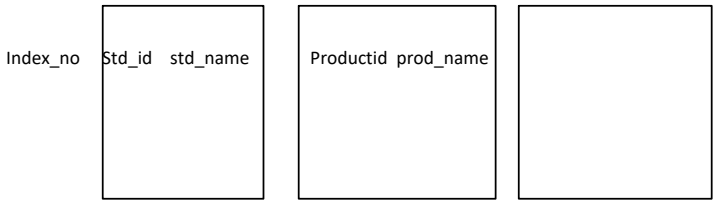
Collection

Friday, January 31, 2025 9:01 AM



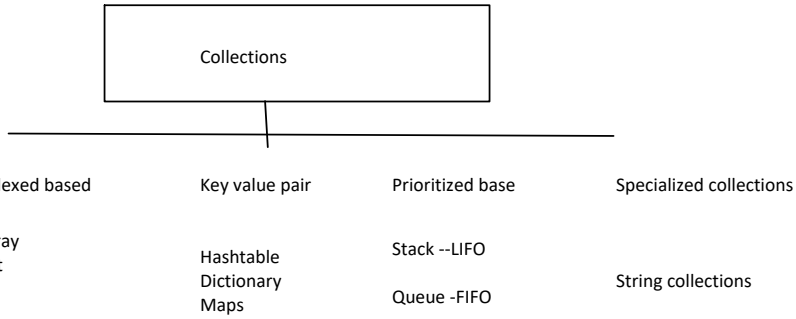
Collections are similar to Arrays :

Collection provides more flexible way of working with group of objects .

Collections --- Collections is a framework , Group of records

Key= : Value=

Categories of collections



Using System.Collections

1. Collections in c# are reusable , more powerful , more efficient ,quality and performance is also good
2. Size can be increased dynamically
3. We can insert an element into the middle of a collection
4. We can also remove or delete elements from the middle of a collection

Three ways to work with collections :

1. System.Collections.Generic --- > Strict types for eg: when we say the list is of integer type
2. System.Collection -
3. Syste.collections.concurrently

Non-generic --- > System.Collections

1. ArrayList
2. Stack
3. Queue
4. Hashtable
5. SortedList

generic --- > System.Collection.Generic

1. List<T>
2. Stack<T>
3. Dictionary<Tkey, Tvalue>
4. LinkedList<T>

Concurrent Collections

1. ConcurrentBag<T>
2. ConcurrentQueue<T>
3. ConcurrentStack<T>

Let's say one university wants to manage student records . The system should allow

1. Adding students(id , name and age)
2. Display student list
3. Remove a student by id if present(to check if that student exist or not)

Create a student class with the properties id, name and age

Then create a list type of student

List<student> l = new list<student>();

-- > Remove
--> Find()

Remove()
RemoveAt(int index)
RemoveRange(int index , int count)

Count -- > Property

Clear()

Clone()

Array

Difference between ArrayLIst and HashTable

Look up : Arraylist can be only looked up via index no. And hashtable via custom defined key
Performance : Arraylist is faster than Hashtable because in Hashtable we have to perform some extra task

Arraylist

Hashtable

Selection of collection on the basis of user requirement & Performance

Collection | Category	Ordering	Contiguous Storage?	Direct Access?	Lookup Efficiency	Manipulate Efficiency	Notes
HashSet	Unordered	Yes	Via Key	Key:O(1)	O(1)	It's unique and unsorted, unlike a dictionary.
Dictionary	Unordered	Yes	Via Key	Key:O(1)	O(1)	For high-performance searches, it's the top choice.
Stack	LIFO	Yes	Only Top	Top: O(1)	O(1)*	It's basically the same as List, except it's LIFO
Queue	FIFO	Yes	Only Front	Front: O(1)	O(1)	It's basically like List, except it's FIFO
List	You can order elements however you want	Yes	Via Index	Index: O(1) Value: O(n)	O(n)	Smaller lists benefit from direct access and no sorting.
LinkedList	You can order elements however you want	No	No	Value:O(n)	O(1)	This is best if you want to insert/ delete in the middle and don't need direct access to the list.
SortedList	Sorted	Yes	Via Key	Key:O(log n)	O(n)	Just like SortedDictionary, but with an array instead of a tree, so it loads slower but looks up data faster.
SortedSet	Sorted	No	Via Key	Key:O(log n)	O(log n)	Sorted collection with the same key and value as SortedDictionary.
SortedDictionary	Sorted	No	Via Key	Key: O(log n)	O(log n)	It uses a binary search tree because it's a compromise between dictionary speed and ordering.

Let's say we employee type of collection where we id , name , designation and salary

On the basis of user's input as Id remove the record of that employee or if you want to update the salary with new basic pay

Printer based application -- (FIFO)

Create a printer class : attributes : pages . Photocopy
Collection add some

Interface // comaparable (Sortedlist)

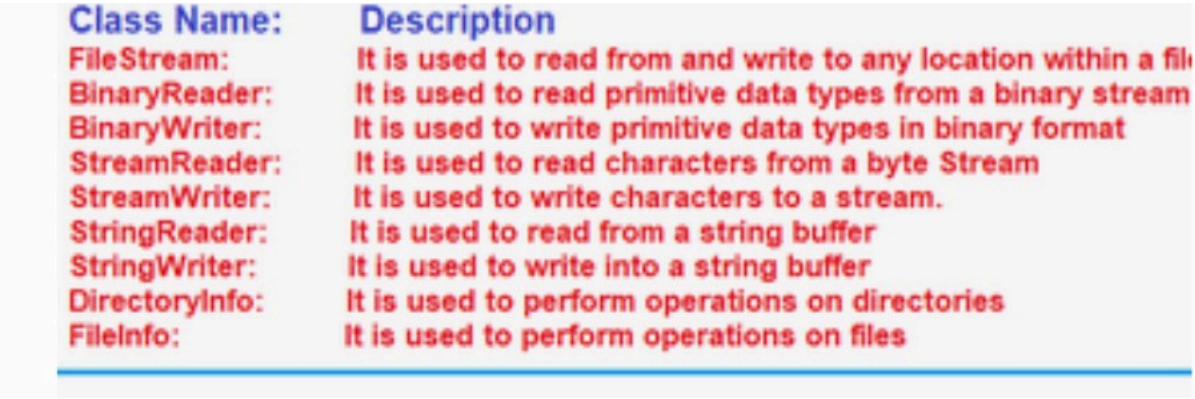
File Handling : the data related information you want to read or write (Stream) in a file

Stream -- Sequence of bytes travelling from source to destination over a communication path. (read -- input stream - Keyboard -- file writing -- Output stream - Console)

For file handling we perform three basic operations:

1. Read
2. Write
3. Append

System.IO ---> File , Directory , Path



Class Name:	Description
FileStream:	It is used to read from and write to any location within a fil
BinaryReader:	It is used to read primitive data types from a binary stream
BinaryWriter:	It is used to write primitive data types in binary format
StreamReader:	It is used to read characters from a byte Stream
StreamWriter:	It is used to write characters to a stream.
StringReader:	It is used to read from a string buffer
StringWriter:	It is used to write into a string buffer
DirectoryInfo:	It is used to perform operations on directories
FileInfo:	It is used to perform operations on files

Create a program that will creates a log folder (Logs/) if it doesn't exist and then add user details (name , age , email and timestamp)to a log file . Read and then prints all user details line by line.