Cursors

We have three types of Cursors in Java.

They are:

Enumeration

Iterator

List Iterator

Enumeration

1. It is used only in **Legacy Classes. Ex Vector and Stack**
2. Enumeration Cursor is **not a universal cursor.**
3. Enumeration Cursor is possible to **read the data** and it is **not possible to update the data**
4. We have **elements() Method** to get the Enumeration Object.
5. We can read the data only in forward direction

Iterator:

1. Iterator is a **Universal Cursor**.
2. Iterator is an **Interface** is it present in JDK 1.2 version
3. It is possible **to read and delete the data** by using Iterator.
4. We have **iterator method** to get the **Iterator Object.**
5. It is not supporting **insert and update** for the data
6. We can **read the data** only in forward direction

ListIterator

1. ListIterator can used for only list type of Objects.
2. ListIterator is an **Interface** is it present in JDK 1.2 version
3. It is possible **to read and update the data and delete the data**.
4. We have **listiterator method** to get the **ListItertaor Object.**
5. We can **read the data** in forward direction and backward direction

```
// Enumeration
Vector<String> v = new Vector<String>();
v.add("NameOne");
v.add("NameTwo");
v.add("NameThree");
v.add("NameFour");
System.out.println(v); // [NameOne, NameTwo, NameThree, NameFour]

Enumeration<String> elements = v.elements();
while (elements.hasMoreElements()) {
System.out.println(elements.nextElement());
}
```

```
NameOne
NameTwo
NameThree
NameFour
```

```java
// Iterator with List
List<String> list = new ArrayList<String>();
list.add("One");
list.add("Two");
list.add("Three");
list.add("Four");
list.add("Five");
System.out.println(list); //[One, Two, Three, Four, Five]

Iterator<String> iterator = list.iterator();
while (iterator.hasNext()) {
System.out.println(iterator.next());
```

```
One
Two
Three
Four
Five
```

```java
// Iterator with Set
Set<String> list = new HashSet<String>();
list.add("One");
list.add("Two");
list.add("Three");
list.add("Four");
list.add("Five");
System.out.println(list); //[Five, One, Four, Two, Three]

Iterator<String> iterator = list.iterator();
while (iterator.hasNext()) {
System.out.println(iterator.next());
}
```

```
Five
One
Four
Two
Three
```

```
//remove element using iterator
List<String> list = new ArrayList<String>();
list.add("One");
list.add("Two");
list.add("Three");
list.add("Four");
list.add("Five");
System.out.println(list); //[One, Two, Three, Four, Five]

Iterator<String> iterator = list.iterator();
while (iterator.hasNext()) {
String next = iterator.next();
if (next.equals("Two")) {
iterator.remove();
}
}
System.out.println(list); // [One, Three, Four, Five]
```

```
//ListIterator
List<String> al = new ArrayList<String>();
al.add("NameOne");
al.add("NameTwo");
al.add("NameThree");
al.add("NameFour");
System.out.println(al); // [NameOne, NameTwo, NameThree, NameFour]

ListIterator<String> listIterator = al.listIterator();
while (listIterator.hasNext()) {
System.out.println(listIterator.next() + "  ");
}
System.out.println();
while (listIterator.hasPrevious()) {
System.out.println(listIterator.previous() + " ");
}
```

Enumerration can be used for Legacy collections
Iterator can be used for collection implementations
ListIterator can be used for List Implementations

Enumeration can be used for only read operations
Iteration can be used to read and remove operations
ListIterator can be used for performing read, insert, remove, update

Enumeration can be used to read elements in forward direction
Iterator can be used to read elements in forward direction
ListIterator can be used to read elements in forward direction and backward direction