Refresh:

* The iterator pattern is a great pattern for providing navigation without exposing the structure of an object.
* Traverse a container. In Java and most current programming languages, there’s the notion of a collection. List, Maps, Sets are all examples of a collection that we would want to traverse. Historically we use a loop of some sort and index into your collection to traverse it.
* Doesn’t expose the underlying structure of the object we want to navigate. Navigating various structures may have different algorithms or approaches to cycle through the data.
* Decouples the data from the algorithm used to traverse it
* It is an interface-based design pattern. Whichever object you want to iterate over will provide a method to return an instance of an iterator from it.
* Follows a factory-based method pattern in the way you get an instance of the iterator.
* Each iterator is developed in such a way that it is independent of another.
* Iterators also Fail Fast. Fail Fast means that iterators can’t modify the underlying object without an error being thrown.

Exercise:

Create a program that implements Iterable<String> interface, a Car.java class and create an array to iterate and print out the different models of the 3 cars that you add on the array.

Check the complete solution at:

[Design Patterns in Java - Iterator Pattern - GeeksforGeeks](https://www.geeksforgeeks.org/design-patterns-in-java-iterator-pattern/)