giganci programowania

Sposoby przechowywania danych

Celem lekcji jest powtórzenie wiadomości z ubiegłych dwóch części programowania w C# oraz przedstawienie sposobów przechowywania danych.

Powtórzenie wiadomości

- **Klasa** - szablon definiujący właściwości i zachowania obiektów. Klasa może zawierać pola, właściwości, metody, zdarzenia i inne elementy.
- **Obiekt** - instancja klasy, która może mieć własne wartości dla właściwości zdefiniowanych w klasie.
- **Metoda** - blok kodu w klasie, który wykonuje określone operacje. Metody mogą przyjmować parametry i zwracać wartości.
- **Pętla** - konstrukcja umożliwiająca wielokrotne wykonanie bloku kodu. W C# dostępne są różne typy pętli, takie jak **for, while i foreach**.

- **Lista -** struktura danych, która przechowuje zbiór elementów w określonej kolejności.
- **Programowanie obiektowe -** paradygmat programowania oparty na koncepcji "obiektów", które mogą zawierać dane i metody.

- **Dziedziczenie** - mechanizm, który pozwala jednej klasie (klasie pochodnej) przejmować

Rekord

- Rekordy w C# służą do przechowywania danych i są zoptymalizowane do niemutowalnych obiektów.
- Raz utworzony obiekt **nie można** zmienić.

```
public record Osoba(string Imie, int Wiek);
```

Słownik

- Słownik (Dictionary) w C# to kolekcja przechowująca pary klucz-wartość.
- Klucz musi być **unikalny** i służy do identyfikacji wartości, która jest z nim powiązana.

```
Dictionary<int, string> giganci = new Dictionary<int, string>();
```

Kolejka

- Kolejka to struktura danych, które przechowuje elementy w kolejności FIFO (First In, First Out).
- Elementy są dodawane na **końcu** kolejki i usuwane z jej **początku**.

```
Queue<string> kolejka1 = new Queue<string>();
```

HashSet

- HashSet to kolejka, która przechowuje **unikalne** elementy bez określonej kolejności.

```
HashSet<int> numery = new HashSet<int>();
```

Zbiory

- **Suma** zbiorów łączy dwa zbiory, zawierając wszystkie unikalne elementy.
- **Różnica** zbiorów zawiera elementy z jednego zbioru, które nie są obecne w drugim zbiorze.
- **Część wspólna** zbiorów zawiera elementy obecne w obu zbiorach.

```
// Suma zbiorów (Union)
set1.UnionWith(set2);
Console.WriteLine("Union: " + string.Join(", ", set1)); // 1, 2, 3, 4, 5, 6, 7, 8
// Część wspólna zbiorów (Intersection)
set1.IntersectWith(set2);
Console.WriteLine("Intersection: " + string.Join(", ", set1)); // 4, 5
 // Różnica zbiorów (Difference)
set1.ExceptWith(set2);
Console.WriteLine("Difference: " + string.Join(", ", set1)); // 1, 2, 3
```