Ireneusz Okniński

310228

Dokumentacja projektu „Monopoly”.

1.Cel projektu.

Celem mojego projektu była implementacja gry w Monopoly. Tworząc go wykorzystałem planszę, karty, oraz zasady gry planszowej „Hasbro Monopoly Classic”.

Założenia:

1. Czytelny interfejs.
2. Gra z komputerem.
3. Wybór sposobu rozstrzygnięcia gry.
4. Wierna implementacja zasad.

2. Opis struktury projektu.

W celu zwiększenia przejrzystości tworzonego przeze mnie programu podzieliłem kod źródłowy na katalogi. W katalogu głównym projektu znajduje się plik monopoly.py, który jest plikiem inicjalizującym grę. Katalog „modules” zawiera pliki z kodem źródłowym projektu. Są to pliki:

__init__.py

ai.py

exceptions.py

fields.py

game.py

gameplay.py

interface.py

player.py

oraz pliki z konfiguracją:

chances.py

constants.py

fields.json

Katalog „tests” zawiera pliki:

__init__.py

test_ai.py

test_fields.py

test_game.py

test_player.py

3. Opis klas.

Program wykorzystuje następujące klasy:

AiPlayer – Klasa dziedzicząca po klasie Player. Reprezentuje gracza sterowanego przez komputer. Odpowiada za podejmowanie decyzji o posunięciu gracza na podstawie jego stanu posiadania, stanu gry, oraz elementowi losowemu, dzięki któremu graj nie jest monotonna.

Player – Klasa reprezentująca gracza. Przechowuje informacje o graczu Monopoly.

Field – Klasa bazowa dla wszystkich pól na planszy.

Start – Klasa dziedzicząca po klasie Field. Reprezentuje pole „Start".

Parking – Klasa dziedzicząca po klasie Field. Reprezentuje pole „Parking".

Jail – Klasa dziedzicząca po klasie Field. Reprezentuje pole „Areszt".

Tax – Klasa dziedzicząca po klasie Field. Reprezentuje pola „Podatek".

DrawField – Klasa dziedzicząca po klasie Field. Reprezentuje pola „Szansa" i „Kasa Społeczna".

BuyableField – Klasa dziedzicząca po klasie Field. Jest klasą bazową dla wszystkich pól, które mogą zostać zakupione przez graczy.

Property – Klasa dziedzicząca po klasie BuyableField. Reprezentuje pola nieruchomości, (działek na których można budować budynki)

Station – Klasa dziedzicząca po klasie BuyableField. Reprezentuje pola „Dworzec Kolejowy".

Service – Klasa dziedzicząca po klasie BuyableField. Reprezentuje pola „Obiekt użyteczności publicznej".

Game – Klasa reprezentująca stan gry. Łączy graczy i pola.

Interface – Klasa reprezentująca interfejs gry.

Gameplay – Klasa reprezentująca rozgrywkę. Łączy stan gry i interfejs.

4. Instrukcja.

Do poprawnego uruchomienia gry wymagane jest posiadanie biblioteki termcolor. Aby uruchomić program użytkownik powinien korzystając z interpretera python3 uruchomić plik monopoly.py z poziomu głównego katalogu projektu.

Program wykorzystuje interfejs tekstowy. Aby wybrać jedną z akcji dostępnych do wykonania, użytkownik powinien wpisać odpowiadający jej znak i potwierdzić klawiszem enter. Jeżeli użytkownik wpisze błędny znak nie zostanie on przyjęty przez program. Gracz musi wpisywać do skutku (podanie poprawnego znaku).

Użytkownik może dodać od 2 od 6 graczy, z czego jeden musi być sterowany przez człowieka. Pozostali mogą być sterowani przez komputer lub nie.

Gra może być rozstrzygnięta na dwa sposoby. Pierwszy z nich to bankructwo wszystkich graczy oprócz jednego . Gracz który nie zbankrutował zostaje zwycięzcą. Drugi z nich to rozstrzygnięcie gry po podanej przez użytkownika liczbie rund. Następuje wtedy podliczenie majątków wszystkich graczy i wyłonienie zwycięzcy (zwycięzców).

Użytkownik może też zakończyć grę bez wyłaniania zwycięzcy. W tym celu należy, będąc w głównym menu, wcisnąć klawisz ‘q’.

5. Podsumowanie.

Pierwszym etapem mojej pracy nad projektem było zebranie materiałów i zapoznanie się z zasadami gry. Stworzyłem plik z konfiguracją pól i rozpocząłem pracę nad kodem. Stworzyłem klasy gracza, pól i gry. Początkowym pomysłem było stworzenie interfejsu graficznego, korzystając z biblioteki pygame. Niestety, po początkowych sukcesach (rysowanie planszy, wyświetlanie pionków graczy, ruch pionków) złożoność tego projektu przerosła moje umiejętności.

Postanowiłem wtedy, że przeniosę projekt na interfejs tekstowy. Tutaj napotkałem na koleje problemy. Jak oddzielić wyświetlanie interfejsu od logiki gry? Jak sprawić, żeby interfejs tekstowy był czytelny i intuicyjny?

Pierwszy z nich rozwiązałem tworząc klasę gameplay, która łączyła klasy logiki, interfejsu oraz pozwalała na łatwe zaimplementowanie gracza AI. Dzięki temu mogłem w łatwy sposób sterować wyświetlaniem komunikatów, przechodzić między okienkami, pobierać dane od użytkownika i wykorzystywać je w modyfikacji stanu gry.

Drugi problem rozwiązałem tworząc funkcje wyświetlające okna, komunikaty, pobierające dane od użytkownika. Od tego momentu tworzenie przebiegu rozgrywki było bardzo schematyczne (dodaj funkcjonalność, wyświetl na ekran, pobierz odpowiedź, zareaguj na odpowiedź).

Podczas testowania rozgrywki wpadłem na pomysł, że plansza z pewnością byłaby bardziej czytelna i estetyczna jeśli nieruchomości byłyby wyświetlane w kolorze swojej dzielnicy. Implementacja tego nie była trudna a przyniosła zadowalające efekty.

Podsumowując, gdybym miał tworzyć ten projekt jeszcze raz, rozpocząłbym od dokładnego przemyślenia struktury i komunikacji między klasami, żeby od razu dało się tworzyć wszystko w przystępny sposób. Zdekomponowałbym klasę gameplay na mniejsze klasy, ponieważ w kilku przypadkach kod jest zbyt skomplikowany i dałoby się go napisać w dużo prostszy i bardziej czytelny sposób. Był to mój pierwszy projekt o takiej dużej skali.

Mimo, że stworzona przeze mnie gra nie zastępuje w 100% gry planszowej, jestem z niej zadowolony. Rozegrałem kilkukrotnie całą partię, zarówno z kolegami jak i z komputerem. Najbardziej w moim Monopoly podoba mi się wyświetlanie planszy oraz handel między graczami.