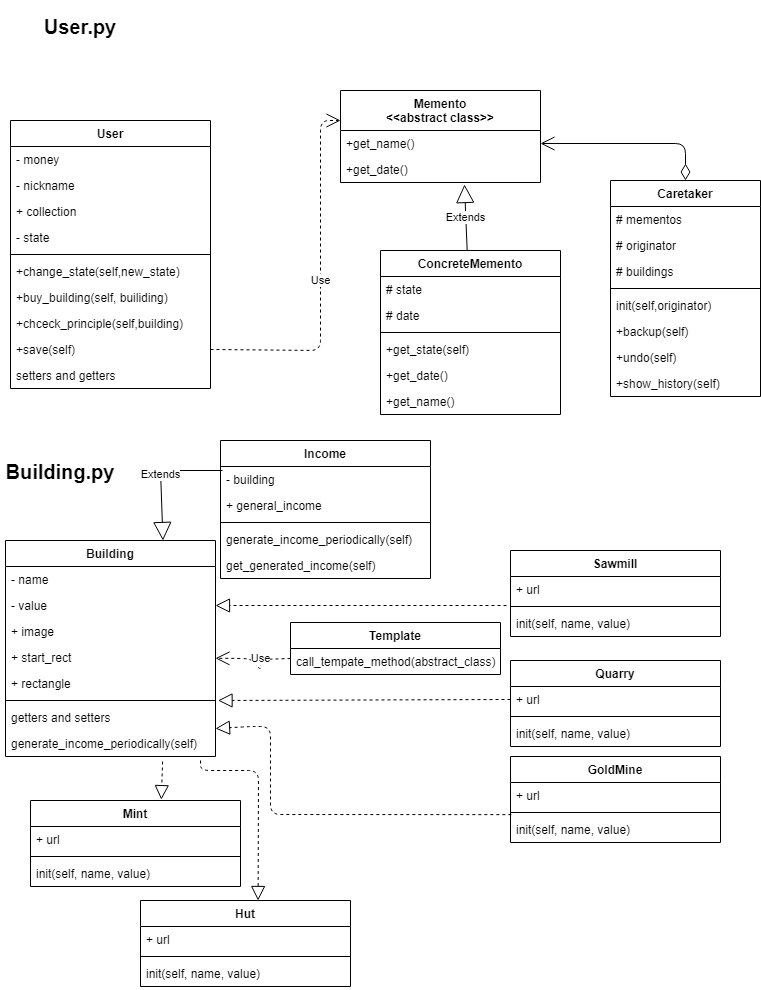
|  |  |  |
| --- | --- | --- |
| Aleksandra Duda | Gr. 31 lab 02 | Techniki Obiektowe |
| Nr indeksu: 130510 | Laboratorium nr 7 | 21.12.20r. |

# Sprawozdanie z labolatorium nr 7

1. **Cel ćwiczenia:**

Rozważenie prostej gry strategicznej, której akcja toczy się w średniowieczu na Górnym Śląsku, umożliwiającą stawianie budynków na mapie

1. **Model UML**



1. **Realizacja ćwiczenia**

Najpierw zdefiniowałam klasę użytkownika **User**, która dawała 2000 „monet” każdemu użytkownikowi na starcie, należało wprowadzić również nick. Każda budowla, którą użytkownik zdecydował się postawić zapisywała się do listy **collection.** Wykorzystałam także wzorzec pamiątki – każda zmiana jaką dokonał użytkonwnik była „zapisywana” np. po zmianie nicku, czy kupienia budowli. W klasie **User**  jest też metoda **save** , która zapisuje aktualy stan w pamiątce. Aby zrelizować podany wzorzec należało zaimplementować **Usera**, który używał klasy abstrakcyjnej **Memento,** który ma dwie abstrakcyjne metody **get\_state** oraz **get\_date** **.**

Następnie po klasie **Memento**  dziedziczy klasa **ConcreteMemento,**  w konstruktorze mamy zmienną odpowiadającą za stan obiektu. Mamy również gettery, które pobierają dostarczone dane do **ConctreteMemento.** Metody takie jak **get\_state,**  czy **get\_name** są używane przez klasę **User** w celu przechywanania swojego stanu.Klasa **Caretaker**  nie zależy od klasy **ConcreteMemento.** Nie ma dostępu do stanu użytkownika, który jest przechowywany w pamiątce.

Klasa **Caretaker**  działa na bazie klasy **Memento.** Ma takie metody jak **backup**  oraz **undo.**  Pierwsza służy do zapisyywania stanu obiektu klasy **User.** Natomiast **undo**  służy do usuniecia wcześniej zapisanego stanu. Aby widzieć wszystkie zmiany, których dokonaliśmy zdefiniowałam metodę **show\_history.**

W klasie **User**  jest także metoda **buy\_building** do sprawdzania, czy można kupić budowlę – sprawdzanie, czy użytkownik ma wystarczająco dużo środków, żeby kupić daną konstrukcję.

