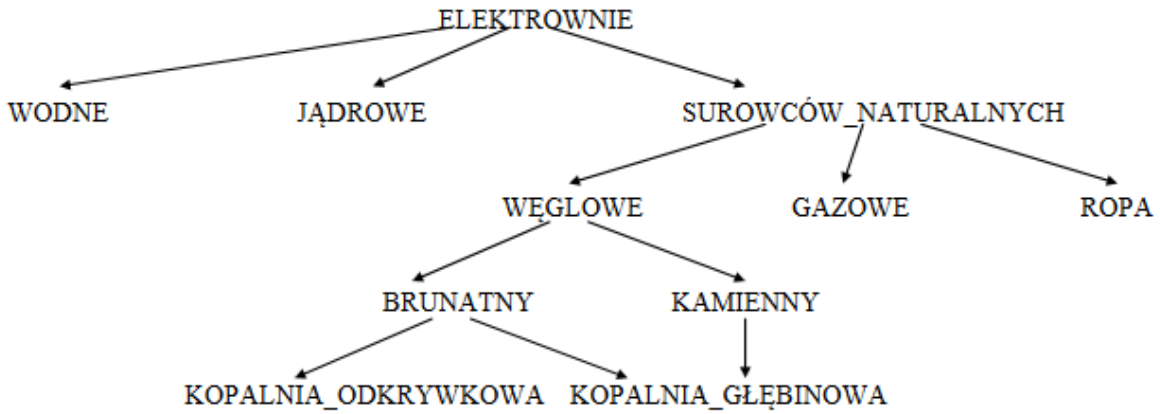
Projekt - Programowanie Obiektowe C++

Adrian Madajewski
145406
Grupa I6.1
Informatyka
semestr 3

Aplikacja realizuje bazę danych obiektów dotyczących Elektrowni według poniższego schematu.



Warstwa użytkownika:

KONSOLA:

Program realizuje następujące funkcje: (komendy można wpisywać wielkimi oraz małymi literami)

- CD [nazwa węzła] – zmiana węzła z dostępnych węzłów:

    - ELEKTROWNIE
    - WODNE
    - JADROWE
    - SUROWCOW_NATURALNYCH
    - GAZOWE
    - ROPA
    - WEGLOWE
    - BRUNATNY
    - KAMIENNY
    - KOPALNIA_ODKRYWKOWA
    - KOPALNIA_GLEBINOWA

Nazwy pól muszą być ciągiem znaków bez spacji.

- MO [nazwa] – tworzy nowy obiekt w bazie o nazwie *nazwa*, o ile już taki nie istnieje. Dostęp do operacji jest możliwy tylko z liści. Wraz z wstępnym utworzeniem obiektu menedżer obiektu pyta o kolejne parametry w zależności od rodzaju tworzonego obiektu.

- DO [nazwa] – usuwa obiekt z bazy (jeśli taki istnieje, w przeciwnym wypadku zwraca błąd). Dostępność tylko z poziomu liści.

- MDO [nazwa] – modyfikacja obiektu o nazwie *nazwa* z obecnego liścia. Zmiana pola odbywa się przez zapytanie, które pole zmienić a następnie o wprowadzenie nowej wartości. (Brak wykrywania błędów wprowadzanych wartości).

- DIR – wypisanie na ekran konsoli nazw obiektów z obecnego węzła.

- SHOW [nazwa] – wyświetla szczegółowe informacje o obiekcie (jeśli istnieje) – wypisuje na ekran konsoli odpowiednio wszystkie jego pola. Komenda dostępna tylko z poziomu liści.

- SAVE – realizuje zapis do pliku. Uruchamiając komendę pojawia się zapytanie o nazwę pliku do zapisu bazy obiektów.

- READ – realizuje odczyt z pliku. Po podaniu komendy należy podać nazwę pliku z którego odczytujemy dane. (Załączono przykładowy plik tekstowy z danymi pod nazwa *Elektrownie.txt*).

- TREE – schemat hierarchii prezentowanych obiektów oraz powiązana miedzy węzłami.

- CLS – wyczyszczenie okna konsoli (realizowane w środowisku operacyjnym za pomocą komendy *system()*).

- HELP – wyświetlenie pomocy (listy dostępnych komend) wraz z krótkim opisem.

- EXIT – wyjście z programu.

Architektura klas programu:

W tej sekcji zamieszczone są informacje na temat pól każdego obiektu, wraz z jego typem w [] nawiasach, oraz domniemaną jednostką w {}.

1. Elektrownie
    - nazwa [String]
    - moc [Double] {MW}
2. Wodne
    - lokalizacja [String]
    - rodzaj zbiornika wodnego [String]
3. Jądrowe
    - lokalizacja [String]
    - pierwiastek [String]
4. Surowców Naturalnych
    - lokalizacja [String]
    - typ surowca [String]
5. Ropa
    - kaloryczność [Double] {kcal/kg}
    - gęstość [Double] {kg/m3}
6. Gazowe
    - rodzaj gazu [String] – np. ziemny.
    - typ gazu [String] – np. *E, LW, LS, LM, LN.*
7. Węglowe
    - rodzaj węgla [String]
    - kaloryczność [Double] {kcal/kg}
8. Brunatny
    - rodzaj kopalni [String]
    - tonaż [Double] {t}

9. Kamienny
    - czystość węgla [Double] {%}
    - koszty wydobycia [Double] {zł}
10. Kopalnia odkrywkowa
    - lokalizacja kopalni [String]
    - liczba górników [Int]
11. Kopalnia głębinowa
    - lokalizacja kopalni [String]
    - liczba górników [Int]

Wszelkie pola dostępnych klas otrzymuje dodatkowo funkcje dostępu oraz funkcje ustawienia wartości rozpoczynając się odpowiednio od przedrostków *get* oraz *set.*

Implementacja klasy Menedzer [T]

Klasa Menedżera to klasa generyczna, która zarówno przechowuje obiekty jak i akcje dostępu, modyfikacji i usuwania obiektów poszczególnego typu. W implementacji klasy menedżera dostępne są następujące funkcje dla każdej z powyższych klas (pod warunkiem dostarczenia odpowiednich funkcji np. *get, set* oraz *przeciążania operatorów* wypisania i wczytywania). Klasa implementuje kilka bazowych funkcji generycznych jak np. znajdywanie obiektów, modyfikacja obiektów, wypisywanie elementów na ekran konsoli, zapis do pliku itp. W celach utworzenia obiektów zastosowano przeciążenie i nadpisanie generycznej metody dodajElement dla każdego z dostępnych w bazie obiektów. Metoda ta wypytuje użytkownika o podanie wszelkich danych do pól obiektu, a następnie dodaje obiekt do odpowiedniej puli obiektów Menedżera. Użytkownik aplikacji nie ma dostępu do samej klasy menedżera jednak dostęp do jej obiektów jest realizowany poprzez akcje z klasy Konsoli – warstwy użytkownika.

Rozszerzalność

Aplikacja została napisana w sposób jak najbardziej rozszerzalny – taka rozszerzalność polega na ewentualnym dodaniu nowego liścia do bazy danych obiektów i wyspecyfikowana w nim funkcji dostępu oraz funkcji wypisywania do pliku, wczytywania z pliku, wypisywania na ekran oraz modyfikacji. Dodatkowo w klasie Konsoli należy określić wtedy nowy Menedżer danych takiego liścia , zdefiniować jego nazwę w typach wyliczeniowych, statycznych mapach oraz zmodyfikować warstwę konsoli o nowe obiekty.