**Iterator**

2016-12-21

이승진

**학습 목표**

Iterator 객체 생성 방법과 사용 방법

ListIterator 객체 생성 방법과 사용 방법

컬렉션을 탐색 도중 삽입과 삭제 하기

**목차**

[1. Iterator 인터페이스 2](#_Toc470109615)

[1) 줄임 말 정리 2](#_Toc470109616)

[2) Iterator 인터페이스 메소드 2](#_Toc470109617)

[3) 이터레이터 객체 생성 2](#_Toc470109618)

[4) 예제1 - 탐색 3](#_Toc470109619)

[5) 예제2 - 탐색 4](#_Toc470109620)

[6) 예제3 - 탐색 중 제거 5](#_Toc470109621)

[7) 예제4 - 탐색 중 제거 6](#_Toc470109622)

[2. ListIterator 인터페이스 7](#_Toc470109623)

[1) ListIterator 인터페이스 메소드 7](#_Toc470109624)

[2) 리스트 이터레이터 객체 생성 7](#_Toc470109625)

[3) 예제5 - 역방향 탐색 8](#_Toc470109626)

[4) 예제6 - 역방향 탐색 9](#_Toc470109627)

[5) 예제7 - 탐색 중 삽입 10](#_Toc470109628)

[6) 예제8 - 탐색 중 삽입 11](#_Toc470109629)

# Iterator 인터페이스

## 줄임 말 정리

아래 설명할 때 사용할 줄임말을 정리하자.

Collection 인터페이스를 구현(implements)한 클래스의 객체 = 컬렉션 객체

Iterator 인터페이스를 구현(implements)한 클래스의 객체 = 이터레이터 객체

List 인터페이스를 구현(implements)한 클래스의 객체 = 리스트 객체

ListIterator 인터페이스를 구현(implements)한 클래스의 객체 = 리스트이터레이터 객체

## Iterator 인터페이스 메소드

Collection 강의 예제에서 컬렉션 객체를 탐색할 때, for 루프를 사용하여 구현했었다.

이터레이터 객체를 사용하여, 컬렉션 객체 탐색을 구현할 수도 있다.

이터레이터 객체는, 컬렉션 객체를 탐색하기 위한 객체이다.

즉 Iterator 인터페이스에는, 컬렉션 객체를 탐색하기 위한 메소드가 포함되어 있다.

이터레이터에는 **현재 위치**라는 개념이 있다.

이터레이터에는 컬렉션의 데이터 항목을 하나 가르키는데,

이터레이터의 **현재 위치**는 이터레이터가 현재 가르키고 있는 데이터 항목을 의미한다.

|  |
| --- |
| **boolean hasNext()**  컬렉션에 아직 탐색할 데이터가 남아 있으면 true를 리턴한다.  모든 데이터의 탐색을 완료하고 탐색이 끝났으면 false를 리턴한다. |
| **E next()**  다음 데이터 항목으로 현재 위치를 이동한 후, 그 데이터 항목을 리턴한다.  이미 현재 위치가 목록의 끝이라서, 다음 데이터 항목이 없다면 예외(NoSuchElementException)가 발생한다. |
| **void remove()**  컬렉션에서 현재 위치의 데이터를 제거한다.  직전의 next() 메소드 호출에서 리턴된 데이터가 제거된다. |

## 이터레이터 객체 생성

Collection 인터페이스의 iterator() 메소드를 호출하면,

현재 컬렉션 객체를 탐색하기 위한 이터레이터 객체가 생성되어 리턴된다.

Collection 인터페이스를 구현(implements)한 클래스들은 모두

iterator 메소드를 구현했다.

Collection 인터페이스 메소드

|  |
| --- |
| **Iterator iterator()**  현재 컬렉션 객체를 탐색하기 위한 이터레이터 객체가 생성되어 리턴된다.  리턴된 이터레이터 객체의 현재 위치는, 아직 아무 것도 가르키고 있지 않다.  이터레이터 객체를 생성한 후, next() 메소드를 호출하면 목록에서 첫 데이터 항목을 가르킨다. |

## 예제1 - 탐색

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | package net.skhu.iterator;  import java.util.ArrayList;  import java.util.Collection;  import java.util.Iterator;  public class Example1 {    public static void main(String[] args) {  Collection<Integer> c = new ArrayList<Integer>();  for (int i = 0; i < 10; ++i)  c.add(i);    for (Integer i : c)  System.out.print(i + " ");    System.out.println();  Iterator<Integer> iterator = c.iterator();  while (iterator.hasNext()) {  Integer i = iterator.next();  System.out.print(i + " ");  }  }  } |

실행 결과 출력

|  |
| --- |
| 0 1 2 3 4 5 6 7 8 9  0 1 2 3 4 5 6 7 8 9 |

(줄10)

Collection<Integer> c = new ArrayList<Integer>();

Integger 객체를 보관하기 위한 컬렉션 객체를 생성한다.

