Zaawansowane metody programowania obiektowego

Zadanie 2

Użycie obiektów, klasy wirtualne, metody wirtualne, przetwarzanie drzew

autor: Michał Przewoźniczek

Wrocław, 13.09.2018

Cel zadania

Zadanie polega na oprogramowaniu klasy `CMenu`, która pozwoli na łatwe i szybkie tworzenie tekstowego interfejsu użytkownika. Obiekty klasy `CMenu` mają za zadanie wczytać tekst podany przez użytkownika i w zależności od jego treści wykonać jedną z następujących akcji:

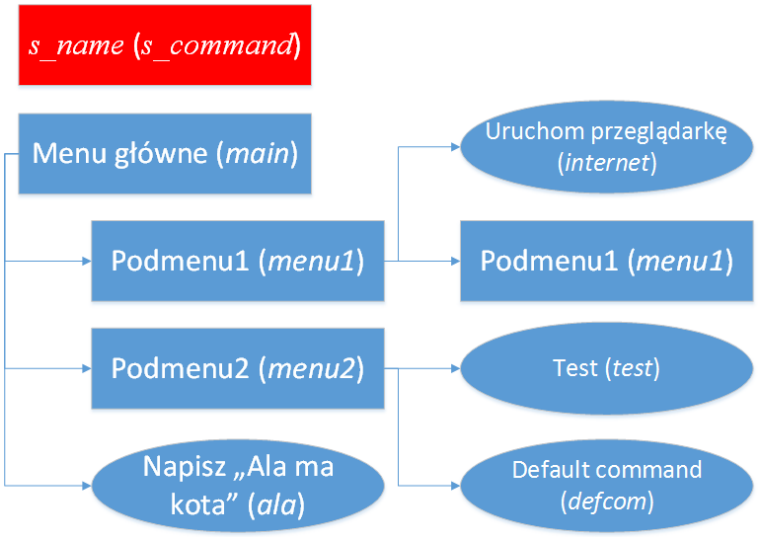
- Poinformować użytkownika, że wprowadził błędne polecenie
- Wykonać zadaną akcję
- Wejść do podmenu

Lista wymogów technicznych do programu:

- Klasa `CMenu` ma zawierać listę pól (w postaci wektora, lub tablicy) klasy `CMenuItem`
- Klasa `CMenuItem` jest klasą czysto wirtualną, po której dziedziczą klasy `CMenu` i `CMenuCommand`
- Każdy obiekt `CMenuItem` ma posiadać
  - Pole `s_command` typu `string`
  - Pole `s_name` typu `string`
  - Metodę `Run`, która uruchamia działanie obiektu. Typ zwracany przez metodę `Run` zależy od koncepcji wykonania programu prze studenta (może być to np. `void`, `int`, lub inny typ)
- Działanie metody `Run` obiektów klasy `CMenu` polega na
  - wyświetleniu swojej nazwy i komendy
  - wyświetleniu nazw wszystkich pozycji (`s_name`) i komend je wywołujących (`s_command`), które są przypisane do tego menu
  - pobraniu od użytkownika tekstu zakończonego naciśnięciem klawisza enter (Uwaga: użytkownicy często piszą głupoty, tak więc należy zabezpieczyć się przed spowodowaniem błędu przez prowadzącego)
  - Jeżeli treść wprowadzanego napisu pasuje do wartości pola `s_command`, któregoś z obiektów `CMenuItem` przypisanego do obiektu klasy `CMenu`, dla którego została wykonana metoda `Run`, to należy wywołać metodę `Run` dla tego obiektu, w przeciwnym przypadku należy poinformować użytkownika o błędzie (patrz: przykład na końcu instrukcji)
  - Jeżeli użytkonik wprowadzi komendę `back`, metoda `Run` kończy działanie

- Działanie metody `Run` dla obiektów klasy `CMenuCommand`, jest następujące. Każdy obiekt posiada wskaźnik na obiekt klasy wirtualnej `CCommand`, która posiada metodę `RunCommand`. Metoda `Run` klasy `CMenuCommand` wywołuje metodę `RunCommand` obiektu klasy `CCommand`, o ile jakiś obiekt klasy `CCommand` został przypisany. Jeżeli nie, to wyprowadza na ekran tekst „pusta komenda". Domyślna treść metody `RunCommand` ma wyprowadzać na ekran tekst „komenda domyślna"
- Należy oprogramować klasę `CMenu` tak, żeby możliwe było dodawanie nowych wpisów do menu, oraz usuwanie i dostęp do istniejących już pozycji. Należy pamiętać że w jednym obiekcie `CMenu` nie mogą wystąpić dwa obiekty o tej samej nazwie i/lub komendzie.
- Należy oprogramować klasę `CMenuCommand` tak, żeby możliwe było definiowanie obiektów `CCommand`
- Przerobić program z listy nr 1 tak, aby w całej komunikacji z użytkownikiem używał klasy `CMenu`

Przykład działania programu



Rys. 1: Przykładowe drzewo menu

Dla drzewa menu przedstawionego powyżej, możliwa jest następująca interakcja z użytkownikiem. Format:
`> <komenda użytkownika>` (<reakcja programu>)

(Program wyprowadza napisy:
```
Menu główne:
1. Podmenu1 (menu1)
2. Podmenu2 (menu2)
3. Napisz "Ala ma kota" (ala))
```

`> Asdjhsakdhaskjdh` (program wyświetla komunikat „nie ma takiej pozycji")

`> ala` (program wyświetla komunikat „Ala ma kota”)

`> menu1` (program wyświetla:
`Podmenu1:`
`1. Podmenu1 (menu1)`
`2. Uruchom przeglądarkę (internet)`

`> internet` (program uruchamia przeglądarkę internetową. Uwaga to jest jedynie przykład, nie trzeba implementować uruchamiania zewnętrznych programów, choć przy takiej strukturze systemu można to zrobić z łatwością)

`> back` (program wraca do menu głównego i powtórnie wyświetla:
`Menu główne:`
`1. Podmenu1 (menu1)`
`2. Podmenu2 (menu2)`
`3. Napisz „Ala ma kota" (ala)`)

`> back` (program kończy działanie)

Pytania i wskazówki

- Zastanów się, która część programu powinna być odpowiedzialna za kasowanie obiektów klasy `CCommand` i obiektów klas dziedziczących po tej klasie