Politechnika Białostocka

Wydział Informatyki

Temat: Gra komputerowa warcaby z wykorzystaniem wzorców projektowych.

Prowadzący: dr inż. Jerzy Krawczuk

Skład grupy:

- Damian Rosiński
- Adam Polejczuk
- Sławomir Romanowski

Grupa Ps: nr 6
Studia stacjonarne: I stopnia
Kierunek: **Informatyka**
Semestr: 5

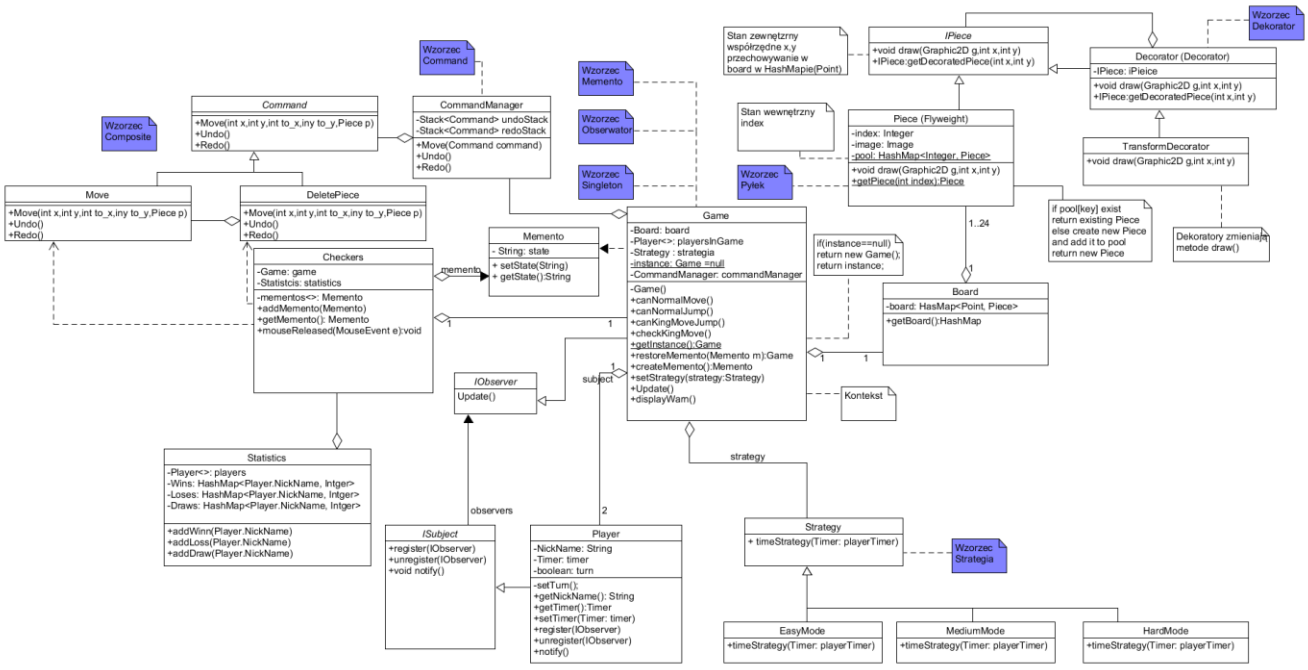
Ocena: ……..

Data: **12-12-2018r**.

1.Opis Projektu

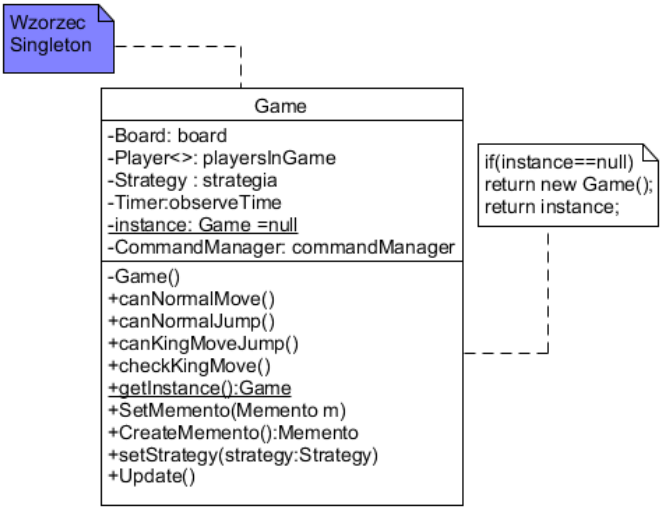
W naszym projekcie zdecydowaliśmy się zaimplementować popularną grę warcaby z wykorzystaniem wzorców projektowych. Rozgrywka będzie umożliwiona dla dwóch graczy wraz z systemem statystyk, który umożliwi sprawdzanie pozycji gracza w rankingu (ranking na podstawie ilości wygranych partii). Zostanie również zaimplementowany mechanizm zapisania i odczytania danej partii warcabów, aby można było daną partię skończyć w innym terminie. Gracze na podstawie kompromisu będą mogli wybrać również trudność rozrywanej partii warcaby(czas dostępny na wykonanie ruchu). Celem całej gry jest zwyciężenie danej partii oraz wspinanie się po szczeblach rankingowych Warcabnicy.