Jeśli gracza stać na budowlę to jest zwracana wartość jaką będzie musiał zapłacić, a jeśli nie to jest zwracana wartość 0. Później w grze sprawdzane jest w warunku co zwraca gfunkcja in na podstawie tej informacji jest tworzony obiekt lub nie.

Metoda **check\_principle** odnosi się do sprawdzania, czy mamy wybudowany kamieniołom, chatkę i tartak – jeśli tak to możemy wybudować mennicę albo kopalnię złota. W przeciwnym razie, nawet jeśli mamy środki, nie będzie to możliwe. Ten warunek równiez jest sprawdzany tuż przed stworzeniem obiektu.

****

********

Stworzyłam także plik **Buildings.py.**  W którym zdefinowałam klasę **Building.**  Ma zdefiniowane settery i gettery, ale także odpowiada za ładowanie obrazu danej budowli, która jest przekazywana przez konstruktory konkretnych budowli. Mam klasy każdego budynku jak m.in. Mint, Hut, Quarry itd., które w swoich konstruktorach zmwierają zmienną **url** , która jest ścieżką do pliku – każday budek , ma inny obrazek. Wykorzystałam także tutaj metodę szablonową, która jest wywoływana z kalasy **Template**  **-**  w metodzie **call\_template\_method**  jest referencja do obiektu klasy **Building.**

Mam także klasę **Income** , która ma metodę odpowiedzialną za generowanie przychodu **generate\_income\_periodically** – wykorzystałam tutaj wątki z racji tego że jeśli zbudujemy kilka budowli to one wszystkie zaczynają generować dochody i tym samym sposobem robi nam się kilka procesów naraz.

Oprócz tego mam pliki takie jak **MenuWindow.py**  oraz **GameWindow.py.**  W pierwszym pliku stworzyłam menu do gry na podstawie bilblioteki **pygame-menu.** W **GameWindow.py**  stworzyłam całą wizualizację gry. W tych plikach zdaję sobie sprawę, że kod nie jest optymalny, aczkolwiek wolałam się skupić na wzorcach i na istocie zagadnienia laboratorium. W tych plikach programowałam wygląd aplikacji, działanie przycisków, aktualizację napisów itp.

Cały interfejs gry został napisany w gwarze śląskiej.

