Dokumentacja użytych Wzorców Projektowych

Wzorzec Projektowy: Proxy

Opis

Wzorzec Proxy jest wzorcem strukturalnym, który umożliwia kontrolę dostępu do innych obiektów poprzez utworzenie obiektu pośredniczącego (proxy). Proxy działa jako zastępstwo dla innych obiektów i umożliwia wykonywanie operacji na nich, kontrolując jednocześnie dostęp.

Użycie w Grze Monopoly

W implementacji gry UJPoly wykorzystano wzorzec Proxy do zarządzania dostępem do głównego obiektu gry o nazwie `Gra` oraz kontroli operacji na nim podczas przeprowadzania testów.

W przeprowadzaniu testów proxy klasy `Gra` umożliwia dodawanie i przy tym kolejkowanie następnych zdarzeń w grze, takich jak: wciśnięcie spacji lub innych przycisków, wystąpienie zdarzeń określonego typu, np. pygame.QUIT. Pozwala to na “pisanie” konkretnych scenariuszy według których gra wykonuje się. Imituje to prawdziwego użytkownika oraz zapewnia stałą kolejność określonych operacji.

Interfejs IGra

Interfejs `IGra` definiuje operacje, które mogą być wykonywane w klasie Gra, takie jak aktualizacja zdarzenia, aktualizacja stanu gry i wyświetlanie. Poniżej znajduje się jego implementacja.

```
import pygame
from abc import abstractmethod


class IGra:
@abstractmethod
        def aktualizacja_zdarzenia(self, event: pygame.event.Event):
        pass

@abstractmethod
        def aktualizacja(self):
        pass

 @abstractmethod
        def wyswietl(self):
        pass
```

Klasa ProxyGra:

```python
class GraProxy(IGra):
    def __init__(self, gra: Gra):

        self._kolejka_test_zdarzen: Kolejka = Kolejka()
        self._instancja_gra: Gra = gra
        self._czy_moze_wykonac_nastepne_zdarzenie = False
        print("HELLO from GraProxy")

    def _zablokuj_wykonywanie_zdarzen(self):
        self._czy_moze_wykonac_nastepne_zdarzenie = False

    # override
    def aktualizacja_zdarzenia(self, event: pygame.event.Event):
        if self._czy_moze_wykonac_nastepne_zdarzenie is True:
            nastepne_zdarzenie = self._kolejka_test_zdarzen.pobierz()

        if nastepne_zdarzenie is not None
            self._instancja_gra.wykonaj_zdarzenie(nastepne_zdarzenie)
            self._zablokuj_wykonywanie_zdarzen()

    # override
    def aktualizacja(self):
        pass

    # override
    def wyswietl(self):
        pass

    def wykonaj_wszystkie_zakolejkowane_wydarzenia(self):
        while not self._kolejka_test_zdarzen.czy_pusta():
            nastepne_zdarzenie = self._kolejka_test_zdarzen.pobierz()

       if nastepne_zdarzenie is not None:
            self._instancja_gra.wykonaj_zdarzenie(nastepne_zdarzenie)

    def pozwol_wykonac_zdarzenie_z_kolejki(self):
        self._czy_moze_wykonac_nastepne_zdarzenie = True

    def dodaj_zdarzenie_do_kolejki(self, event: pygame.event.Event):
        self._kolejka_test_zdarzen.dodaj(event)

    def pobierz_instancje_gry(self) -> Gra:
        return self._instancja_gra
```