2. Diagram klas – UML



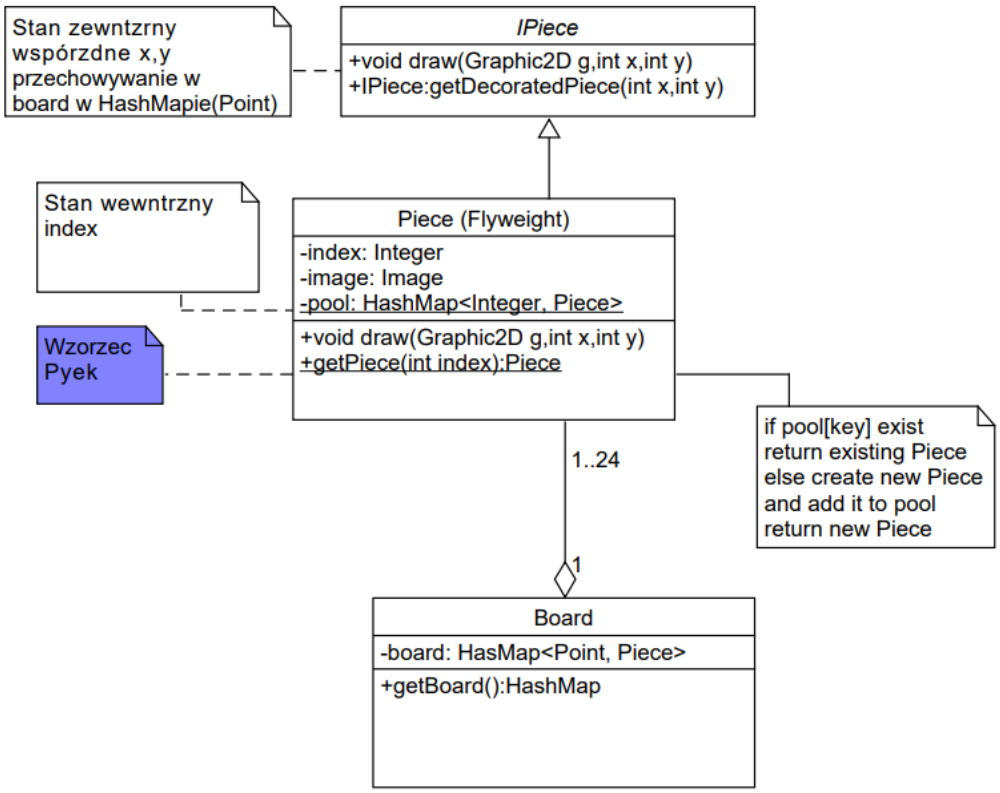
3. Indywidualny opis wykorzystanych wzorców

