**Języki Programownia, Zadanie 4, 26.10.2011**

Zadanie ma na celu dalsze przećwiczenie użycia wielu klas, a także tworzenia konstruktorów, destruktorów oraz metod i pól statycznych w klasach.

Tworzymy program obsługujący bramki na autostradzie. Gdy dany samochód wjeżdża na autostradę zostaje dodany do listy samochodów, jeśli wyjeżdża, zostaje usunięty z tej listy. Ponadto, niektórzy klienci mają wykupione karnety, a ich dane również widnieją w bazie.

Baza danych zarówno samochodów jak i klientów ma formę tablicy wskaźników:
```
Samochod* baza_samochodow[1000];
Klient* baza_klientow[1000];
```
Przy tworzeniu nowych samochodów korzystamy z operatora ***new***. Podobnie postępujemy z klientami.

Program powinien umożliwiać (w pętli):
- *Dodanie nowego samochodu* (kamery przy bramkach automatycznie odczytują znaki na tablicy rejestracyjnej = dane wpisywane z klawiatury). Jeśli samochód z daną tablicą rejestracyjną widnieje już w bazie danych klientów (klient ma wykupiony karnet), to nowo stworzony samochód ma ustawioną opłatę równą 0. Jeśli dany samochód nie widnieje w bazie, opłata powinna zostać ustawiona na 7.5.
- *Dodanie nowego klienta* (podajemy imię, nazwisko, oraz nr rejestracji samochodu z klawiatury)
- *Wypisanie aktualnie korzystających z autostrady samochodów*
- *Wypisanie bazy danych klientów*
- *Usunięcie samochodu / klienta z bazy (w obu przypadkach wyszukujemy odpowiedniego obiektu bazując na numerach tablicy rejestracyjnej podanej z klawiatury)*

W tym celu należy stworzyć klasy:
`Samochod`
Pola składowe (prywatne!):
- `rejestracja` (std::string)
- `ilosc` (int, **pole statyczne** - podliczające sumaryczną liczbę samochodów na autostradzie)
- `oplata` (double)

Konstruktory:
- `Samochod`(string) - tworzący nowy obiekt samochod na podstawie podanego numeru tablicy rejestracyjnej. Domyślna opłata – 7.5 zł, pole `ilosc` powinno zostać zwiększone o 1
- `Samochod`(string, double) - tworzący nowy obiekt samochod na podstawie podanego numeru tablicy rejestracyjnej i opłaty, pole `ilosc` powinno zostać zwiększone o 1

Destruktor - pole `ilosc` powinno zostac zmniejszone o 1
Metody:
- `int IloscSamochodow()` - metodę statyczną zwracającą wartość statycznego pola `ilosc`.
- `std::string GetRejestracja();` - zwraca wartość pola "`rejestracja`" dla danego samochodu
- `void WypiszSamochod();` - wypisuje rejestrację oraz opłatę dla danego samochodu na ekran

*Wszystkie pola składowe (obu klas) powinny być prywatne, natomiast metody (funkcje składowe) powinny być publiczne.*

Klient
Pola składowe (prywatne!):
- `Imię` (std::string)
- `Nazwisko` (std::string)
- `Rejestracja` (std::string)
- `ilosc` (int, **pole statyczne)** - podliczające sumaryczną liczbę klientów w systemie

Konstrukor
- `Klient(string,string,string)` – imię, nazwisko, oraz nr tablicy samochodu klienta. Pole `ilosc` powinno zostać zwiększone o 1.

Destruktor: pole `ilosc` zmniejszone o 1.

- `bool SprawdzRejestracje(std::string r)` – porównuje otrzymaną rejestrację r z rejestracją zapisaną dla danego klienta, zwraca odpowiednio `true` lub `false`;
- `static int IloscKlientow()` - metoda statyczna, zwracająca wartość statycznego pola `ilosc`.
- `void WypiszKlienta()` - wypisuje na ekran imię, nazwisko oraz rejestrację samochodu klienta.

Wartość pola statycznego `ilosc` powinna na początku wynosić 0, wywołanie konstruktora ma nam je zwiększać za każdym razem o 1, destruktora zaś zmniejszać o 1. Nie tworzymy dodatkowych zmiennych kontrolujących ilość samochodów/klientów, korzystamy z wartości odpowiednich pól statycznych.

**Program powinien być podzielony na 5 plików: klient.cpp i .h, samochod.cpp i .h oraz main.cpp.**

Należy kolejno:

1. Stworzyć klasy Samochod (.h i .cpp), Klient (.h i .cpp) oraz plik z programem głównym (funkcją main). Klasy powinny mieć :
    - odpowiednie składniki i konstruktory / destruktor (program powinien się kompilować)
        - Należy stworzyć obiekt klasy Samochod oraz obiekt klasy Klient w main() **1.5 p**
    - odpowiednie, działające funkcje składowe
        - Należy Zademonstrować działanie wszystkich funkcji w main() **1 p**
2. Zadeklarować statyczne tablice wskaźników: `baza_samochodow` oraz `baza_klientow`. Program powinien w pętli umożliwiać:
    - dodawanie nowych klientów (wskaźniki!) do bazy (w odpowiednie miejsce tablicy, opierając się na wartości pól statycznych) - użycie operatora `new` **0.5 p**
    - wypisanie bazy danych klientów **0.5 p**
    - dodanie nowego samochodu do bazy danych – program pyta o numer rejestracji (Uwaga! Jeśli dana rejestracja istnieje już w bazie danych klientów należy ustawić opłatę na 0 zł! W przeciwnym wypadku koszt przejazdu wynosi 7.5 zł.) **0.5 p**
    - wypisanie bazy danych samochodów **0.5 p**
    - usunięcie samochodu z listy (bazując na numerze rejestracji) **0.5 p**

    Dodatkowo: Przedyskutować z prowadzącym sens tworzenia tablicy wskaźników zamiast tablicy obiektów. Uprzednio należy wypisać na ekran wielkość na dysku wskaźnika na obiekt Samochod `sizeof(Samochod*)` oraz wielkość samego obiektu `sizeof(Samochod)`.

```
g++ klient.cpp samochod.cpp main.cpp –o autostrada
```