dr inż. Paweł Głuchowski (Politechnika Wrocławska, Wydział Informatyki i Telekomunikacji)

Inżynieria Oprogramowania

Laboratoria nr 6–8

wersja 1.2

Temat: Wykonanie warstwowego, obiektowego modelu struktury systemu w postaci diagramów klas.

Wstępne informacje

Warstwowy model struktury oprogramowania

Należy wykonać warstwowy, obiektowy model struktury oprogramowania systemu informatycznego i go zaimplementować dla wcześniej wybranych dwóch przypadków użycia (tylko).

Logiczny (nie fizyczny) model struktury powinien być co najmniej 3-warstwowy. Zaleca się architekturę MVP (model - widok - prezenter).

Poszczególne warstwy powinny być współpracującymi ze sobą komponentami przedstawionymi na diagramie komponentów.

Struktura każdego komponentu powinna być przedstawiona na osobnym diagramie klas.

Budowa diagramu klas

Należy wykonać analizę wspólności i zmienności dla scenariuszy wybranych przypadków użycia i zidentyfikować klasy i powiązania miedzy klasami należącymi do modelowanej warstwy oprogramowania:

- klasy bazowe: ogólne, abstrakcyjne i interfejsy;
- ich klasy pochodne: powiązane z nimi relacją uogólnienia lub realizacji (odpowiednio do klasy bazowej);
- strukturalne związki między klasami, czyli relacje: asocjacji, agregacji lub kompozycji;
- opis tych relacji: kierunek dostępu, liczność i pełnione role.

Należy zastosować założenia obiektowego programowania SOLID.

Należy zastosować odpowiednie wzorce projektowe (kreacyjne, strukturalne i czynnościowe) i w sprawozdaniu wyjaśnić ich powód zastosowania i szczegóły działania w ich konkretnym zastosowaniu.

Uwaga: Podczas kolejnego etapu zajęć (diagramy sekwencji w modelowaniu zachowania oprogramowania) wykonane diagramy mogą zostać rozbudowane.

Zadanie 1. (Diagram komponentów)

Polecenie: Należy wykonać koncepcyjny diagram komponentów przedstawiający tworzone oprogramowanie w postaci architektury warstwowej.

Każda warstwa powinna być osobnym komponentem.

Współpracę między warstwami powinny modelować interfejsy (klasy ze stereotypem *«interface»*) w postaci lizaka i łapki, łączące komponenty ze sobą. Nie należy tu używać relacji zależności.

Objaśnienie: Komponenty powinny pozostać puste, czyli bez zawartych w nich klas.

Interfejs zapewniany przez komponent definiuje operacje implementowane przez ten komponent, a których wykonanie może inicjować inny komponent, tego interfejsu wymagający. Dzięki temu każdy komponent jest niezależny od innych, ale w określony sposób z nimi współpracujący.

Interfejsy powinny definiować tylko operacje konieczne do współpracy komponentów ze sobą w celu zapewnienia działania programu i realizacji wybranych w projekcie przypadków użycia. Operacje te będą widoczne w tych interfejsach na odpowiednich diagramach klas.

Zadanie 2. (Diagram klas dla warsty biznesowej (prezenter / kontroler))

Polecenie: Należy wykonać diagram klas dla warstwy biznesowej pełniącej rolę sterującą. Dla architektury 3-warstwowej jest to warstwa *preznetera* (w MVP) lub *kontrolera* (w MVC), opierająca się na klasach sterujących. Dla architektury 5-warstwowej jest to część warstwy biznesowej i warstwa integracji.

Objaśnienie: Warstwa biznesowa (w zakresie sterowania oprogramowaniem) powinna być ograniczona do zapewnienia działania oprogramowania i realizacji wybranych przypadków użycia.

Warstwy klienta i danych nie będą wykorzystane, więc ta warstwa powinna też symulować ich działanie w ramach realizacji tych przypadków użycia.

Klasa sterująca: steruje działaniem oprogramowania i rozdziela warstwy oprogramowania. Odpowiada więc za logikę biznesową (np. realizację przypadków użycia) i przepływ sterowania między klasami tej samej i różnych warstw. W modelu analitycznym posiada stereotyp «control».

Na diagramie należy m.in.:

- umieścić, wypełnić odpowiednimi operacjami i wykorzystać klasy ze stereotypem «interface», które na diagramie komponentów łączą się z modelowaną warstwą;
- zastosować wzorce projektowe:
  - *fasada* – ograniczenie dostępu do tej warstwy, gdy zapewnia usługi dla innej warstwy;
  - *łańcuch zobowiązań* (opcjonalnie) – wieloetapowa realizacja przetwarzania danych przez wyspecjalizowane obiekty;
  - *metoda wytwórcza* lub *fabryka abstrakcyjna* – tworzenie różnorodnych i podobnych obiektów;
  - *obiekt dostępu do danych (DAO)* (opcjonalnie) – dostęp do danych trwałych z warstwy danych (np. z bazy danych);
  - *obserwator* (opcjonalnie) – o zmianach stanu obiektu informowanie obiektów go obserwujących;
  - *strategia* – dynamiczne wybieranie algorytmu np. przetwarzania danych;
  - inne (opcjonalnie).

