PACMAN

Lidia Dmitruk
Paweł Siemieniuk

1. Instalacja

- Wymagania

  Apache Ant - https://ant.apache.org/

  Java - https://www.java.com

- Build

  Linux/MacOS:

  ./build.sh

- Uruchamianie

  Linux/MacOS:

  java -jar build/pacman.jar

2. Instrukcja

- Cel gry

  Zebranie wszystkich ciastek i uzyskanie jak najwyższego wyniku

- Sterowanie

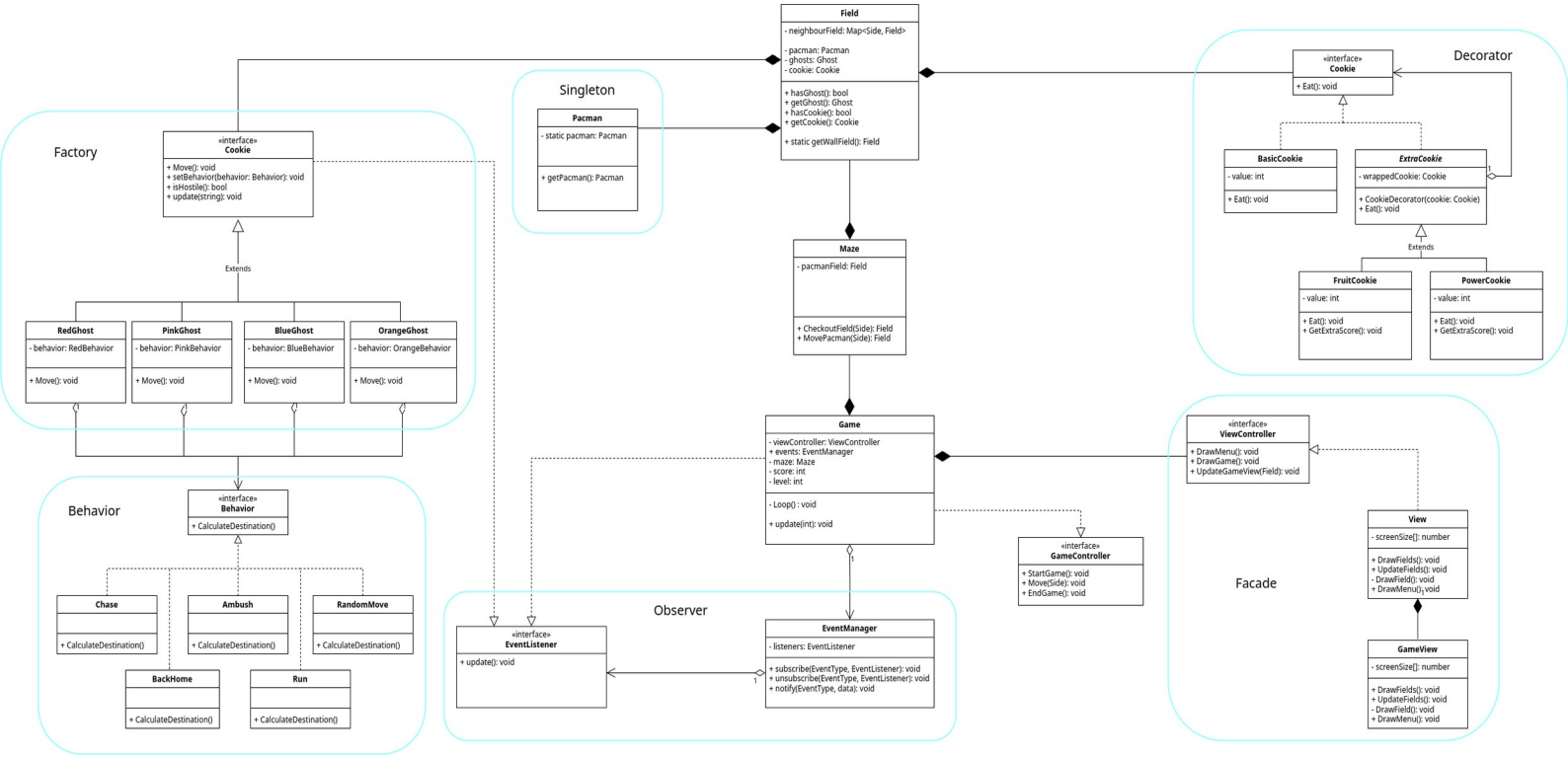
  Poruszanie się po menu – W,S lub Strzałki i Enter