정확히 표현하면, Integer 객체에 대한 참조를 보관하기 위한 컬렉션 객체를 생성한다.

컬렉션 객체에 기본 자료형 int를 보관할 수 없고, 객체에 대한 참조만 보관할 수 있다.

(줄15~16) c 컬렉션 객체 탐색

(줄20~24) c 컬렉션 객체 탐색

실행 결과 출력

|  |
| --- |
| 0 1 2 3 4 5 6 7 8 9  0 1 2 3 4 5 6 7 8 9 |

## 예제2 - 탐색

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | package net.skhu.iterator;  import java.util.ArrayList;  import java.util.Collection;  import java.util.Iterator;  public class Example2 {    public static void main(String[] args) {  Collection<String> c = new ArrayList<String>();  for (int i = 0; i < 10; ++i) {  String s = String.format("%02d ", i);  c.add(s);  }    for (String s : c)  System.out.print(s);    System.out.println();  Iterator<String> iterator = c.iterator();  while (iterator.hasNext()) {  String s = iterator.next();  System.out.print(s);  }  }  } |

실행 결과 출력

|  |
| --- |
| 00 01 02 03 04 05 06 07 08 09  00 01 02 03 04 05 06 07 08 09 |

(줄10)

Collection<String> c = new ArrayList<String>();

String 객체를 보관하기 위한 컬렉션 객체를 생성한다.

정확히 표현하면, String 객체에 대한 참조를 보관하기 위한 컬렉션 객체를 생성한다.

실행 결과 출력

|  |
| --- |
| 00 01 02 03 04 05 06 07 08 09  00 01 02 03 04 05 06 07 08 09 |

## 예제3 - 탐색 중 제거

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | package net.skhu.iterator;  import java.util.ArrayList;  import java.util.Collection;  import java.util.Iterator;  public class Example3 {    public static void main(String[] args) {  Collection<Integer> c = new ArrayList<Integer>();  for (int i = 0; i < 10; ++i)  c.add(i);    for (int i : c)  System.out.print(i + " ");    System.out.println();  Iterator<Integer> iterator = c.iterator();  while (iterator.hasNext()) {  int i = iterator.next();  if (i % 2 == 0) iterator.remove();  }    for (int i : c)  System.out.print(i + " ");  }  } |

실행 결과 출력

|  |
| --- |
| 0 1 2 3 4 5 6 7 8 9  1 3 5 7 9 |

(줄23)

if (i % 2 == 0) iterator.remove();

c 컬렉션 객체에서 짝수인 값들이 제거된다.

위 예제 코드에서,

c 컬렉션 객체에 int 값을 add 할 때, 자동으로 Integer 객체로 변환되어 add 된다. (auto-boxing)

c 컬렉션 객체에서 int 값을 꺼낸ㄹ 때, 자동으로 Integer 객체가 int로 변환되어 리턴된다. (auto-unboxing)

실행 결과 출력

|  |
| --- |
| 0 1 2 3 4 5 6 7 8 9  1 3 5 7 9 |

## 예제4 - 탐색 중 제거

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | package net.skhu.iterator;  import java.util.ArrayList;  import java.util.Collection;  public class Example4 {    public static void main(String[] args) {  Collection<Integer> c = new ArrayList<Integer>();  for (int i = 0; i < 10; ++i)  c.add(i);    for (int i : c)  System.out.print(i + " ");    System.out.println();  for (int i : c)  if (i % 2 == 0) c.remove(i);  for (int i : c)  System.out.print(i + " ");  }  } |

(줄19~20)

c 컬렉션 객체에서 짝수인 값들을 제거하는 코드를 위와 같이 구현할 수는 없다.

컬렉션 객체를 for 문으로 탐색하는 도중에, 그 컬렉션 객체가 수정되는 메소드(add, remove, addAll, clear 등)가 호출되면 에러가 발생한다.

컬렉션 객체를 탐색하는 도중에, 이터레이터 객체의 remove 메소드를 호출하는 것은 허용된다.