a) Singleton



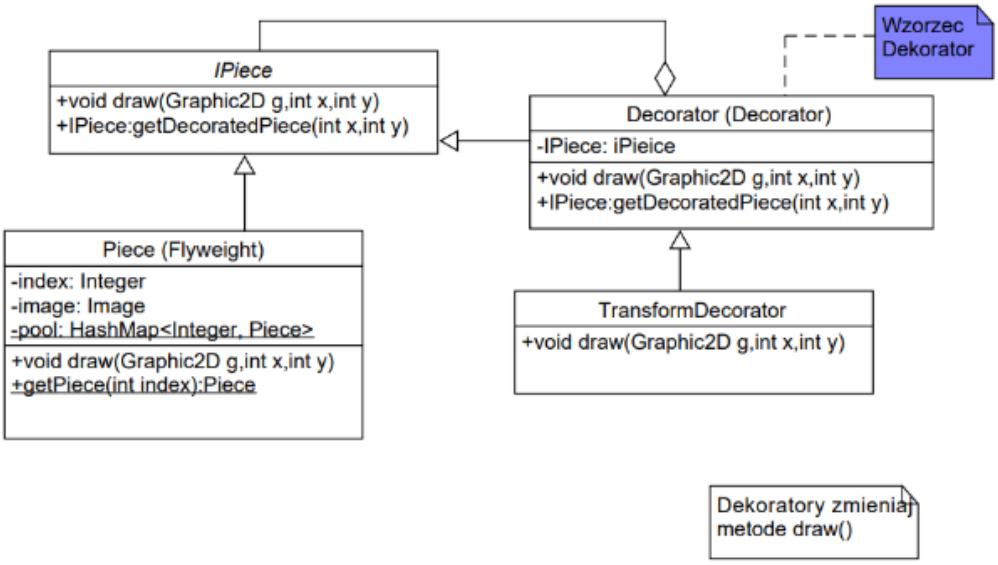
Klasa Game wykorzystuje wzorzec Singleton, aby stworzyć tylko jedną instancję tej klasy. W tym celu zawiera ona prywatny konstruktor oraz statyczną metodę getInstance(). Metoda getInstance() tworzy obiekt tej klasy za pomocą konstruktora tylko przy pierwszym wywołaniu tej metody. Przy kolejnych wywołaniach zwraca ona jedyną, stworzoną wcześniej instancję.

b) Pyłek



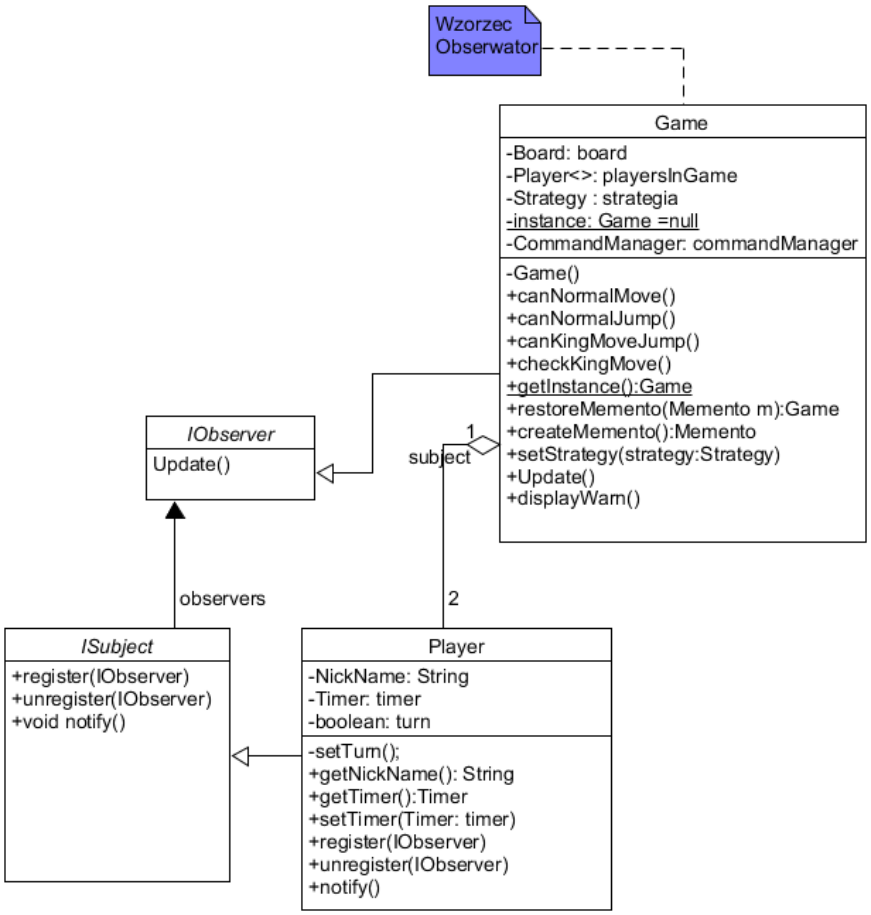
Aby nie tworzyć zbędnych obiektów, prawie identycznych pionków, wykorzystywany jest wzorzec Pyłek. Stan wewnętrzny tego obiektu, będzie przechowywał jego graficzną reprezentację oraz index, który wskazuje który obraz pionka wybrać. Stan zewnętrzny, czyli pozycja pionka na warcabnicy przekazywana jest w metodzie draw(Graphic2D,int x,int y) i przechowywana w klasie Board w hashmapie(w Point).

