Campus Nyköping

PROGRAMMERING I .NET C#1 - 19

OOP6 - Collections

Förra gången

- G-uppgiften
- VG-uppgiften
- Generic Methods
  - Code Along Demo
  - Code Along ConsoleWindow-projektet

Idag

- Upprop
- Collections

Kollektioner

- C#-kollektioner är utformade för att lagra, hantera och manipulera liknande data mer effektivt. Datamanipulering omfattar att lägga till, ta bort, söka efter och infoga data i samlingen.
- Kollektioner implementerar följande vanliga funktioner:
  - Lägga till och infoga objekt i en samling
  - Ta bort objekt från en samling
  - Hitta, sortera, söka efter objekt
  - Byta ut objekt
  - Kopiera och klona samlingar och objekt
  - Egenskaperna Capacity och Count för att hitta samlingens kapacitet och antalet objekt i samlingen
  - .NET stöder två typer av samlingar, generiska och icke-generiska samlingar.

Array

- Enklaste formen av kollektion är en array:

```
int[] numbers = new int[] { 345, 72, 13};
for(int i = 0; i < numbers.Length; i++)
{
	Console.WriteLine(numbers[i]);
}
```

Non-Generic collection

- I icke-generiska samlingar kan varje element representera värden av **olika typer**.
- Storleken är inte fast.
- Objekt från samlingen kan läggas till eller tas bort vid körning.

ArrayList

- Klassen ArrayList är en samling som kan användas för alla datatyper och objekt.
- Arraylist liknar en matris, men den kan användas för att lagra värden av olika slag.
- En Arraylist har ingen specifik storlek.
- Valfritt antal element kan lagras.

```
ArrayList list = new ArrayList();
list.Add(47);
list.Add(“Hej”);
```

- Ta bort:

```
List.Remove("Hej");
```

HashTable

- HashTable liknar arraylist men representerar objekten som en kombination av en nyckel och ett värde.
- En hashtable sparar posterna slumpmässigt

```
Hashtable hashTable = new Hashtable();
hashTable.Add(”Mindata”, "Hej världen");
hashTable.Add(2, “Två”);
```

- Ta bort:

```
hashTable.Remove(“Mindata");
```

SortedList

- Är en klass som har kombinerar arraylist och hashtable.
- Representerar data som ett nyckel- och värdepar.
- Ordnar alla objekt i sorterad ordning.
- Nycklarna måste vara av samma typ

```
SortedList sorted = new SortedList();
sorted.Add(“Post B”, "Ett");
sorted.Add(“Post A”, "Två");
// I en foreach-loop visas Post A först
```

Stack

- En stack är en LIFO-datastruktur (sist in först ut).
- Tänk på stack som en samling objekt där allt du sätter in i en hög kommer att placeras högst upp och om du behöver ta bort något kommer det att tas bort från toppen.
- En bunt tallrikar eller en bokstapel är två vanliga exempel på en stapel.

```
Stack stack = new Stack();
stack.Push(“En text");
stack.Push(97);
```

```
Ta bort:
stack.Pop() // Tar bort 97 från stacken
```

Queue

- En queue i C# representerar en FIFO-samling (först in, först ut) med objekt. Ett exempel på en kö är en rad människor som väntar.

```
Queue queue = new Queue();
queue.Enqueue(“Post 1");
queue.Enqueue(“Post 2");
```

- Ta bort:

```
queue.Dequeue() // Tar bort ”Post 1”
```

Generic collection

- Generiska kollektioner fungerar med den **specifika typ** som anges i programmet medan icke-generiska samlingar fungerar på objekttypen.
- Specifik typ(int, string, object…)
- Matrisstorleken är inte fast
- Element kan läggas till / tas bort vid körning

List

- List sparar alla poster I ordning
- Måste ange datatyp
- Har många funktioner för sortering m m
- Kanske den mest vanliga kollektionen

```
List<string> texts = new List<string>();
texts.Add("Det");
texts.Add("var");
texts.Add("en gång");

Ta bort:
texts.Remove("Det");
texts.RemoveAt(index);
```

