Iterating an ArrayList

Below are the different methods to iterate over an ArrayList.

Using `for` loop

An ArrayList can be iterated easily using a simple `for` loop or an enhanced `for` loop as shown below.

```
import java.util.ArrayList;
import java.util.List;

public class ArrayListDemo {

    public static void main(String args[]) {
        List list = new ArrayList();
        list.add(10);
        list.add(20);
        list.add(30);
        list.add(40);

        for (int i = 0; i < list.size(); i++) { //Simple for loop
            System.out.println(list.get(i));
        }

        for (Integer i : list) {   //Enhanced for loop
            System.out.println(i);
        }
    }
}
```

Run Save Reset

Using Iterator

The `iterator()` method in ArrayList returns an Iterator type object. The Iterator interface declares the below methods that help with iterating an ArrayList.

1. `hasNext()` - This method returns `true` if there are more elements in the list; otherwise, it returns `false`.
2. `next()` - This method returns the next element in the list. Before calling `next()`, we should always call `hasNext()` to verify that there is an element; otherwise, `NoSuchElementException` will be thrown.
3. `remove()` - This method removes the last element returned by the iterator. It can be called only once per call to the `next()`.
4. `forEachRemaining(Consumer<? super E> action)` - This method was introduced in Java 8. It performs the given action for each remaining element until all elements have been processed or the action throws an exception. This method's benefit is that we do not need to check if there is a next element every time.

> To understand the working of the `forEachRemaining()` method, you should be familiar with basic concepts of functional programming that were introduced in Java 8. If you are interested then you can learn more about Java 8 here

Below is an example of iterating an ArrayList using the iterator.

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class ArrayListDemo {

    public static void main(String args[]) {
        List<Integer> list = new ArrayList<>();
        list.add(10);
        list.add(20);
        list.add(30);
        list.add(40);
        list.add(10);

        Iterator<Integer> itr = list.iterator();

        while(itr.hasNext()) {
            System.out.println(itr.next());
        }

        // Iterating using forEachRemaining() method
        System.out.println("Iterating using forEachRemaining() method");
        Iterator<Integer> newItr = list.iterator();
        newItr.forEachRemaining(element -> System.out.println(element));
    }
}
```

Run Save Reset

If we try to directly remove an element while iterating an ArrayList using an iterator is created, then `ConcurrentModificationException` will also be thrown. We should always use the `remove()` method in the iterator to remove an element from the ArrayList.

The below program will fail because we are trying to delete the element from the list directly.

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class ArrayListPractice {

    public static void main(String args[]) {
        List<Integer> list = new ArrayList<>();
        list.add(10);
        list.add(20);
        list.add(30);
        list.add(40);
        list.add(10);

        Iterator<Integer> itr = list.iterator();

        while (itr.hasNext()) {
            int next = itr.next();

            if (next == 30) {
                list.remove(new Integer(30));
            }
        }
    }
}
```

Run Save Reset

The code shown below is the correct way to delete an element from the list.

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class ArrayListDemo {

    public static void main(String args[]) {
        List<Integer> list = new ArrayList<>();
        list.add(10);
        list.add(20);
        list.add(30);
        list.add(40);
        list.add(10);

        Iterator<Integer> itr = list.iterator();

        while(itr.hasNext()) {
            int next = itr.next();
            if(next == 30) {
                itr.remove();
            }
        }
        System.out.println(list);
    }
}
```

Run Save Reset

If an element is added to the ArrayList after the iterator is created then also `ConcurrentModificationException` will be thrown.

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;


public class ArrayListDemo {

    public static void main(String args[]) {
        List<Integer> list = new ArrayList<>();
        list.add(34);
        list.add(45);

        Iterator<Integer> itr = list.iterator();
        list.add(54);

        while(itr.hasNext()) {
            System.out.println(itr.next());
        }
    }
}
```

Run Save Reset