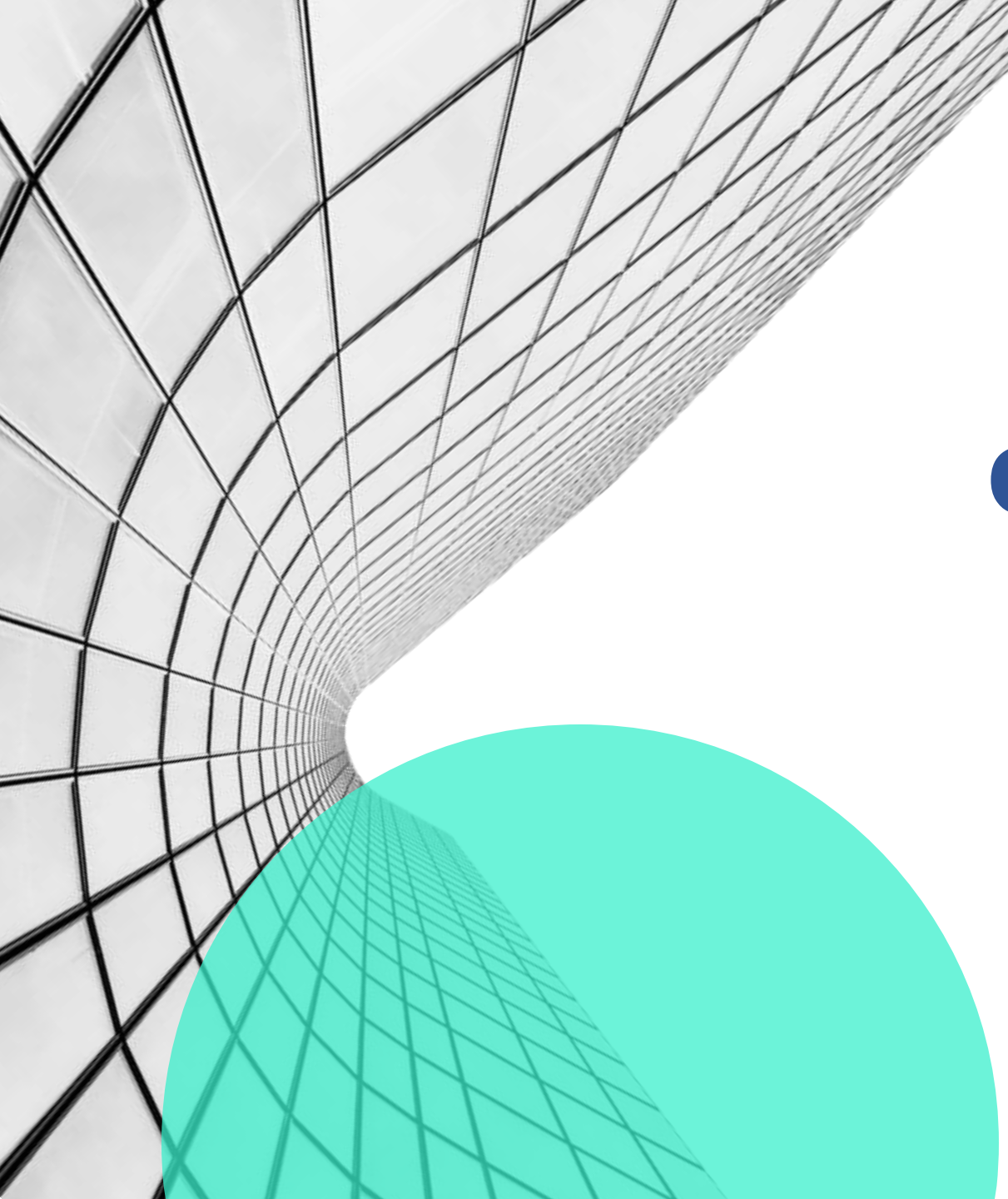

COLLECTIONS

Eng"Abdelrahman Sorour

OVERVIEW

LinkedList

Queue

Stack

NonGeneric Collection – HashTable

Generic Collection - Dictionary (HashTable)

