

Etap II

Programowanie obiektowe

March 2021

1 Informacje wstępne

Skład grupy:

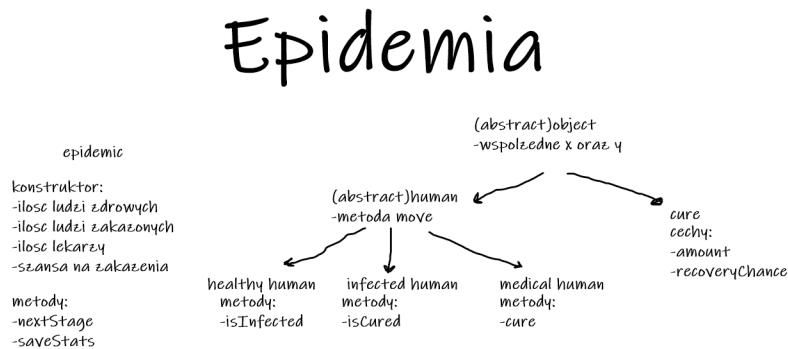
- Damian Gneciak 259065 - przedstawiciel
- Miłosz Siemiński 241339

Język programowania: Java

Środowisko automatycznego budowania: Gradle

2 Propozycja pierwsza - symulacja epidemii

2.1 Wizualizacja



Rysunek 1: Epidemia

2.2 Opis słowny

Główną klasą będzie klasa **Epidemic**, w konstruktorze będzie przyjmować następujące wartości: ilość ludzi zdrowych, ilość ludzi zakażonych, ilość lekarzy, wielkość planszy, skuteczność lekarstwa oraz szanse na zakażenie. Następnie będzie tworzyć podaną ilość obiektów, które będą przechowywane w odpowiednich kontenerach. Klasa będzie posiadała dwie metody: **nextStage** oraz **saveStats**. Metoda **nextStage** będzie sprawiała poruszenie się przechowywanych obiektów oraz za interakcje pomiędzy nimi w przypadku spotkania. Metoda **saveStats** będzie zapisywała do pliku tekstowego numer etapu przebiegu symulacji oraz ilość ludzi zdrowych, zakażonych oraz ilość istniejących lekarstw.

Klasa **Object**, będzie klasą abstrakcyjną, która będzie posiadała dwie współrzędne: położenie *x* oraz położenie *y*. Jej konstruktor będzie przyjmował wielkość planszy oraz ustalał losowe współrzędne nowo powstałego obiektu. Po klasie **Object** będą dziedziczyć dwie klasy: **Human** oraz **Cure**.

Klasa **Cure** w konstruktorze poza rozmiarem planszy będzie przyjmowała szanse na wyleczenie. Dodatkowo będzie przechowywał statyczną zmienną **amount**, która będzie przechowywała ilość lekarstw znajdujących się w symulacji, każdy nowo powstały obiekt **Cure** będzie zwiększał zmienną **amount**.

Klasa **Human** będzie posiadała metodę **move**, która będzie poruszała obiekt o losową ilość pól. Dodatkowo będzie odpowiedzialna za to, aby obiekt nie przekroczył granic mapy.

Klasa **HealtyHuman** będzie posiadała metodę **isInfected**, która będzie pozwalała sprawdzić, czy obiekt klasy **InfectedHuman** znajduje się odpowiednio blisko, aby mógł go zarazić. Gdy takie się stanie, zostanie sprawdzona przechowywana przez obiekt klasy **InfectedHuman** szansa na zakażenie i z odpowiednim prawdopodobieństwem zwrócona wartość **true** lub **false**. Zwrócenie wartości **true** spowoduje że klasa **Epidemic** usunie obiekt klasy **HealtyHuman**, a w jego miejsce utworzy obiekt klasy **InfectedHuman** o takich samych współrzędnych.

Klasa **InfectedHuman** będzie posiadała metodę **isCured**, która będzie zwracała wartość **true** lub **false** w zależności czy obiekt klasy **Cure** będzie dostatecznie blisko, oraz czy leczenie się powiedzie (klasa **Cure** przechowuje również szanse na wyleczenie). Zwrócenie wartości **true** spowoduje usunięcie obiektu klasy **InfectedHuman** oraz stworzenie w jego miejsce obiektu klasy **HealtyHuman**.

Klasa **MedicalHuman** będzie posiadała metodę **cure**, która będzie zwracała **true**, jeżeli leczony obiekt klasy **InfectedHuman** będzie dostatecznie blisko. Dodatkowo klasa **MedicalHuman** będzie posiadała metodę **canCreateCure**, która z prawdopodobieństwem 50% będzie zwracać **true**, co będzie skutkowało stworzeniem przez klasę **Epidemic** nowego obiektu klasy **Cure**.

3 Propozycja druga - symulacja gry planszowej

3.1 Wizualizacja

Gra planszowa



Rysunek 2: Gra planszowa

3.2 Opis słowny

Zasady gry:

W każdej rundzie gracze będą przemieszczać się o określoną liczbę pól, na swojej drodze będą mogli napotkać karty, o różnych działaniach. Wygrywa ten, kto pierwszy przejdzie przez całą planszę.

Konstruktor klasy **Game** będzie przyjmował ilość graczy, ilość kart oraz długość planszy. Następnie tworzył określoną ilość obiektów klasy **Player** oraz losował rozmieszczenie poszczególnych kart. Klasa **Game** będzie posiadać dwie metody: `nextRound`, która będzie przesuwała graczy o losową ilość pól, korzystając z metod klasy **Dice** oraz metodę `saveStats`, która będzie zapisywała do pliku tekstowego położenie poszczególnych graczy, oraz statystyki użycia poszczególnych kart.

Klasa **Player** będzie posiadała aktualne pole gracza, prawo do wykonania ruchu, wielkość jego ostatniego ruchu oraz numer gracza.

Klasa **Dice** będzie posiadała dwie metody, symulujące rzucenie jedną lub dwiema kośćmi do gry.

Klasa **Card** będzie posiadała abstrakcyjną metodę, określającą działanie poszczególniej karty, zmienną określającą ile razy dana karta została użyta oraz numer karty. Klasy dziedziczące po klasie **Card** będą nadpisywały metodę, określającą działanie poszczególniej karty.