**Dictionary**

**Plan:**

* Dictionary nədir? Nə zaman istifadə olunur?
* Dictionary yaratmaq və element əlavə etmək;
* Dictionary-dən məlumat çəkmək və yoxlamaq;
* Elementləri silmək;
* Dictionary üzrə iterasiya;
* Qabaqcıl xüsusiyyətləri və optimizasiyası
* Dictionary ilə bağlı müsahibə sualları

**Dictionary nədir?**

**Dictionary<TKey, TValue> C# - da açar – dəyər cütlərini saxlamaq üçün istifadə olunan kolleksiya tipidir. Bu, verilənlərə indekslə deyil, unikal açarla müraciət etməyə imkan verir.**

**Yaddaşda Heap və Stack bölmələrdə müxtəlif formalarda saxlanılır. Dictionary reference type olduğu üçün object kimi heap-da saxlanılır. Lakin içindəki TKey və TValue tiplərindən asılı olaraq məlumatlar həm stack, həm də heap-da saxlanıla bilər. Məsələn,**

**Dictionary<int, string> dict = new Dictionary<int, string>();**

**dict.Add(1, “Leyla”); - Burada, dict dəyişəni stack-da saxlanılır, Dictionary<int, string> əmri isə heap-da yeni object yaradır və stack-dakı dict ona göstərici olur.**

**Int (TKey) value type olduğu üçün stack-da saxlanılır.**

**String (TValue) reference type olduğu üçün isə heap-da saxlanılır, stack-dakı dict onun göstəricisi olur.**

**Nə zaman istifadə olunur?**

* Məlumatları tez-tez əldə etmək lazım olduqda (search və lookup əməliyyatları üçün);
* Unikal açarlar ilə əlaqəli dəyərləri saxlamaq lazım olduqda;
* Konfiqurasiya və parametrləri saxlamaq lazım olduqda;
* API və JSON formatında gələn dataları saxlamaq və işlətmək lazım olduqda.

**Dictionary vs List**

|  |  |  |
| --- | --- | --- |
| **Xüsusiyyət** | **Dictionary** | **List** |
| Müraciət metodu | Açar ilə (key) | İndeks ilə |
| Məlumat əldə etmə sürəti | O(1) çox sürətlidir | O(n) element sayl artdıqca yavaşlayır |
| Dublikatlara icazə | Açarlar unikal olduğu üçün təkrarlana bilməz. | Dublikat elementlər ola bilər |
| Tətbiq sahəsi | Key-value əsaslı saxlama və lookup əməliyyatları | Sırf indekslə işləmə |

**Dictionary yaratmaq.**

C# - da Dictionary**<TKey, TValue> istifadə edərək açar – dəyər cütləri saxlaya bilərik. TKey açar tipini, TValue isə dəyər tipini göstərir. Dictionary yaratmağın bir neçə üsulu var:**

1. Boş Dictionary yaratmaq:  
   **Dictionary<int, string> students = new Dictionary<int, string>();  
   Burada int açar, string isə dəyər tipidir.**
2. **Başlanğıc dəyərlərlə Dictionary yaratmaq:  
   var students = new Dictionary<int, string>()  
   {  
    {1, “Leyla”},  
    {2, “Həsən”},  
    {3, “Aysel”}  
   };  
   Burada var istifadə edərək Dictionary<int, string> avtomatik təyin edilir.**
3. **Dictionary üçün KeyValuePair istifadə etmək:  
   var students = new Dictionary<int, string>()  
   {  
    new KeyValuePair(1, “Leyla”),  
    new KeyValuePair(2, “Həsən”)  
   };  
   Bu üsul da eyni işi görür, lakin KeyValuePair<TKey, TValue> obyektindən istifadə edir. Daha səliqəli kod yazmağa imkan verir, açar və dəyəri aydın şəkildə oxumağa kömək edir. Əgər həm açar, həm də dəyər qaytarılmalıdırsa, KeyValuePair yaxşı seçimdir. Ondan List və ya başqa kolleksiyalarda da açar – dəyər tipli data saxlamaq üçün istifadə etmək olar.  
     
   Dictionary vs KeyValuePair**

|  |  |
| --- | --- |
| Bütün açar – dəyər cütlərini saxlayır. | Bir açar – dəyər cütü saxlayır. |
| Add, Remove, ContainsKey kimi metodları var. | Yalnız key və value xassələri var. |
| Açarlar unikal olmalıdır. | Tək bir cütü saxladığı üçün belə məhdudiyyəti yoxdur. |
| Performans üçün optimallaşdırılmışdır | Sadəcə məlumat struktur kimi istifadə edilir. |

Dictionary-ə elemet əlavə etmək üçün:

1. Add metodu:  
   **students.Add(4, “Orxan”);  
   students.Add(5, “Nİgar”);**Add metodu eyni açarla təkrar element əlavə etməyə imkan vermir və əks halda, ArgumentException qaytarır.
2. [ ] operatoru ilə əlavə etmək, yaxud dəyəri yeniləmək:  
   **students[6] = “Elvin” –** Yeni dəyər əlavə edir;  
   **students[3] = “Lalə”** – mövcud açarın dəyərini yeniləyir

Dictionary – də elementlərin sayını bilmək üçün Count istifadə olunur:  
**students.Count;**