Iterators and Comparators

SoftUni Team

Technical Trainers

Software University

https://about.softuni.bg/

Table of Contents

IEnumerable<T> and IEnumerator<T>

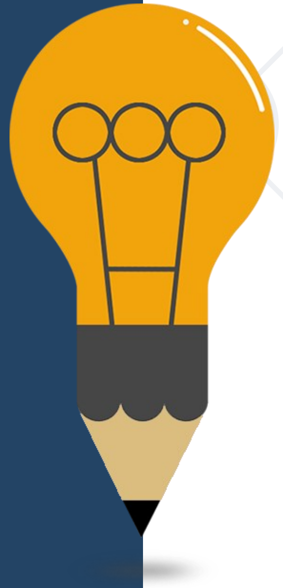
IEnumerable<T>

- .NET interface for sequences of elements
  - Enables **simple iteration** over a collection
- Contains a single method **`GetEnumerator()`**, which returns an **`IEnumerator<T>`**
- A class that implements the **`IEnumerable<T>`** can be **used in a foreach** loop traversal

IEnumerable<T> Example

```
public interface IEnumerable<T> : IEnumerable
{
  IEnumerator<T> GetEnumerator();
}
// Non-generic version
// (compatible with the legacy .NET 1.1)

public interface IEnumerable
{
  IEnumerator GetEnumerator();
}
```



IEnumerator<T>

- Provides the **sequential**, **forward-only iteration** over a collection of **any type**
- Methods:
  - **`MoveNext`**`()` – advances the enumerator to the next element of the collection
  - **`Reset`**`()` – sets the enumerator to its initial position
- Properties
  - **`Current`** – returns the element in the collection at the current position of the enumerator

IEnumerator<T> – Example

```
public interface IEnumerator<T> : IEnumerator
{
  bool MoveNext();
  void Reset();
  T Current { get; }
}
public interface IEnumerator
{
  bool MoveNext();
  void Reset();
  object Current { get; }
}
```

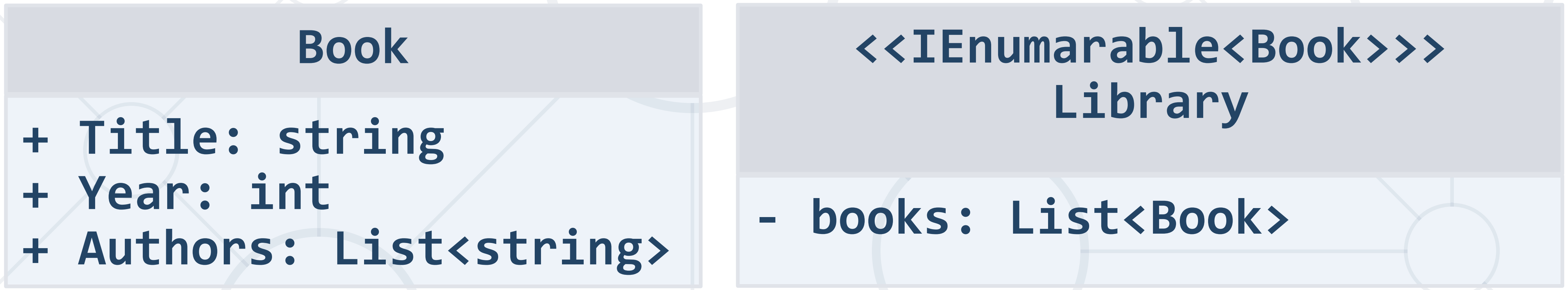
Yield Return

- Indicates that the **member**, in which it appears, is **an iterator**
- Simplifies the `IEnumerator<T>` implementations
- Returns **one element** upon **each** loop cycle
- Example: return iterator of integers 10, 20, 30, … 100

```
public IEnumerator<int> GetEnumerator()
{
  for (int num = 10; num <= 100; num++)
    yield return num;
}
```

Problem: Library Iterator (1)

- Create a class **Library**, which should store a collection of books and implement the **IEnumerable<Book>** interface



```
Book
+ Title: string
+ Year: int
+ Authors: List<string>
```

```
<<IEnumarable<Book>>>
Library
- books: List<Book>
```

Solution: Library Iterator

- Inside the **Library** class create method **GetEnumerator()**, which implements **IEnumerator<Book>** and use **yield return**

```
private List<Book> books;

public IEnumerator<Book> GetEnumerator()
{
  for (int i = 0; i < this.books.Count; i++)
    yield return this.books[i];
}
```

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/3183#12

Variable Number of Arguments: Params

- In C# methods can take variable number of arguments
  - Use the **params** keyword as shown below

```
void PrintNames(params string[] names)
{
  foreach(var name in names)
    Console.WriteLine(name);
}
```

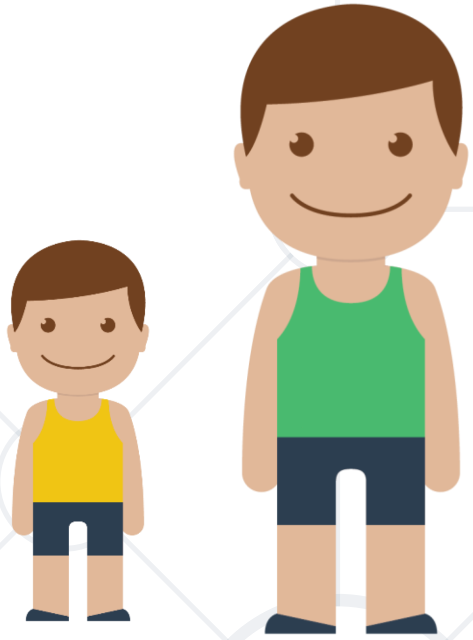
```
PrintNames("Pesho", "Stamat", "Jivko");
```