Generic Collection - SortedDictionary (BST)

Generic Collection - SortedList (TwoArray)

Generic Collection - HashSet (HashTable)

Generic Collection - SortedSet (BBST)

LinkedList

Structure:

LinkedList consists of nodes where each node contains a data field and a reference(link) to the next node in the list. In C#, LinkedList is the generic type of collection which is defined in System.Collections.Generic

Business Case:

Useful when frequent insertions and deletions are required

Example:

```
LinkedList<int> Numbers = new LinkedList<int>();
Numbers.AddFirst(1);
Numbers.AddAfter(Numbers.First, 2);
Numbers.AddLast(3);
foreach (int number in Numbers)
{
    Console.WriteLine(number);
}
```

Time Complexity:

- AddFirst, AddLast: O(1)
- AddAfter/AddBefore: O(1) (if position
- is known)
- Remove, Search: O(n)

Stack

Structure:

represents a last-in, first out collection of object. It is used when you need a last-in, first-out access to items, This class comes under System.Collections.Generic namespace.

Example:

```
Stack<int> Numbers = new Stack<int>();
Numbers.Push(1);
Numbers.Push(2);
Numbers.Push(3);
Console.WriteLine(Numbers.Pop());
Console.WriteLine(Numbers.Pop());
Console.WriteLine(Numbers.Pop());
Console.WriteLine(Numbers.TryPop(out int
LastNumber));
foreach (int number in Numbers)
{
    Console.WriteLine(number);
}
```

Business Case:

Managing presenting different screens to a user as they navigate around them. Showing a screen pushes it to the stack, going 'back' pops it

Time Complexity:

- Push, Pop, Peek: O(1)

Queue

Structure:

Represents a first-in, first-out collection of objects, This class comes under System.Collections.Generic namespace.

Example:

```
Queue<int> Numbers = new Queue<int>();
Numbers.Enqueue(1);
Numbers.Enqueue(2);
Numbers.Enqueue(3);
Console.WriteLine(Numbers.Dequeue());//1
Console.WriteLine(Numbers.Dequeue());//2
Console.WriteLine(Numbers.Dequeue());//3
Console.WriteLine(Numbers.TryDequeue(out int
LastNumber));
foreach (int number in Numbers)
    Console.WriteLine(number);
```

Business Case:

Managing customer service requests in the order they are received, Like Order Processing, Task Scheduling, Ticket Reservation System

Time Complexity:

- Enqueue O(1) (amortized)
- Dequeue O(1)
- Peek O(1)
- Contains O(n)
- Clear O(1)
- Count O(1)

NonGeneric Collection - HashTable

Structure:

Represents a collection of key/value pairs that are organized based on the hash code of the key (not type-safe) . This class comes under System.Collections namespace.

Example:

```
Hashtable Note = new Hashtable();
Note.Add("Ahmed", 123);
Note.Add("Hamed", 748);
Note.Add("Abdelrahman", 234);
if (!Note.ContainsKey("Abdelrahman"))
    Note.Add("Abdelrahman", 999);
foreach (DictionaryEntry person in Note)
{
    Console.WriteLine($"{person.Key}");
}
```

Business Case:

Counting Unique Elements, Storing

Application Settings, aching

Frequently Accessed Data

Time Complexity:

	Avg	Worst
•Add (Insert)	O(1)	O(n) (due to hash collisions)
•Remove (Delete)	O(1)	O(n) (due to hash collisions)
•Lookup (Search)	O(1)	O(n) (due to hash collisions)
•ContainsKey	O(1)	O(n)
•Iteration	O(n)	O(n)

Generic Collection - Dictionary (HashTable)

Structure:

Represents a collection of keys and values (type-safe), This class comes under System.Collections.Generic namespace.

