giganci programowania

Materiały powtórzeniowe

Lekcja 12-13. Metody

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 wprowadzenie do dzielenia kod na części i przedstawienie metod.

Metody to takie elementy kodu, które gromadzą jakąś funkcjonalność – przyjmują jakieś wartości (parametry), wykonują na nich dowolne operacje, a następnie zwracają wynik.

Jeżeli posiadamy fragment kodu, który będziemy uruchamiać więcej niż jeden raz, nie powinniśmy go po raz kolejny (**nie powielamy kodu!**). Kody powtarzające się w wielu miejscach należy przenosić do tzw. metod, które „wołają", tam gdzie są im potrzebne. Dlaczego? Np. gdy będziemy wprowadzać zmiany w jednej części, możemy zapomnieć wprowadzić je w drugiej, a tak mamy gwarancję, że uruchamiamy zawsze dokładnie ten sam kod. Pozwala to też skrócić liczbę linii kodu. Każda metoda, powinna odpowiadać najlepiej za jedną konkretną czynność i posiadać nazwę jednoznacznie na nią wskazującą np. ObliczPotęgę(), WyświetlWynik(). Unikamy metod zbiorczych i nazw nie pozwalających na określenie za co dana metoda odpowiada.

Dobrym przykładem są poznane na wcześniejszych lekcjach metody klasy Math wykonujące działania matematyczne. Przykładowo, dzięki metodzie Pow(x, y) możemy w naszym programie podnieść dowolną liczbę x do dowolnej potęgi y.

```
double x = 5;
double y = 2;
double wynik = Math.Pow(x, y);
```

Dzięki niej, nie musimy za każdym razem pisać kodu, który wykona te obliczenia. Wystarczy, że wywołamy już gotową metodę, która zwróci nam prawidłowy wynik. Dodatkowo do metody przekazujemy zmienne, które są potrzebne do wykonania w niej obliczeń. W powyższym przykładzie, musimy podać 2 liczby, na których ma zostać wykonane działanie.

Podsumowując - 3 główne cechy metod:

1. Wykonuje konkretne zadanie np. działanie matematyczne.

2. Przyjmuje tzw. parametry, czyli zmienne potrzebne do wykonania zadania np. liczby do działania (choć jeśli jego wykonanie ich nie wymaga, nie musi niczego przyjmować).

3. Zwraca wynik swojego działania w postaci odpowiedniej zmiennej np. wynik działania (choć jeśli nie jest potrzebny, nie musi niczego zwracać).

Cel metod - Do tej pory nasze programy składały się głównie z kilkunastu, maksymalnie kilkudziesięciu linii kodu. W rozwiniętych programach, potrafi ich być kilkaset tysięcy a nawet kilka milionów. Wydzielanie poszczególnych fragmentów kodu do metod służy do jego usystematyzowania, zwiększa znacząco jego czytelność i ułatwia znajdowanie błędów w systemie.

Ponadto ważną cechą jest wspomniana wcześniej możliwość wielokrotnego użycia metody w różnych miejscach programu. Dodatkowo, dzięki wydzielaniu kodu do metod, w łatwy sposób możemy modyfikować działanie naszego programu poprzez zmianę działania pojedynczej metody, bez konieczności modyfikowania dużej ilości kodu.

Istnieją trzy główne rodzaje metod:

1) metody nie przyjmujące parametrów i nie zwracające wartości
2) metody przyjmujące parametry, ale niczego nie zwracające
3) metody przyjmujące parametry i zwracające jakiś wynik do programu

Definicja metod

Definicja metody składa się z kilku elementów (elementy ujęte w nawiasy kwadratowe są opcjonalne). Pierwsza linia definicji, która obejmuje modyfikatory, typ, nazwę i listę argumentów stanowi deklarację metody. Deklaracja metody wraz z ciałem metody – stanowią definicję metody.

Składnia:

```
[Modyfikatory] Typ Nazwa ([Lista argumentów])
{
    [Ciało metody]
}
```

gdzie,

- Modyfikatory - określają zachowanie i dostępność metody,
- Typ – typ danej zwracanej przez metodę,
- Nazwa – nazwa metody, różna od nazwy klasy, w której została zdefiniowana,
- Lista argumentów – lista argumentów przekazywanych do metody,
- Ciało metody – inaczej treść metody, kod (zbiór instrukcji) realizujący działanie metody.

Przykłady:

1) Metoda nie przyjmuje parametrów i nie zwracająca wartości:

```
static void Main(string[] args)
{
    WypiszImie();
}

private static void WypiszImie()
{
    Console.WriteLine("Moje imie to Marek");
}
```

2) Metoda przyjmująca parametry, ale niczego nie zwracająca:

```
static void Main(string[] args)
{
    DodajDwieLiczby(5,10);
}

private static void DodajDwieLiczby(int liczba1, int liczba2)
{
    int wynik = liczba1 + liczba2;
    Console.WriteLine("Wynik dodawania to {0}", wynik);
}
```

3) Metoda przyjmująca parametry i zwracająca wynik działania do programu

```
static void Main(string[] args)
{
    float wynikDzialaniaMetody = OdejmijLiczby(5, 10);
}


private static float OdejmijLiczby(float liczba1, float liczba2)
{
    return liczba2 - liczba1;
}
```

Dodatkowa misja Programisty:

1. Napisz prosty kalkulator dostępny w konsoli, w którym:
   - Użytkownik poda znak działania, które chce wykonać
   - Poda pierwszą liczbę
   - Poda drugą liczbę
   - Zostanie wywołana odpowiednia metoda (każde działanie ma własną)
   - Na konsoli wyświetli się wynik

Dostępne będą podstawowe operacje matematyczne: +,-,*,/.

2. Stwórz prosty program do wyszukiwania tytułów książek w bazie biblioteki (lub piosenek, filmów czy gier w serwisie internetowym). Do zasymulowania bazy danych użyj tablicy z tytułami. Program powinien:

- Przyjąć od użytkownika tytuł,
- Wywołać metodę, która sprawdzi, czy tytuł jest dostępny w bazie i zwróci info o dostępności,
- W każdej z możliwości, program wyświetli użytkownikowi odpowiedni komunikat o dostępności.

3. Rozbuduj wykonywany na lekcji projekt *Kamień, papier, nożyce* dodając*:*

- Metodę wyświetlającą powitanie gracza
- Metodę wyświetlającą podsumowanie gry
- Statystyki zwycięstw:przegranych w ramach jednej sesji
- Tryb drugiego gracza