c) Dekorator



Wzorzec dekorator jest potrzebny do prawidłowego wyświetlania pionków na warcabnicy. W tym celu dekorator zmienia działanie metody draw(Graphic2D,int x,int y), gdyż do wyświetlania pionków potrzebne są współrzędne w pikselach, a pionki do poruszania wykorzystują indeksy pól na szachownicy. Dekorator wykorzystywany jest również w mechanizmie przesuwania figur. Na czas przesuwania obiekt figury opakowany jest w kolejny dekorator. Po zakończeniu przesuwania, obiekt figury jest odpakowywany z dodatkowego dekoratora.

d) Obserwator

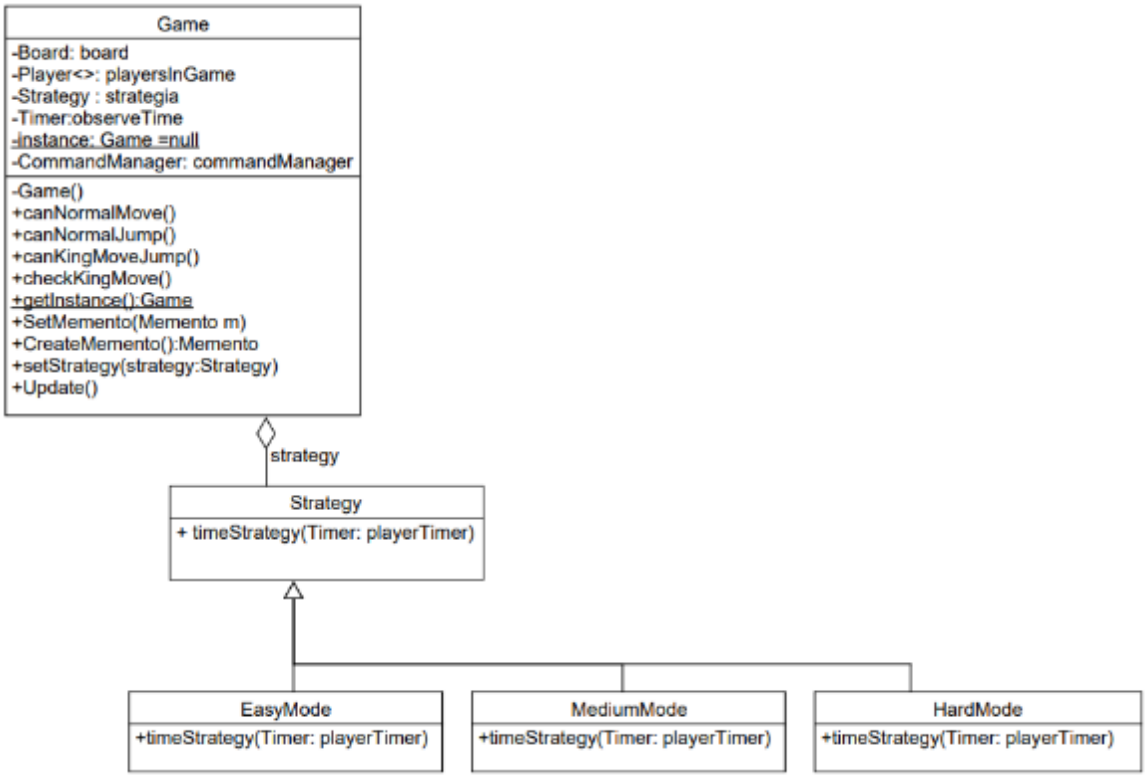


Czynnościowy wzorzec projektowy obserwator, został zaimplementowany w celu informowania gracza w sytuacjach gdy zostało mu mało czasu na podjęcie ruchu.

Game – W naszym projekcie będzie Obserwatorem, który implementuje interfejs IObserver. Posiada referencję do każdego gracza, czyli obiektu obserwowanego, w takim celu, aby w momencie tworzenia obserwatora można było dodać obiekty, które ma obserwować, oraz ustawić obserwatora dla każdego obiektu obserwowanego.