  Zmiana kierunku ruchu Pacmana – A,W,S,D lub Strzałki

  Pauza – Esc

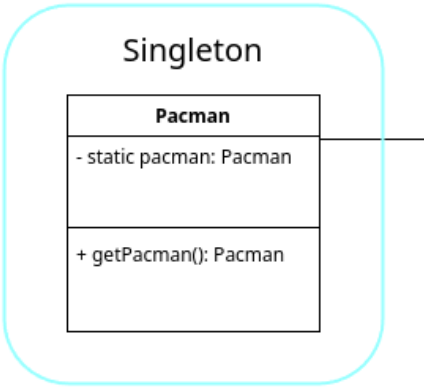
  Wyjście z gry – Q

3. Diagram Klas



4. Wzorce

4.1. Singleton



- Cel wzorca

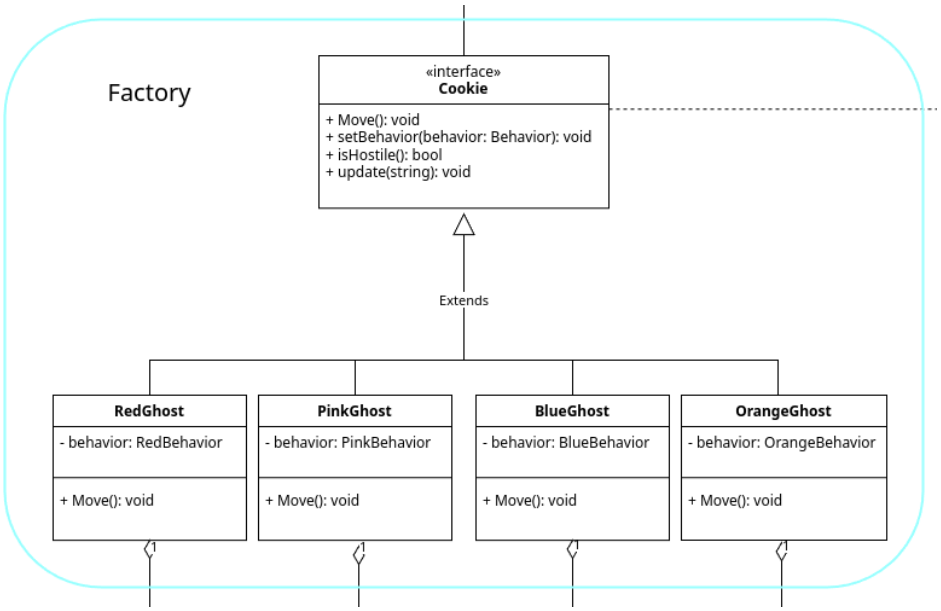
  Ograniczenie do jednej instancji obiektu, ze względu na możliwość kontroli przez gracza, tylko jednego “Pacmana”. Dostępność obiektu, wraz z jego właściwościami, w odrębnych częściach kodu.

- Lokalizacja w projekcie

  Wzorzec zawiera 1 plik:

  - Klasa *Pacman* | ./src/entities/Pacman.java

4.2. Fabryka



- Cel wzorca

  Wzorzec pozwolił na tworzenie tego samego rodzaju przeciwników (duchów), różniących się niektórymi właściwościami i odrębnym sposobem poruszania się po mapie.

- Role klas

  Interfejs *IGhost* – Udostępnia interfejs wspólny dla wszystkich duchów
  Klasy *RedGhost*, PinkGhost, *BlueGhost, OrangeGhost* – Konkretne klasy tworzące różne duchy, udostępniające ten sam interfejs *IGhost*.

