20. Ability to traverse a collection without knowing its underlying structure

You have a fairly complex data structure due to various constraints that you have to overcome. Perhaps, you need to arrange the data into a binary search tree to enable an efficient search algorithm. Or perhaps it's a linked list, as each item in the collection needs to know about the next item.

In any case, the client class that will be using your library would not care about what algorithm you used and how your data is structured. All it cares about is that it can work with the data in your collection.

Suitable design patterns

Iterator

Iterator is a design pattern that hides implementation details of a collection-manipulating algorithm and only exposes some basic methods that enable any external object to work with the collection, such as `MoveNext` and `HasMore`.

So, as far as your client is concerned, the collection you are dealing with consists of sequential flat data, just like an array would be. However, internally, the data structure can be anything but.

These days, you will rarely have to implement Iterator yourself, as core system libraries in all major programming languages already have a whole range of collection types that implement iterators. But there are still some scenarios where you need to write a special type of collection yourself. And this is where Iterator would be very useful.

Why would you use Iterator

- Abstracts away complex implementation details of collection management.
- Simplifies the view of the collection to the client.