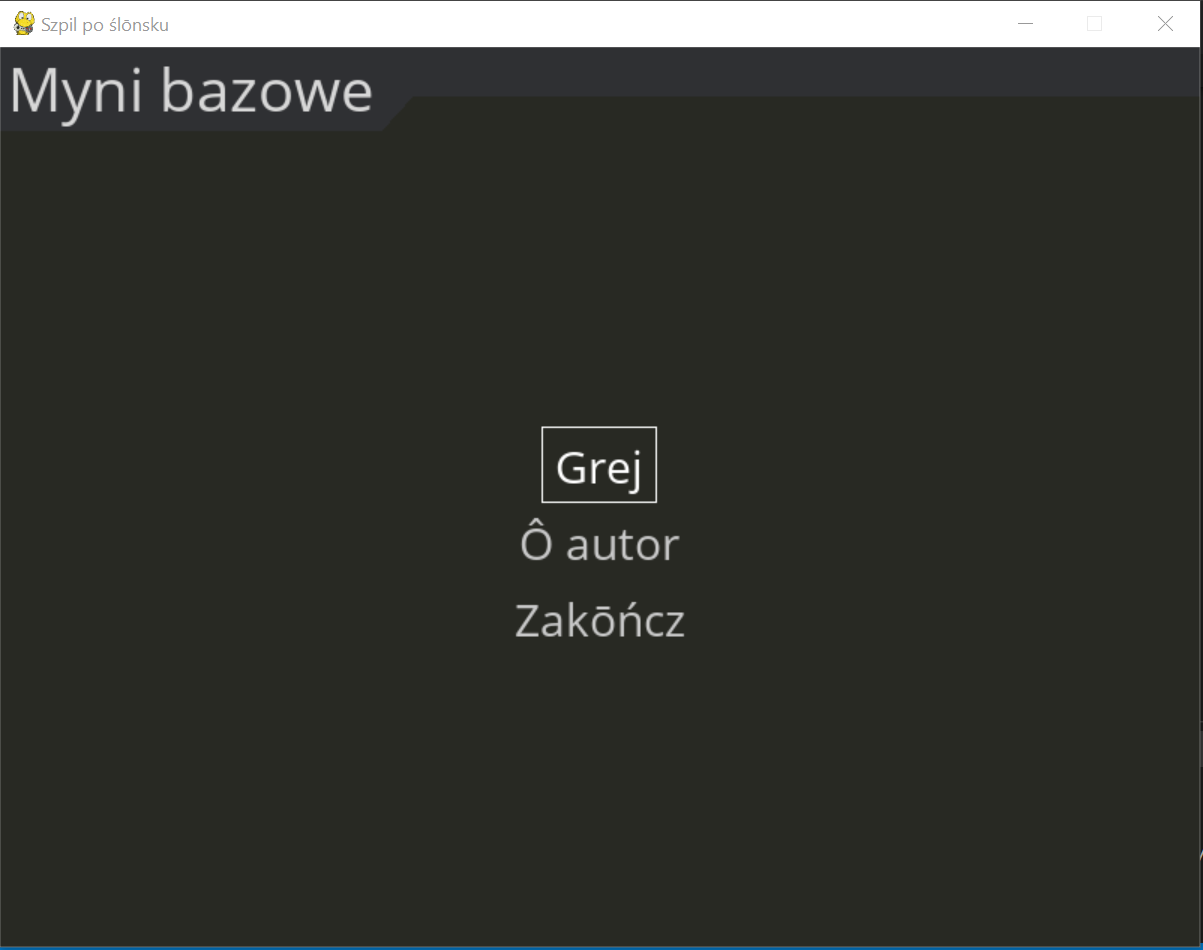


1. **Wnioski i działanie programu**

* Wzorzec Pamiątki może znakomicie posłużyć do późniejszej implementacji funkcji do wczytania gry. Możemy też łątwo zobaczyć jakich zmian i kiedy dokonaliśmy.
* Metoda szablonowa pozwala w szybki sposób zdefniować następujące po sobie akcje bez potrzeby implementacji ich od nowa w każdej klasie.
* W przyapdku generowania przychodów z budowli bardzo przydały mi się wątki. Uzżytkownik z założenia może wybudować wiele budynków, więc każda postawiona budowla po jakimś czasie zaczyna generować dochód, przez co mamy więcej procesów.

**Tutaj krótkie przedstwienie działania gry**

Mamy menu z możliwością zagrania, informacje o autorze gry oraz możliwość wyjścia z gry.



Następnie mamy opcję wyboru nowej gry , wczytania gry (ta funkcjonalność nie została zaimplementowana) oraz moliwwość powrortu.



Jeśli wybraliśmy „nową grę” to wpisujemy nick użytkownika i klikamy „Grej”. Tym spsosbem przechodzimy do gry.



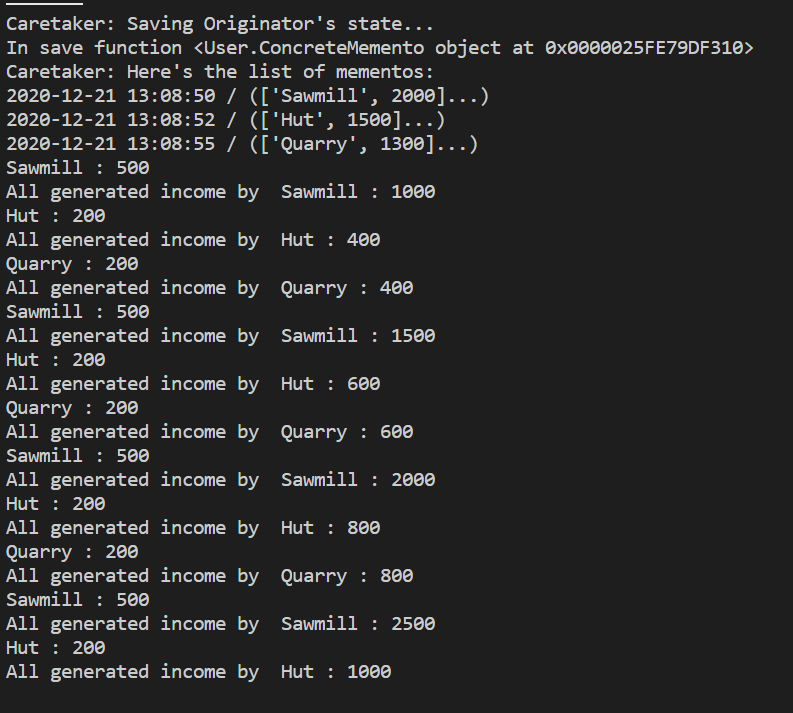
Na pocątku startujemy ze srodkami w wysokości 2000. Możemy kupić jakąś budowlę, po czym ona zaczyna generować dochody.



Kiedy kupujemy budowle oczywiście poczatkowo saldo konta nam spada, bo musimy najpierw kupić, zeby wygenerować przychód.



Następnie wszystkie rzeczy, które robimy są automatycznie zapisywane – każda kupiona budowla zapisuje się w naszej historii. Zaczynają się generować przychody z zakupionych budowli.



Kiedy usiłuję kupić mennicę to pokazuje mi sie komunikat, ze mam za mało pieniędzy, mimo, iż mam zbudowany tartak, kamieniołom i chatkę drwala. Wynika to z faktu, ze oprócz tego, że muszę spełnić wcześniej wspmniany warunek, to muszę jeszcze jeszcze mieć na kupno odpowiednią ilość środkow.



Natomiast wybudowane budowle generują środki, więc za chwilę kiedy będę mieć „więcej na koncie” będę mogła kupić mennicę. Natomiast, gdybym nie miała tartatku, chatki i kamieniołomu to nie mogłabym wybudować mennicy, ani kopalniii złota.

