa to deskryptor pojedynczego puzzla, posiada obrazek oraz jego numer.

- Klasa *PuzzleGrid*
  Metadata planszy, posiada całą logikę gry i może działać niezależnie od warstwy widoku aplikacji. Posiada enumerator `Direction` reprezentujący wektor, po którym poruszać się może pojedynczy puzzle planszy.

  **Metody:**

    - isOrdered - metoda sprawdzająca ułożenie planszy
    - shuffle - wywołuje metodę shuffle klasy `ArrayIterator` i upewnia się, że zawsze w prawym dolnym rogu jest pusty slide.
    - slide - sprawdza możliwe ruchy slide iterując przez `EnumSet` zwracany w metodzie `getMoveableEdges`
    - getMoveableEdges - zwraca rogi, w kierunku których może być przesunięty slide
    - extractPuzzles - pozwala na wygenerowanie informacji meta z `ImageTile` obrazu

- Klasa *SecondsTimer*
  Timer uruchamiany w thread pool liczącym jedynm wątek i działający w odstępach 1s.

### Warstwa widoku aplikacji

Warstwa ta odpowiada za reprezentacje ruchów użytkownika na ekranie. Renderuje ogólny state aplikacji.

- Klasa *GameBoard*
  Metoda rysująca planszę i dekodująca kliknięcia w planszę na wektor dwuwymiarowy indeksu pojedynczego slide co pozwala na jego przesunięcie. Wczytuje także losowy `ImageTile` z zasobów aplikacji. Pod planszą gry renderuje układany obraz z ustawioną przeźroczystością. Klasa umożliwia grę na planszach innych niż 4x4 ale zostało to zablokowane ze względu na wymagania projektu.

- Klasa *GamePanel / GameWindow*
  Klasy rysujące okienko i zapewniające mu prawidłowy layout.

- Klasa *Resources*
  Menedżer zasobów aplikacji. Pozwala na internationalizację aplikacji, wczytywanie obrazków oraz ikon. Pozwala na formatowanie translacji.

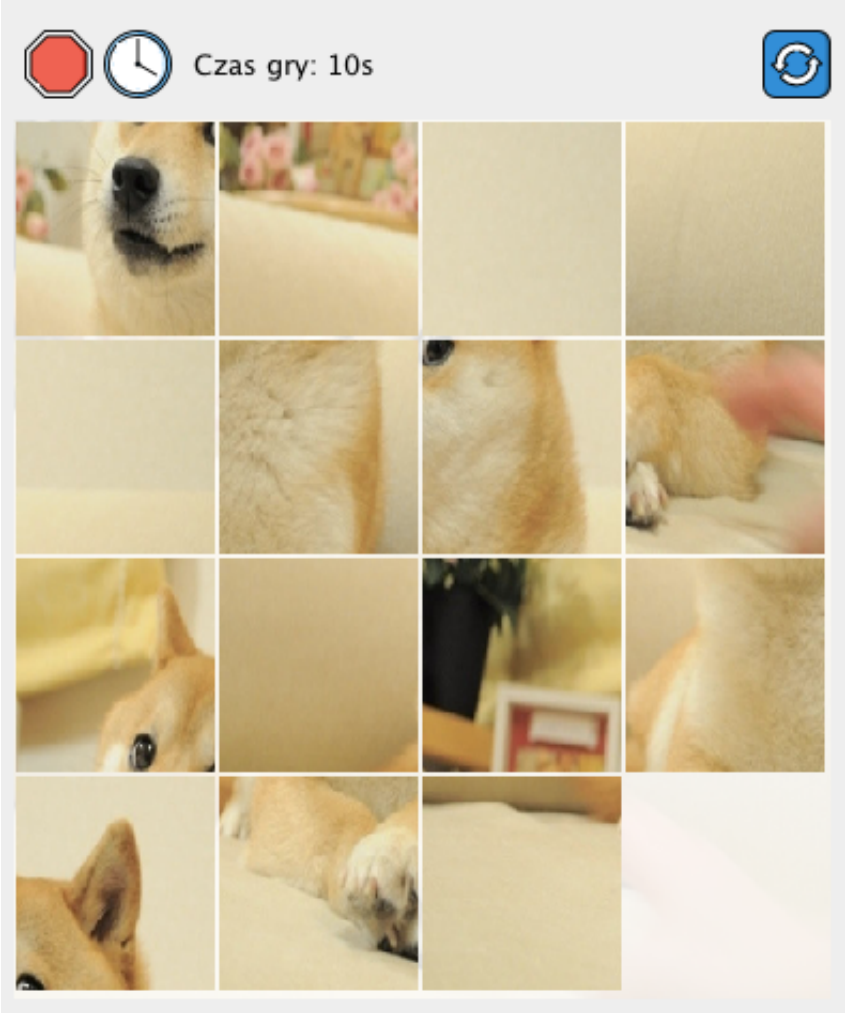
## Podsumowanie

Projekt został zrealizowany zgodnie z pierwotnymi założeniami a podczas jego implementacji nie natrafiono na żadne trudności.

## Zbudowanie paczki

```
mvn package
```

## Zrzut ekranu




## Ikony

Ikony stworzone przez Freepik z www.flaticon.com licencjonowane przez CC 3.0 BY

## Autorzy

Mateusz Bagiński cziken58@gmail.com

Artur Jakiel

## Licencja

GNU