## Projektowanie obiektowe oprogramowania Zestaw 7

Wzorce czynnościowe (2)

2022-04-12

Liczba punktów do zdobycia: 8/50 Zestaw ważny do: 2022-05-03

1. (4p) (Event Aggregator) Napisać aplikację okienkową (Windows.Forms, Swing) która jest prostym rejestrem użytkowników.

Okno rejestru składa się z hierarchicznego drzewa użytkowników oraz z panelu roboczego, pokazującego w zależności od wyboru - listę użytkowników (jeśli w drzewie wybrano węzeł kategorii) lub kartotekę użytkownika (jeśli w drzewie wybrano węzeł użytkownika).

Lista użytkowników w panelu roboczym jest widokiem typu *read-only* ale posiada przycisk "Dodaj" przywołujący okno modalne, z widokiem kartoteki w trybie *dodawania*.

Kartoteka użytkownika w panelu roboczym jest widokiem typu read-only, ale posiada przycisk "Zmień" przywołujący okno modalne, z widokiem kartoteki w trybie do edycji.

Za pomocą powiadomień przesyłanych między komponentami (widokami) przez Event Aggregator, zaimplementować następujące schematy komunikacji widoków:

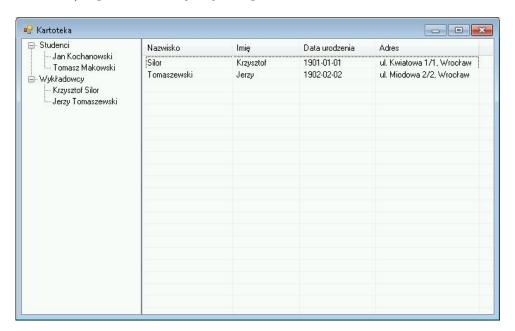
- wybór węzła kategorii na drzewie powoduje załadowanie listy użytkowników w panelu roboczym
- wybór węzła użytkownika na drzewie powoduje załadowanie kartoteki użytkownika w panelu roboczym
- dodanie danych użytkownika na oknie modalnym powoduje odświeżenie listy (pod spodem) w panelu roboczym oraz dodanie węzła do drzewa
- modyfikacja danych użytkownika na oknie modalnym powoduje odświeżenie kartoteki (pod spodem) w panelu roboczym oraz modyfikację opisu węzła drzewa

Formalnie: każde powiadomienie to osobna klasa, ich stan zależy od tego czego dotyczy powiadomienie (na przykład powiadomienie o wyborze węzła kategorii może mieć jako stan informację o wybranej kategorii, powiadomienie o wyborze węzła użytkownika - informację o użytkowniku itd.). Dobrze jest narzucić jakiś porządek nazewniczy (na przykład sufiks Notification: CategorySelectedNotification, UserProfileSelectedNotification itp).

Obiekty formantów interfejsu użytkownika (drzewo, panel listy) publikują/subskrybują odpowiednie powiadomienia - publikacja zwykle następuje po zdarzeniu z interfejsu użytkownika (na przykład kliknięciu w węzeł). Należy zwrócić uwagę na powiązanie rejestracji subskrybenta w Event Aggregatorze (metody AddSubscriber i RemoveSubscriber) z czasem życia tego subskrybenta (kontrolki interfejsu użytkownika): o ile oczywiste jest że rejestracja (AddSubscriber) następuje w chwili powołania kontrolki-subskrybenta do życia) o tyle wyrejestrowanie (RemoveSubscriber) łatwo przegapić i doprowadzić do sytuacji

w której kontrolka jest już niewidoczna dla użytkownika a nadal niepotrzebnie reaguje na powiadomienia).

Za każde z w/w powiadomień jest jeden punkt.

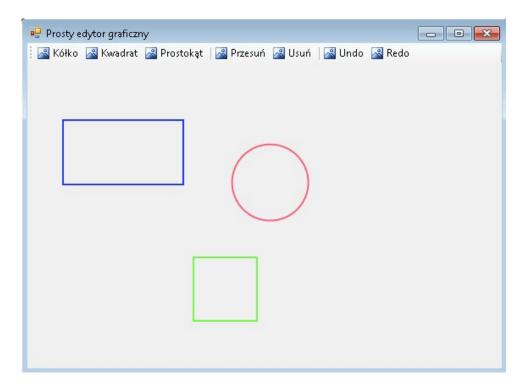


Rysunek 1: Rysunek do zadania 1

2. (**3p**) (**Prototype, Memento**) Napisać aplikację okienkową (Windows.Forms, Swing) symulującą bardzo uproszczony edytor graficzny.

Okno główne aplikacji ma pasek narzędziowy z następującymi "funkcjami":

- Trzy pierwsze funkcje pozwalają wybrać jedną z figur (okrąg, kwadrat, prostokąt). Kiedy aplikacja jest w trybie "figura", to kliknięcie okna roboczego umieszcza na nim w miejscu kliknięcia kolorową figurę (klonowaną z przygotowanego wcześniej prototypu).
- Czwarta funkcja służy do "przesuwania" figur. Po jej wybraniu, kliknięciu w obszar roboczy okna w miejsu w którym znajduje się figura (kliknięcie w obszar figury) i przytrzymaniu myszy, figura daje się przesuwać po obszarze roboczym okna.
- Piąta funkcja służy do usuwania figur. Jej wybranie powoduje przejście w tryb usuwania po kliknięciu w obszar roboczy okna w miejscu w którym znajduje się figura (kliknięcie w obszar figury), figura jest usuwana.
- Dwie kolejne funkcje to Undo i Redo, czyli możliwość nieograniczonego cofania stanu edytora graficznego (cofanie atomowych operacji: dodawanie figury, przesuwanie figury, usuniecie figury) oraz ponawiania cofniętego stanu. Zwrócić uwagę na efektywną implementację Memento (zapamiętywana jest tylko "delta", różnica między stanem poprzednim a następnym, a nie zawsze pełna lista wszystkich figur i ich położeń).
- 3. (1p) (Chain of Responsibility) (za Head-First Design Patterns) Do firmy SA nadchodzi coraz więcej wiadomości e-mail. W trakcie analizy wyróżniono cztery rodzaje wiadomości: (1) pochwalne, (2) skargi, (3) zamówienia (4) pozostałe. Listy pochwalne mają trafiać do



Rysunek 2: Rysunek do zadania 2

prezesa, skargi do działu prawnego, zamówienia do działu handlowego, a pozostałe - do działu marketingu.

Dodatkowo wszystkie wiadomości powinny być archiwizowane.

Przygotować implementację takiego systemu obsługi poczty w oparciu o wzorzec Chain of Responsibility.

Na wejściu pojawia się list (literał), poszczególne handlery łańcucha dokonują klasyfikacji na podstawie treści (jakaś prosta, umowna klasyfikacja).

Uwaga! W odróżnieniu od klasycznej implementacji, tu każda wiadomość jest przetwarzana zawsze przez co najmniej dwa handlery!

Wiktor Zychla