Business Case:

Customer Support Ticket System

Example:

```
Dictionary<string, int> Note = new Dictionary<string, int>();
Note.Add("Ahmed", 123);
Note.Add("Omar", 342);
Note.Add("Amany", 432);
if (!Note.ContainsKey("Abdelrahman"))
Note.Add("Abdelrahman", 999);
foreach (KeyValuePair<string, int> person in Note)
 {
   Console.WriteLine($"{person.Key} :: {person.Value}");
 }
```

Time Complexity:

	Avg	Worst
•Add	O(1)	O(n) (due to hash collisions)
•Delete	O(1)	O(n) (due to hash collisions)
•Lookup (Search)	O(1)	O(n) (due to hash collisions)

Generic Collection - SortedDictionary (BST)

Structure:

stores key-value pairs in sorted order based on the key. It is implemented as a binary search tree (BST), providing guaranteed O(log n) performance for lookups, insertions, and deletions..

Business Case:

Price-Based Product Filtering,

Dynamic Fee Calculation System

Example:

```
SortedDictionary<string, int> Note = new
SortedDictionary<string, int>();
Note.Add("Omar", 444);
Note.Add("Ziad", 222);
Note.Add("Fady", 111);
foreach (var person in Note)
{
    Console.WriteLine($"{person.Key}
:{person.Value}");
}
```

Time Complexity:

- **Add (Insert)** O(logn)
- **Remove (Delete)** O(logn)

Generic Collection - SortedList (TwoArray)

Structure:

Stores key-value pairs in sorted order. Unlike SortedDictionary<TKey, TValue>, which uses a binary search tree (BST), the SortedList<TKey, TValue> uses two arrays to store the keys and values, one for the keys and another for the corresponding values.
The key features of SortedList make it unique for specific use cases where both efficient access by index and sorted order are important.

Example:

```
SortedList<string, int> SortedNote = new
SortedList<string, int>();
SortedNote.Add("Omar", 444);
SortedNote.Add("Ziad", 222);
SortedNote.Add("Fady", 111);
foreach (var person in SortedNote)
{
    Console.WriteLine($"{person.Key} ::
{person.Value}");
}
```

Business Case:

In an e-commerce application, products need to be stored in a catalog, sorted by price, so customers can filter by price range efficiently.

Time Complexity:

- Add (Insert) O(n)
- Remove (Delete) O(n)

Generic Collection - HashSet (HashTable)

Structure:

provides a set data structure, backed by a hash table (similar to a Dictionary but with no associated values). It is primarily used to store unique elements and offers fast lookup, addition, and removal operations.

Example:

```
HashSet<int> Numbers = new HashSet<int>();
Numbers.Add(1);
Numbers.Add(2);
Numbers.Add(3);
Numbers.Add(1);
foreach (int number in Numbers)
{
    Console.WriteLine(number);
}
```

Business Case:

Avoiding Duplicate Transactions in a Banking System, Detecting Plagiarism in Educational Software, Ensuring Unique Email Addresses in a Newsletter System

Time Complexity:

- **Add (Insert)** O(1) but O(n) in worst case
- **Remove (Delete)** O(1) but O(n) in worst case

Generic Collection - SortedSet (BBST)

Structure:

provides a set-like data structure backed by a balanced binary search tree (BBST). It ensures uniqueness of elements and automatically maintains them in sorted order based on the natural comparer for the type or a custom comparer.

Example:

```
SortedSet<int> Numbers = new SortedSet<int>();
Numbers.Add(1);
Numbers.Add(2);
Numbers.Add(3);
Numbers.Add(1);
foreach (int number in Numbers)
 {
    Console.WriteLine(number);
 }
```

Business Case:

Range Queries in Stock Price Monitoring, Autocomplete Suggestions, Managing Ranks in a University System

Time Complexity:

- Add (Insert) O(log n)
- Remove (Delete) O(log n)
- Contains O(log n)

THANK YOU

Eng"Abdelrahman Sorour

abdelrahman-sorour (LinkedIn)

abdosorour10@gmail.com