**forEach() method in Iterable interface**

Whenever we need to traverse through a Collection, we need to create an Iterator whose whole purpose is to iterate over, and then we have business logic in a loop for each of the elements in the Collection. We might get ConcurrentModificationException if the iterator is not used properly.

Graphical user interface, text, application

Description automatically generated

Java 8 has introduced forEach method in java.lang.Iterable interface so that while writing code we focus on business logic. The forEach method takes java.util.function.Consumer object as an argument, so it helps in having our business logic at a separate location that we can reuse. Let’s see forEach usage with a simple example.

Graphical user interface, text, application

Description automatically generated

Now we have two terminology that we have used above

* ConcurrentModificationException
* java.util.function.Consumer

You might not have familiar with it.

So here we will only discuss about **ConcurrentModificationException**

This exception occurs when we are iterating over a Collection using **Enhance for loop, Regular for loop** or **Iterator** and while iterating we are trying to modify ( removing, adding or setting elements ) our collection.

For example :

Text

Description automatically generated

Text

Description automatically generated

So according to java developers, if you really want to remove elements while iterating then you should iterate the collection using **Iterator** and use Iterator’s remove() method.

Below is the demonstration :

Text

Description automatically generated

In the next tutorial, we will talk about what is functional programming. Also, java.util.function.Consumer will be cover there.