Zadanie 3. (Diagram klas dla warsty biznesowej (model))

Polecenie: Należy wykonać diagram klas dla warstwy biznesowej pełniącej rolę przechowującą dane. Dla architektury 3-warstwowej jest to warstwa *modelu* (w MVP i MVC), opierająca się na klasach encji. Dla architektury 5-warstwowej jest to część warstwy biznesowej.

Objaśnienie: Warstwa biznesowa (w zakresie modelowania danych) powinna być ograniczona do modelowania danych tymczasowych, przetwarzanych podczas realizacji wybranych przypadków użycia.

Zaleca się wykorzystanie pasywnej warstwy modelu.

Klasa encji: modeluje dane lub struktury danych przetwarzane przez oprogramowanie (np. w celu realizacji przypadków użycia). W przypadku aktywnej warstwy modelu ta klasa może też modelować operacje zmieniające jej stan, czyli stan modelowanych danych. W modelu analitycznym posiada stereotyp «entity».

Aktywna warstwa modelu: zawiera część logiki biznesowej, aby jej obiekty mogły same zmieniać swój stan.

Pasywna warstwa modelu: jej obiekty same nie zmieniają swego stanu; robi to inna warstwa.

Na diagramie należy m.in.:

- umieścić, wypełnić odpowiednimi operacjami i wykorzystać klasy ze stereotypem «interface», które na diagramie komponentów łączą się z modelowaną warstwą;
- zastosować wzorce projektowe:
  - *fasada* (w przypadku aktywnej warstwy modelu) – ograniczenie dostępu do tej warstwy, gdy zapewnia usługi dla innej warstwy;
  - *metoda wytwórcza* lub *fabryka abstrakcyjna* (w przypadku aktywnej warstwy modelu) – tworzenie różnorodnych i podobnych obiektów;
  - *obserwator* (opcjonalnie, w przypadku aktywnej warstwy modelu) – o zmianach stanu obiektu informowanie obiektów go obserwujących;
  - *pyłek* (opcjonalnie) – współdzielenie wspólnej części wielu podobnych obiektów;
  - inne (opcjonalnie).

Zadanie 4. (Diagram klas dla warsty prezentacji (widok))

Polecenie: Należy wykonać diagram klas dla warstwy prezentacji. Dla architektury 3-warstwowej jest to warstwa *widoku* (w MVP i MVC), opierająca się na klasach granicznych. Dla architektury 5-warstwowej jest to warstwa prezentacji.

Objaśnienie: Warstwa prezentacji powinna być ograniczona do wykorzystania terminala w trybie tekstowym do prezentacji wyniku realizacji przypadków użycia i ewentualnie do wprowadzania danych wejściowych; dlatego nie powinna zawierać klas modelujących okna i ich elementy.

Klasa graniczna: zapewnia połączenie między warstwami oprogramowania oraz między oprogramowaniem a aktorem (np. GUI, API). Jest więc interfejsem. W modelu analitycznym posiada stereotyp «boundary».

Na diagramie należy m.in.:

- umieścić, wypełnić odpowiednimi operacjami i wykorzystać klasy ze stereotypem «interface», które na diagramie komponentów łączą się z modelowaną warstwą;
- zastosować wzorce projektowe:
  - *fasada* – ograniczenie dostępu do tej warstwy, gdy zapewnia usługi dla innej warstwy;
  - inne (opcjonalnie).

Zadanie 5. (Implementacja struktury oprogramowania)

Polecenie: Na podstawie wykonanych diagramów klas należy wykonać kod klas: definicje klas, ich atrybutów, operacji itd.

Objaśnienie: Kod należy wygenerować automatycznie przy pomocy programu *Visual Paradigm* i umieścić go w projekcie programistycznym utworzonym w wybranym narzędziu IDE.

Następnie należy ręcznie poprawić i uzupełnić ten kod, aby możliwe było uruchomienie tworzonego systemu informatycznego i wywołanie operacji inicjujących realizację wybranych przypadków użycia.

Operacje te powinny znajdować się w warstwie biznesowej (prezenter lub kontroler) i pozostać puste (nic jeszcze nie robić). Ich kod zostanie utworzony podczas kolejnego etapu zajęć na podstawie diagramów sekwencji.

Opracowano na podstawie instrukcji 5 „Laboratorium z przedmiotu: Inżynieria Oprogramowania W04ITE-SI0011G”, autorstwa dr inż. Zofii Kruczkiewicz.

Ta instrukcja zawiera też przykłady wykonania podobnych zadań i totoriale do programu Visual Paradigm.