



Национална програма  
"Обучение за ИТ умения и кариера"  
<https://it-kariera.mon.bg>

Министерството на  
образованието и науката  
<https://www.mon.bg>



# Итератори

Iterators

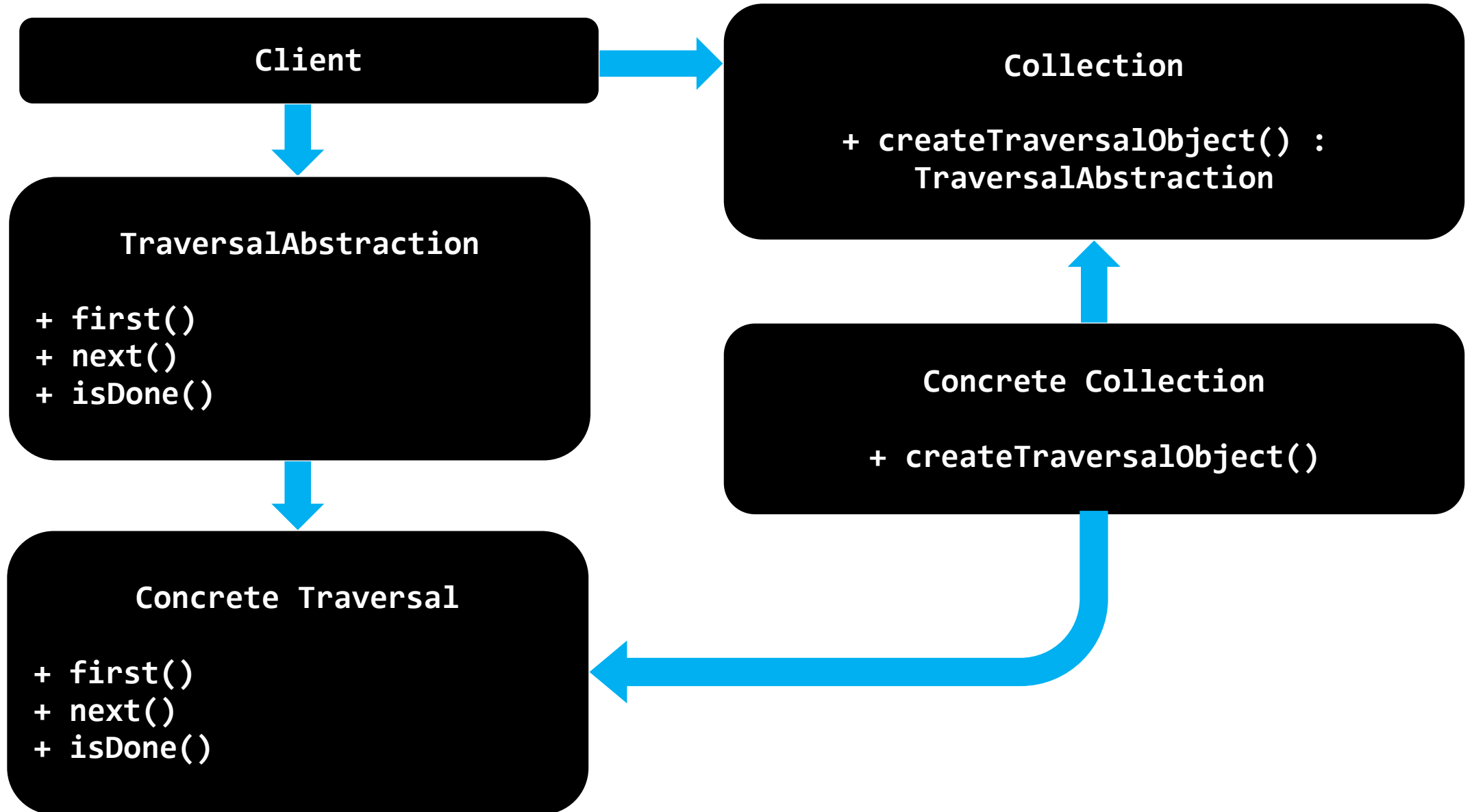
# Съдържание

1. Базови шаблони за дизайн (Design Patterns)
  - `Iterator` шаблон
2. Итератори
  - `IEnumerable<T>` интерфейс
  - `yield return`
  - `params`