Player – Będzie obiektem obserwowanym, który implementuje interfejs ISubject po to, aby mógł przechowywać swoich obserwatorów i w razie potrzeby nadać do nich sygnał o zmianie stanu czasu.

e) Strategia



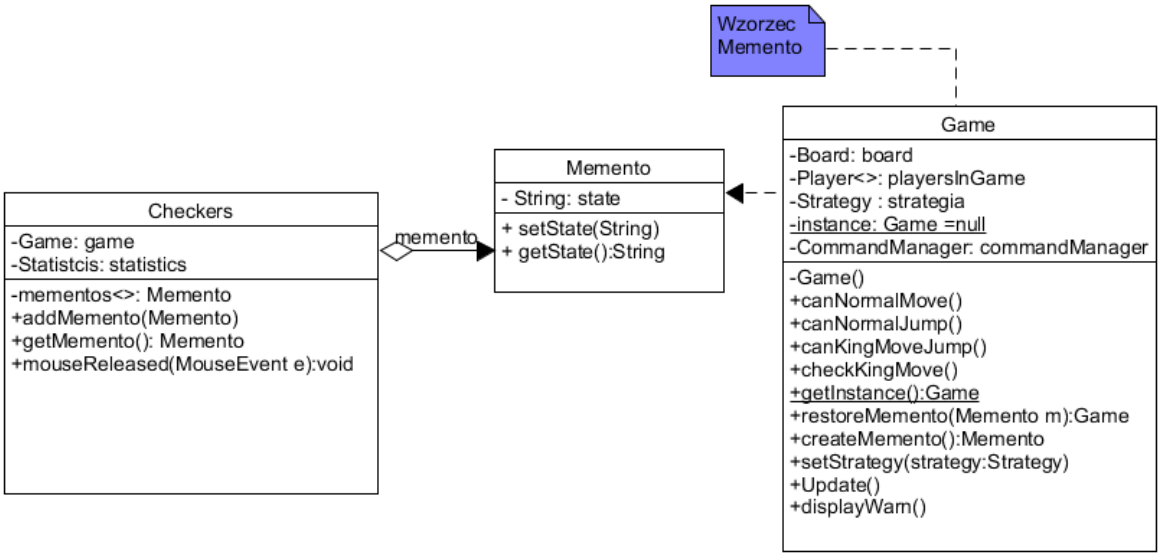
Game –klasa kontekstu, zawiera pole strategia, które wskazuje na obiekt strategii realizującej konkretną wersję algorytmy(w tym przypadku jest to algorytm który przydziela graczowi czas na wykonanie ruchu)

EasyMode - realizuje algorytm, w którym do ustalonego czasu gry, dodowany jest czas pozostały przy podjęciu ruchu,

MediumMode - realizuje algorytm, w którym dodawany jest średni czas pozostały przy wszystkich ruchach.

HardMode – realizuje algorytm, w którym czas na wykonanie ruchu jest zależny od ilości pionków.

f) Memento



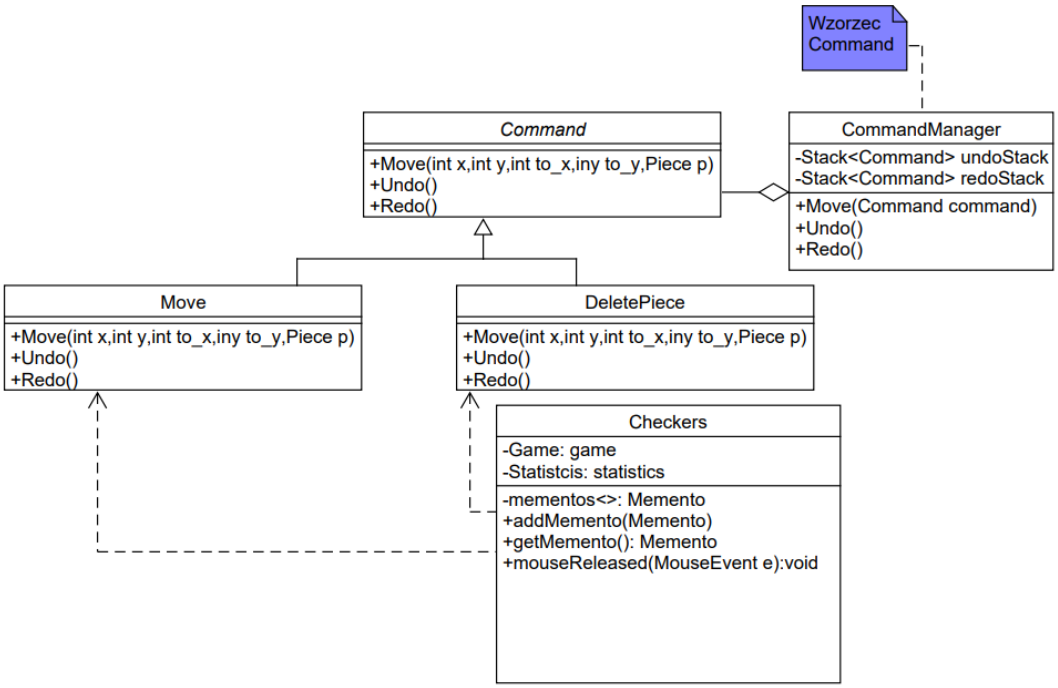
Wzorzec czynnościowy memento, został zaimplementowany w celu przechowywania stanu system, dzięki czemu możliwa będzie implementacja zapisu i odczytu niedokończonych partii.