IComparable<T> and IComparer<T>

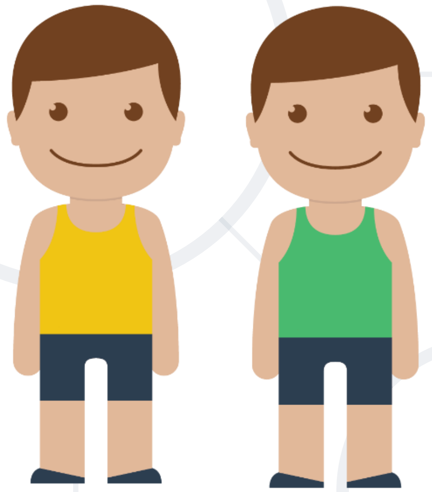
IComparable<T>

- Reads out as "**I am Comparable**"
- Provides a method of **comparing two objects** of a particular type – `CompareTo()`
- Defines the **default sort order** for a particular object type
- **Affects** original class

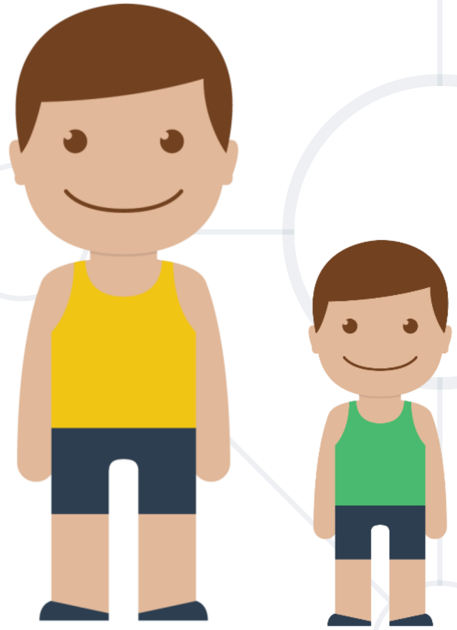
CompareTo(T) Method Returns



< 0



= 0

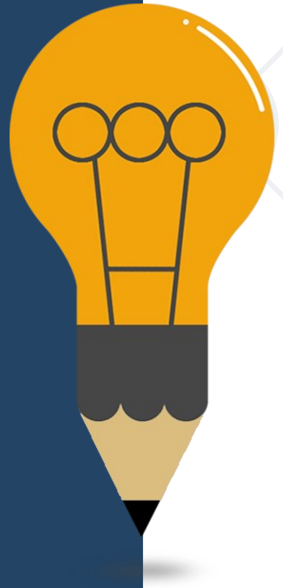


> 0

IComparable<T> – Example

```
class Point : IComparable<Point>
{
  public int X { get; set; }
  public int Y { get; set; }

  public int CompareTo(Point otherPoint)
  {
    if (this.X != otherPoint.X)
      return (this.X - otherPoint.X);
    if (this.Y != otherPoint.Y)
      return (this.Y – otherPoint.Y);
    return 0;
  }
}
```



IComparer<T>

- Reads out as "**I'm a comparer**" or "**I compare**"
- Provides a way to **customize** the **sort order** of a **collection**
- Defines a **method** that a type implements to **compare two objects**
- Doesn't **affect** original class

IComparer<T> – Example

```
class Cat
{
  public string Name { get; set; }
}
```

```
class CatComparer : IComparer<Cat>
{
  public int Compare(Cat x, Cat y)
  {
    return x.Name.CompareTo(y.Name);
  }
}
```

```
IComparer<Cat> comparer = new CatComparer();
var catsByName = new SortedSet(comparer);
```

Problem: Comparable Book

- Implement the `IComparable<Book>` interface in the existing class `Book` (which holds `Title` and `Year`)
  - First sort them in **ascending chronological** order (by year)
  - If two books are published in the **same year**, sort them **alphabetically**
- Override the `ToString()` method in your Book class so it returns a string in the format:
  - `"{title} - {year}"`
- Change your `Library` class so that **it stores the books** in the **correct** order

Solution: Comparable Book

```
public class Book : IComparable<Book>
{
  public string Title { get; set; }
  public int Year { get; set; }
  public int CompareTo(Book other)
  {
    int result = this.Year.CompareTo(other.Year);
    if (result == 0)
      result = this.Title.CompareTo(other.Title);
    return result;
  }
}
```

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/3183#13

Problem: Book Comparer

- Create a class **`BookComparator`**, which should implement the **`IComparer<Book>`** interface
- **`BookComparator`** must **compare two** books by:
  - Book title – **alphabetical order**
  - Year of publishing a book - from **the newest to the oldest**
- Modify your **`Library`** class once again to implement the new sorting

Solution: Book Comparer

```
public class BookComparator : IComparer<Book>
{
  public int Compare(Book x, Book y)
  {
    int result = x.Title.CompareTo(y.Title);
    if (result == 0)
    {
      result = y.Year.CompareTo(x.Year);
    }
    return result;
  }
}
```

Check your solution here: https://judge.softuni.org/Contests/Practice/Index/3183#14

Summary

- Iterators
  - **IEnumerable<T>** and **IEnumerator<T>**
  - **yield return**
- **Params**: accept multiple method parameters
- Comparators
  - **IComparable<T>** and **IComparer<T>**

Software University

Questions?

SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, softuni.org
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License