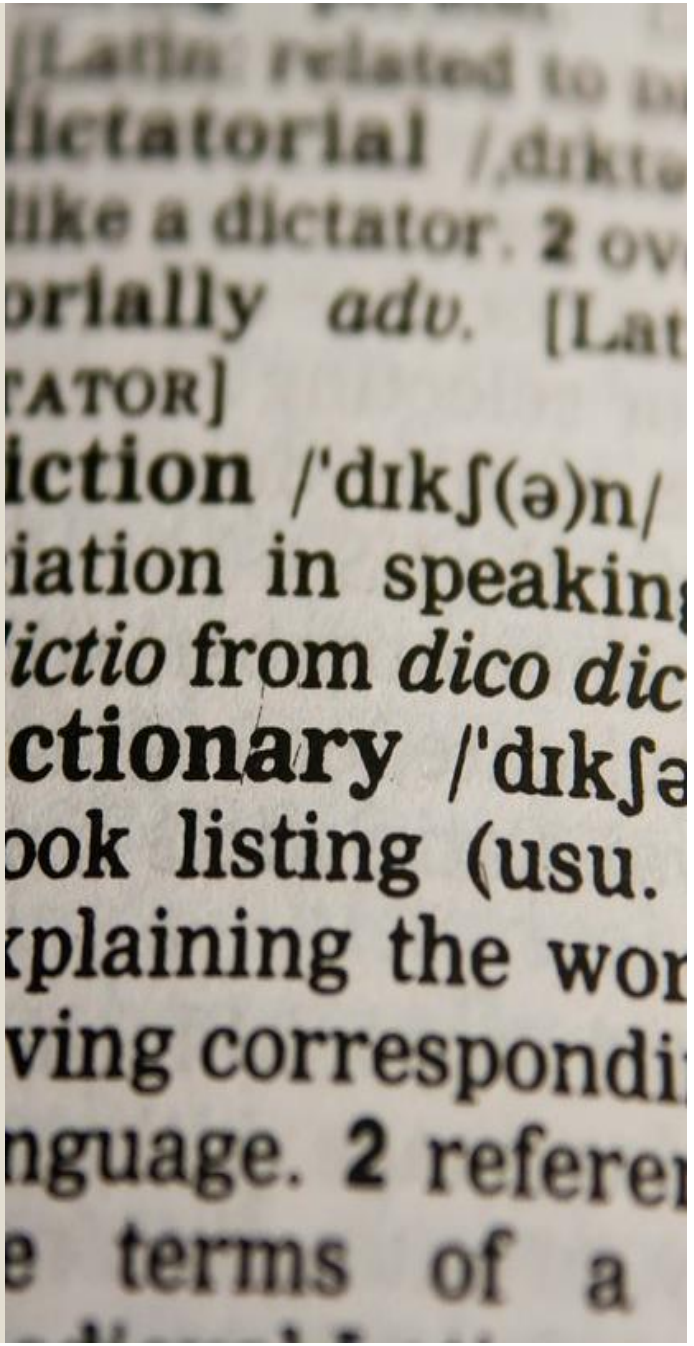
Dictionary

- Dicitionary liknar Hashtable och representerar objekten som en kombination av en nyckel och ett värde.
- Kräver att man anger datatyper

```
Dictionary<string, int> dict = new Dictionary<string, int>();
dict.Add(“Siffra 1", 24);
dict.Add(“Siffra 2", 95);
```

- Ta bort

```
dict.Remove("Siffra 2");
```



SortedList, Stack och Queue

- Det går att ange datatyp på vissa av de icke-generiska kollektionerna
- `SortedList<string, string> sl = new SortedList<string, string>();`
- `Stack<string> stk = new Stack<string>();`
- `Queue<string> q = new Queue<string>();`

Code Along - Collections

Länkar

- Collections in C# (c-sharpcorner.com)
  - ArrayList in C# (c-sharpcorner.com)
  - Hashtable in C# (c-sharpcorner.com)
  - C# SortedList Tutorial (c-sharpcorner.com)
  - Working With Stack In C# (c-sharpcorner.com)
  - C# Queue Tutorial (c-sharpcorner.com)
  - C# List Tutorial (c-sharpcorner.com)
  - Dictionary In C# (c-sharpcorner.com)