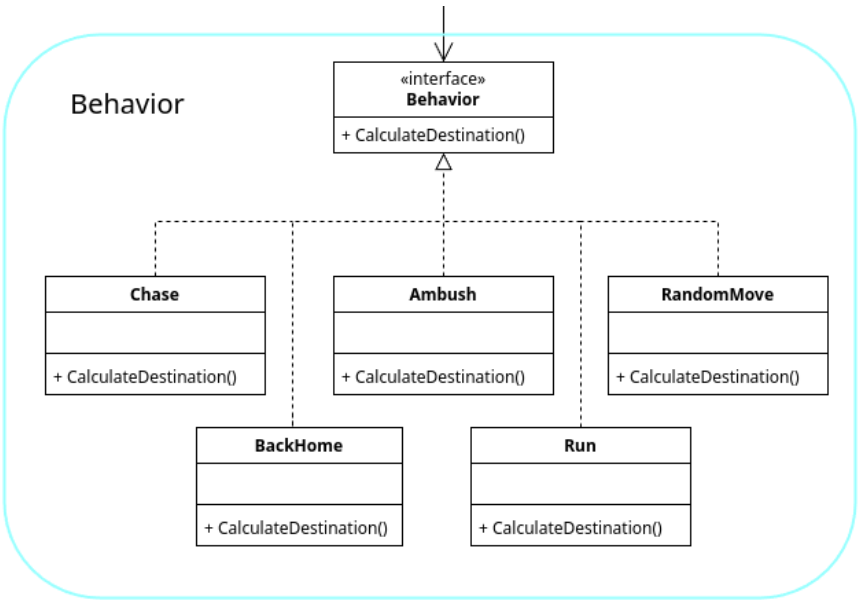
- Lokalizacja w projekcie

  Wzorzec zawiera 5 plików:
  - Interfejs *IGhost* | ./src/entities/ghost/IGhost.java
  - Klasa *RedGhost* | ./src/entities/ghost/RedGhost.java
  - Klasa *PinkGhost* | ./src/entities/ghost/PinkGhost.java
  - Klasa *BlueGhost* | ./src/entities/ghost/BlueGhost.java
  - Klasa *OrangeGhost* | ./src/entities/ghost/OrangeGhost.java

- Wektor zmian

  Wzorzec pozwala na dodawanie kolejnych rodzajów duchów z innym zachowaniem i właściwościami (np. kolorem).

4.3. Strategia



- Cel wzorca

  Wzorzec ma na celu uprościć zmianę zachowań duchów w trakcie rozgrywki.

- Role klas

  Interfejs *IBehavior* – Udostępnia interfejs ze wspólną metodą, obliczającą kolejny ruch ducha, dla każdej klasy z zachowaniem.
  Klasy *Chase, Ambush, RandomMove, BackHome, Run* – Implementują tę metodę.

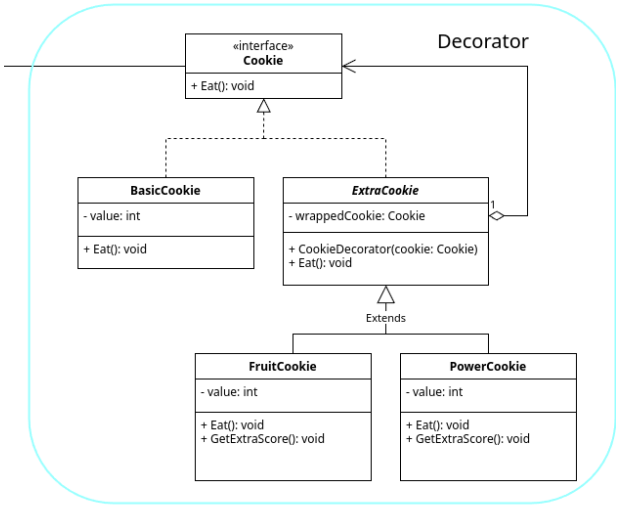
- Lokalizacja w projekcie

  Wzorzec zawiera 6 plików:
  - Interfejs *IBehavior* | ./src/entities/ghost/behavior/IBehavior.java
  - Klasa *Chase* | ./src/entities/ghost/behavior/Chase.java
  - Klasa *Ambush* | ./src/entities/ghost/behavior/Ambush.java
  - Klasa *RandomMove* | ./src/entities/ghost/behavior/RandomMove.java
  - Klasa *BackHome* | ./src/entities/ghost/behavior/BackHome.java
  - Klasa *Run* | ./src/entities/ghost/behavior/Run.java

- Wektor zmian

Wzorzec pozwala na dodawanie kolejnych rodzajów zachowań duchów, implementujących przykładowo inne algorytmy poruszania się po mapie.

4.4. Dekorator



- Cel wzorca

  Wzorzec został wykorzystany do stworzenia dodatkowych rodzajów “ciastek”, mających inne funkcjonalności.

- Role klas

  Interfejs *ICookie* – Udostępnia interfejs podstawowego i ponadpodstawowego ciastka.
  Klasa *BasicCookie* – Implementuje podstawową wersję ciastka.
  Klasa abstrakcyjna *ExtraCookie* – Implementuje dodawanie funkcjonalności do podstawowego ciastka.
  Klasy *FruitCookie, PowerCookie* – Implementują rozszerzenia do funkcjonalności podstawowego ciastka.