# Шаблони в проектирането (Design Patterns)

- Общовалидни **повторяеми решения** на обичайни проблеми в софтуерния дизайн
- Предоставят **тествани** и **доказани** в разработката модели
- Подпомагат **четливостта** на кода за разработчици, вече запознати с тези шаблони

# Шаблон Iterator



# IEnumerable<T>

- Основен интерфейс в .NET, позволяващ просто обхождане на колекция
- Съдържа един-единствен метод `GetEnumerator()`, който връща един `IEnumerator<T>`
- Клас, реализиращ `IEnumerable<T>` може да бъде използван за обхождане с цикъла `foreach`

# IEnumerable<T> npumep

```
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>

- Предоставя последователно, едноръчно обхождане на колекция от произволен тип
- Методи
  - **MoveNext()** – премества итератора към следващия елемент в колекцията.
  - **Reset()** – връща итератора на началната му позиция
- Свойства
  - **Current** – връща елемента от колекцията, който е на текущата позиция на итератора

# IEnumerator<T> npumep

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

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



# Yield Return

- Указва, че **методът**, в който се появява, **е итератор**
- Опростява реализацията на **IEnumerator<T>**
- Връща **един елемент** за **всяко** повторение на цикъла

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

# Params

- Приема **променлив брой** параметри
- Само една **params** команда е допустима в декларацията на метод
- Трябва винаги да е последна

```
PrintNames("Pesho", "Stamat", "Jivko", "Stavri");  
  
public static void PrintNames(params string[] names)  
{  
    foreach(var name in names)  
        Console.WriteLine(name);  
}
```

# Задача: итератор за библиотека [1/2]

- Създайте клас **Library**, който трябва да съдържа колекция от книги и реализирайте **IEnumerable<Book>** интерфейса

## Book

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

## <<IEnumerable<Book>>> Library

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

# Задача: итератор за библиотека [2/2]

Вътре в класа Library създайте вложен клас LibraryIterator, който реализира **IEnumerator<Book>**

```
<<IEnumerator<Book>>>  
LibraryIterator
```

```
-currentIndex: int  
-books: List<Book>  
+ Current: Book
```

```
+Reset(): void  
+MoveNext(): bool  
+Dispose(): void
```

## Решение: оператор за библиотека [1/3]

```
public class Book
{
    public Book(string title, int year, params string[] authors)
    {
        this.Title = title;
        this.Year = year;
        this.Authors = authors;
    }

    public string Title { get; private set; }
    public int Year { get; private set; }
    public IReadOnlyList<string> Authors { get; private set; }
}
```

## Решение: итератор за библиотека [2/3]

```
public class Library : IEnumerable<Book>
{
    private List<Book> books;
    public Library(params Book[] books)
    {
        this.books = new List<Book>(books);
    }
    public IEnumerator<Book> GetEnumerator()
    {
        return new LibraryIterator(this.books);
    }
    IEnumerator IEnumerable.GetEnumerator()
    => this.GetEnumerator();
}
```

# Решение: итератор за библиотека [3/3]

```
private class LibraryIterator : IEnumerator<Book>
{
    private readonly List<Book> books;
    private int currentIndex;
    public LibraryIterator(IEnumerable<Book> books)
    {
        this.Reset();
        this.books = new List<Book>(books);
    }
    public void Dispose() {}
    public bool MoveNext() => ++this.currentIndex < this.books.Count;
    public void Reset() => this.currentIndex = -1;
    public Book Current => this.books[this.currentIndex];
    object IEnumerator.Current => this.Current;
}
```

# Обобщение

- Итератори
  - `IEnumerable<T>` интерфейса позволява просто обхождане на колекция с цикъл `foreach`
  - `IEnumerator<T>` интерфейса предоставя методите за обхождане на колекцията
- `yield return` връща отделния елемент от колекцията
- `Params` приема променлив брой параметри





Национална програма  
"Обучение за ИТ умения и кариера"  
<https://it-kariera.mon.bg>

Министерството на  
образованието и науката  
<https://www.mon.bg>



Документът е разработен за нуждите на Национална програма "Обучение за ИТ умения и кариера" на Министерството на образованието и науката (МОН) и се разпространява под свободен лиценз CC-BY-NC-SA (Creative Commons Attribution-Non-Commercial-Share-Alike 4.0 International).