giganci programowania

Materiały powtórzeniowe

Wstęp do OOP

Witaj w materiałach powtórzeniowych przygotowanych specjalnie dla Ciebie. Znajdziesz tutaj powtórkę materiału z ostatniej lekcji oraz zadania do wykonania dla chętnych. Z pewnością sobie poradzisz. Powodzenia!

Cel lekcji

Celem lekcji jest zapoznanie uczestników z podstawowymi zagadnieniami, koncepcją programowania obiektowego oraz pracą w Solution Eksplorerze.

Klasa to pewnego rodzaju szablon do tworzenia obiektów.
Zawiera zbiór własności (zmiennych) oraz zachowań (metod) opisujących dany obiekt klasy.
Klasa może opisywać każdą dowolną rzecz (Krzesło, Pisak, Osoba).

Obiekt – tak jak klasa jest szablonem, przepisem jakiejś rzeczy, tak obiekt jest to stworzony na podstawie tego szablonu konkretny element. Obiekt jest zatem egzemplarzem danej klasy, technicznie używa się także stwierdzenia, że obiekt jest instancją klasy.

Wyobraźmy sobie inną klasę – Pracownik. Pracowników mogą charakteryzować takie dane jak imię, nazwisko, data urodzenia, adres zamieszkania czy zarobki. Klasa Pracownik jedynie wyszczególnia (definiuje) dane opisujące pracowników, a obiektami tej klasy są konkretni pracownicy.

Nazewnictwo klas

Nazwa klasy powinna wprost mówić, jakie obiekty będzie można dzięki niej stworzyć Jeśli obiektami mają być ludzie, to dobrą nazwą dla takiej klasy jest wyraz ***Człowiek***. Obok obowiązkowych zasad dotyczących nazewnictwa w C# (takich samych jak w

przypadku nazw zmiennych i metod), istnieją dodatkowe zalecane standardy odnośnie nazywania klas. Zgodnie z tymi zaleceniami, nazwy klas powinno się tworzyć przy pomocy rzeczowników lub rzeczowników z przymiotnikami (zazwyczaj w liczbie pojedynczej). Ponadto zaleca się, aby nazwy klas zaczynać się z dużej litery, a jeśli nazwa ma kilka słów, kolejne słowa także powinny zaczynać się z dużej litery, np. ***WysokiCzlowiek***, jest to tak zwany ***PascalCase***.

Modyfikatory dostępu

W C# mamy cztery podstawowe modyfikatory dostępu:

- public – dostęp do elementów wszędzie,
- internal – dostępność w ramach jednego projektu w solucji (w sytuacji, gdyby projektów było więcej niż jeden),
- protected – prywatne poza klasą, ale dostępne dla klas pochodnych
- private – dostęp do elementów tylko wewnątrz klasy

Tworzenie klasy

```
ModyfikatorDostepu class NazwaKlasy
{
   Własności
   Metody
}
```

Przykładowa klasa:

```
public class Osoba
```

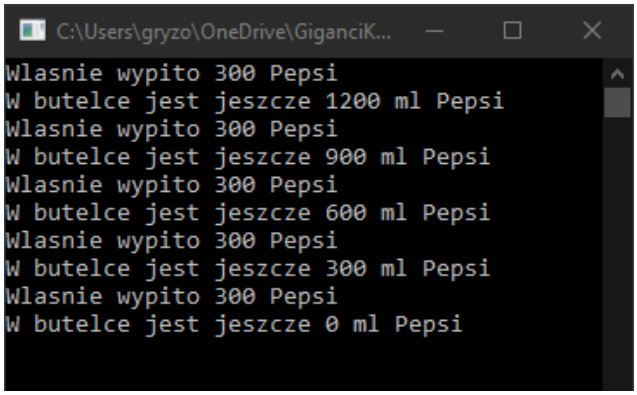
```
{
    // własności klasy
    public string Imie;
    public string Nazwisko;

    // metoda klasy
    public void PustaMetodaKlasy()
    {
    }
}
```

Dodatkowa misja Programisty:

Napisz aplikację konsolową, której zadaniem będzie zasymulowanie picia wody z butelki. W projekcie należy stworzyć klasę ***Butelka***, która powinna mieć własności (*TypNapoju*, *AktualnaIloscNapoju*) oraz metody (*Wypij*, *SprawdzIlosc*)

- Metoda **Wypij** powinna przyjmować ilość napoju do wypicia i jej zadaniem jest wyświetlenie informacji na konsoli o wypiciu napoju oraz zmniejszenie aktualnej jego ilości w butelce
- Metoda **Sprawdźllość** powinna wypisywać na konsole, ile napoju zostało jeszcze w butelce
- Jeżeli nastąpi próba wypicia większej ilości napoju niż znajduje się aktualnie w butelce powinien pojawić się komunikat o braku możliwości wypicia napoju.

Przykładowy wynik działania programu:


C:\Users\gryzo\OneDrive\GiganciK...
Wlasnie wypito 300 Pepsi
W butelce jest jeszcze 1200 ml Pepsi
Wlasnie wypito 300 Pepsi
W butelce jest jeszcze 900 ml Pepsi
Wlasnie wypito 300 Pepsi
W butelce jest jeszcze 600 ml Pepsi
Wlasnie wypito 300 Pepsi
W butelce jest jeszcze 300 ml Pepsi
Wlasnie wypito 300 Pepsi
W butelce jest jeszcze 0 ml Pepsi


HTML
JS
C#