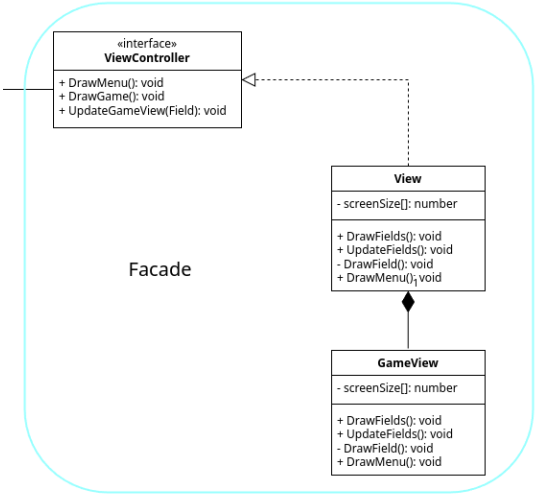
- Lokalizacja w projekcie

  Wzorzec zawiera 5 plików:
  - Interfejs *ICookie* | ./src/cookie/ICookie.java
  - Klasa *BasicCookie* | ./src/cookie/BasicCookie.java
  - Klasa *ExtraCookie* | ./src/cookie/ExtraCookie.java
  - Klasa *FruitCookie* | ./src/cookie/FruitCookie.java
  - Klasa *PowerCookie* | ./src/cookie/PowerCookie.java

- Wektor zmian

  Wzorzec daje możliwość dodawania kolejnych rozszerzeń do podstawowej wersji ciastka.

4.5. Fasada



- Cel Wzorca

  Uproszczenie komunikacji między częścią logiczną gry, a częścią wizualną.

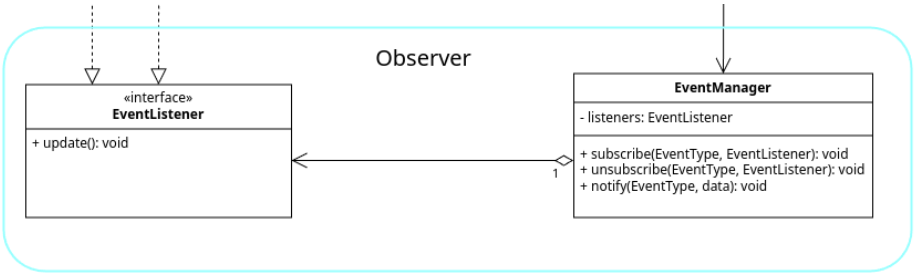
- Role klas

  Interfejs *IViewController* – Udostępnia interfejs do komunikacji z widokiem gry.
  Klasa *View* – Implementacja widoku z uproszczonymi publicznymi metodami.
  Klasa *GameView* – Implementacja widoku rozgrywki.

- Lokalizacja w projekcie

  Wzorzec zawiera 3 pliki:
  - Interfejs *IViewController* | ./src/view/IViewController.java
  - Klasa *View* | ./src/view/View.java
  - Klasa *GameView* | ./src/view/GameView.java

4.6. Obserwator



- ## Cel wzorca

  Informowanie elementów gry o zachodzących zdarzeniach. Przykładowo, zjedzenie przez Pacmana ciastka z mocą, informuje duchy oraz grę o tym wydarzeniu. Gra zaczyna odliczanie do końca trwania efektu, a duchy zmieniają zachowanie.

- ## Role klas

  Interfejs *IEventListener* – Udostępnia interfejs klasom czekającym na zajście konkretnego wydarzenia.
  Klasa *EventManager* – Zarządza zachodzącymi w grze wydarzeniami i wysyła do odpowiednich odbiorców powiadomienia.

- ## Lokalizacja w projekcie

  Wzorzec zawiera 2 pliki:
  - Interfejs *IEventListener* | ./src/game/IEventListener.java
  - Klasa *EventManager* | ./src/game/EventManager.java

- ## Wektor zmian

  Przypisywanie nowo stworzonym klasom (np. nowemu rodzajowi ducha), interfejsu z tymi samymi powiadomieniami. Tworzenie nowych powiadomień.

5. Podział pracy

- Lidia Dmitruk
  - do uzupełnienia
- Paweł Siemieniuk
  - do uzupełnienia

6. Biblioteki

Do stworzenia graficznej części projektu została wykorzystana biblioteka “Lanterna” w wersji 3.1.2.

Źródło: https://repo1.maven.org/maven2/com/googlecode/lanterna/lanterna/3.1.2/