SIT232: Object Oriented Development

IEnumerable: iterating interface

What is IEnumerable?

IEnumerable is an interface in C# that represents a sequence of elements. It allows objects to support iteration and can be used with foreach loops. By implementing IEnumerable, a class declares its ability to be enumerated. The key member is the GetEnumerator() method, which returns an IEnumerator object responsible for iterating over the elements. IEnumerable enables encapsulation and provides a consistent way to work with collections.

Advantages of IEnumerable:

- I**teration Support:** IEnumerable provides a standardized way to iterate over a collection of elements using foreach loops or other iteration constructs.
- **Encapsulation:** Implementing IEnumerable allows classes to hide their internal implementation details while still providing access to the elements in a controlled manner.
- **Code Reusability:** By conforming to the IEnumerable interface, classes can be easily used in different contexts and with various iteration-based operations, promoting code reuse

Implemetation:

The implementation syntax and way for IEnumerable is:

1.

```
public class Car : Vehicle, IEnumerable<int>
```

The "Car" class explicitly implements the '**IEnumerable<int>'** by defining the '**GetEnumerator()**' method given below:

```
public IEnumerator<int> GetEnumerator()
{
  foreach (int mileage in _mileageData)
  {
      yield return mileage;
  }
}
```

This method returns an '**IEnumerator<int>**' object responsible for iterating over the mileage data.

2.
```
foreach (int mileage in _mileageData)
        {
            yield return mileage;
        }
```

Iterator pattern for implementation of '**GetEnumerator()**' in the '**Car**' class uses the '**yield return**' statement to create an iterator. The '**yield return**' allows the method to return each mileage value one at a time, enabling the foreach loop iterate over them. By using the **yield return** statement, the **Car** class abstracts the complexity of iteration and provides a simplified way to access the mileage data.

Objectives:

- **Abstraction:** IEnumerable abstracts away the specific implementation details of a collection, allowing consumers to focus on the act of iteration rather than the underlying structure.
- **Standardization:** By following the IEnumerable interface, classes conform to a standard pattern, making their usage and interaction more consistent across different parts of an application.
- **Simplicity:** IEnumerable simplifies the process of working with collections by providing a single, unified interface for iteration, eliminating the need for custom iteration code in every consuming context.

Your review:

If you have something else to add in the topic, or something I missed please tell me here:

mailto:vansh4856.be22@chitkara.edu.in

What you need to learn for understanding this:

You need to have some understanding on the following topics:

Encapsulation: https://github.com/17012004/Encapsulation

Abstraction: https://github.com/17012004/Abstraction

I have provided my notes in those links with a code as an example;

Summarize what you learned:

We learn many important points from IEnumeration:

1. IEnumerable is an interface in C# that represents a sequence of elements.
2. It provides a standardized way to iterate over a collection of objects using foreach loops or other iteration constructs.
3. By implementing IEnumerable, a class declares its ability to be enumerated, allowing user to traverse its elements without exposing the underlying implementation details.
4. he key member of IEnumerable is the **GetEnumerator()** method, which returns an IEnumerator object responsible for iterating over the elements.
5. IEnumerable promotes encapsulation, code reusability, and simplifies working with collections by providing a unified interface for iteration.

Feedback:

If my notes helped you please give you feedbacks and if you have queries mail me on

mailto:vansh4856.be22@chitkara.edu.in

And add CC/BCC mail as:

mailto:harsimran.kaur@chitkara.edu.in