Game – Obiektem, który będzie przechowywać stan aktualnej rozgrywki. Zarówno będzie mógł tworzyć obiekty Memento, które będą konkretnymi seave'ami w grze, oraz je odczytywać.

Memento – Obiekt, który zawiera swój stan rozgrywki, na podstawie stanu memento będziemy mogli odtworzyć stan obiektu Game. Stan w pamiątce będzie skompresowany do postaci zaimplementowanego wzoru.

Checkers – Pełni rolę zarządcy, który będzie przechowywał stworzone save jako obiekty Memento w strukturze danych, oraz będzie mógł je usuwać, gdy nie będą potrzebne.

g) Command



Wzorzec został zastosowany w celu realizacji mechanizmu undo-redo.

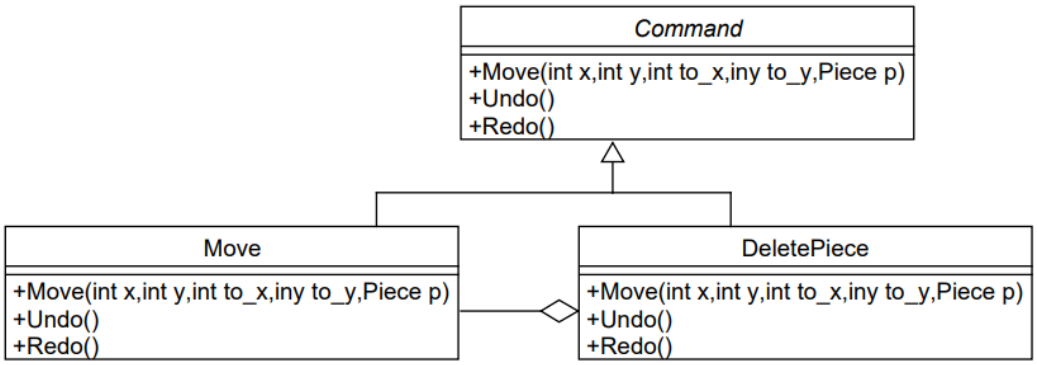
Command – abstrakcyjny typ bazowy z abstrakcyjną metodą Move(),Undo(),Redo()

Checkers – tworzy polecenia (np. poprzez metodę mouseReleased), wywołuje metodę Move() z typu bazowego Command, natomiast po tym wywołaniu wykonuje się metoda Move() nadpisana w klasie Move lub DeletePiece(w zależności czy ruch jest zwykłym ruchem czy biciem)

Move,DeletePiece – konkretne polecenie, dziedziczy z typu bazowego Command

CommandManager – wykonuje konkretne polecenie poprzez wywołanie metody Move(), posiada dwa stosy zapamiętana wykonanych czynności undo-redo

h) Composite



W przypadku cofnięcia bicia figury wymagane jest dwukrotne naciśnięcie przycisku undo (pierwsze do cofnięcia ruchu figury, drugie do cofnięcia zbicia figury). Aby tego uniknąć wykorzystany jest wzorzec Kompozyt. Kiedy dochodzi do bicia figury, tworzone jest polecenie złożone(makro), złożone z 2 poleceń