# 3 - Strukturované datové typy

* Objekty, které se skládají z několika členů
* K jednotlivým členům lze přistupovat pomocí selektorů
* Homogenní – komponenty jsou stejného typu
* Heterogenní – komponenty jsou různého typu

# Výčtový typ

* Konečná množina hodnot a jejich identifikátorů
* Hodnota je obvykle reprezentována číslem (defaultně od 0)

# Objekt

* => instance
* Místo v paměti alokované a nakonfigurované třídou/strukturou
* Může být uložen v proměnné/kolekci
* Základní stavební kámen OOP

# Struktura

* Skupina více proměnných dohromady (mohou mít různé datové typy)
* Hodnotový typ
* Nepodporuje dědičnost, nemá konstruktor
* Struktura x Třída
  + Struktura je hodnotová, třída referenční
  + Struktura je uložena na zásobníku, třída na haldě
  + Struktura nepodporuje dědičnost
  + Struktura nemůže mít bezparametrický kontruktor

# Pole

* Množina proměnných stejného datového typu
* S polem se zachází jako s celkem, k jednotlivým proměnným se přistupuje přes index
* V C# se indexuje od nuly
* Pevný počet položek
* pole.Length – celkový počet prvků ve všech dimenzích
* pole.GetLength(int x) – celkový počet prvků v dimezi x (pro více rozměrná pole)

# Kolekce + Generické kolekce

* Kolekce jsou standardní datové struktury doplňující pole
* System.Collections:
  + Nespecifikovaný datový typ, kolekce mohou obsahovat různé datové typy
  + Prvky získané z kolekce jsou typu object, pro uložení do proměnné jiného typu je potřeba přetypovat
  + SortedList
  + Stack
  + Queue
  + Hashtable
* System.Collections.Generic:
  + Datový typ se specifikuje ve chvíli vytvoření instance
  + Generický typ slouží jako zástupce pro budoucí datový typ
  + Dictionary<TKey,TValue>
  + List<T>
  + Queue<T>
  + SortedList<TKey,TValue>
  + Stack<T>

# Thread-Safe kolekce

* Bezpečné pro přístup z více vláken
* Umožňují bezpečně a efektivně přidávat nebo odebírat položky bez další dodatečné synchronizace v uživatelském kódu
* HashTable, ConcurrentStack, ConcurrentQueue