실행 결과 출력

|  |
| --- |
| 0 1 2 3 4 5 6 7 8 9  Exception in thread "main" java.util.ConcurrentModificationException  at java.util.ArrayList$Itr.checkForComodification(Unknown Source)  at java.util.ArrayList$Itr.next(Unknown Source)  at net.skhu.iterator.Example4.main(Example4.java:19) |

# ListIterator 인터페이스

## ListIterator 인터페이스 메소드

|  |
| --- |
| **void add(E e)**  현재 위치에 e를 끼워 넣는다.  직전의 next() 메소드 호출로 리턴된 데이터 바로 뒤이거나,  직전의 previous() 메소드 호출에서 리턴된 데이터의 바로 앞에 e가 삽입된다. |
| **boolean hashNext()**  순방향으로 컬렉션을 탐색 중일 때, 컬렉션에 아직 탐색할 데이터가 남아 있으면 true를 리턴한다.  즉 현재 위치의 뒤에 데이터 항목이 있으면 true를 리턴하고 없으면 false를 리턴한다. |
| **boolean hashPrevious()**  역방향으로 컬렉션을 탐색 중일 때, 컬렉션에 아직 탐색할 데이터가 남아 있으면 true를 리턴한다.  즉 현재 위치의 앞에 데이터 항목이 있으면 true를 리턴하고 없으면 false를 리턴한다. |
| **E next()**  다음 데이터 항목으로 현재 위치를 이동한 후, 그 데이터 항목을 리턴한다.  이미 현재 위치가 목록의 끝이라서, 다음 데이터 항목이 없다면 예외(NoSuchElementException)가 발생한다. |
| **int nextIndex()**  다음 데이터 항목으로 현재 위치를 이동한 후, 그 데이터 항목의 인덱스를 리턴한다.  이미 현재 위치가 목록의 끝이라서, 다음 데이터 항목이 없다면 예외(NoSuchElementException)가 발생한다. |
| **E previous()**  이전 데이터 항목으로 현재 위치를 이동한 후, 그 데이터 항목을 리턴한다.  현재 위치가 목록의 선두라서, 이전 데이터 항목이 없다면 예외(NoSuchElementException)가 발생한다. |
| **int previousIndex()**  이전 데이터 항목으로 현재 위치를 이동한 후, 그 데이터 항목의 인덱스를 리턴한다.  이미 현재 위치가 목록의 선두라서, 이전 데이터 항목이 없다면 예외(NoSuchElementException)가 발생한다. |
| **void remove()**  컬렉션에서 현재 위치의 데이터를 제거한다.  직전의 next() 메소드 호출이나 previous() 메소드 호출에서 리턴된 데이터가 제거된다. |
| **void set(E e)**  현재 위치에 데이터 e를 덥어쓴다.  직전의 next() 메소드 호출이나 previous() 메소드 호출에서 리턴된 데이터가 덥어써진다. |

## 리스트 이터레이터 객체 생성

List 인터페이스의 listIterator() 메소드를 호출하면,

현재 컬렉션 객체를 탐색하기 위한 이터레이터 객체가 생성되어 리턴된다.

List 인터페이스를 구현(implements)한 클래스들은 모두

listIterator 메소드를 구현했다.

List 인터페이스의 메소드

|  |
| --- |
| **ListIterator listIterator()**  현재 리스트 객체를 탐색하기 위한 리스트이터레이터 객체가 생성되어 리턴된다.  리턴된 리스트이터레이터 객체의 현재 위치는, 아직 아무 것도 가르키고 있지 않다.  리스트이터레이터 객체를 생성한 후, next() 메소드를 호출하면 목록에서 첫 데이터 항목을 가르킨다. |
| **ListIterator listIterator(int index)**  현재 리스트 객체를 탐색하기 위한 리스트이터레이터 객체가 생성되어 리턴된다.  리턴된 리스트이터레이터 객체의 현재 위치는, **index** 위치의 데이터 항목을 가르킨다. |

## 예제5 - 역방향 탐색

Example5.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | package net.skhu.iterator;  import java.util.ArrayList;  import java.util.List;  import java.util.ListIterator;  public class Example5 {    public static void main(String[] args) {  List<Integer> list = new ArrayList<Integer>();  for (int i = 0; i < 10; ++i)  list.add(i);    for (int i = list.size() - 1; i >= 0; --i)  System.out.print(list.get(i) + " ");    System.out.println();  3  ListIterator<Integer> iterator = list.listIterator(list.size());  while (iterator.hasPrevious()) {  Integer i = iterator.previous();  System.out.print(i + " ");  }  }  } |

실행 결과 출력

|  |
| --- |
| 9 8 7 6 5 4 3 2 1 0  9 8 7 6 5 4 3 2 1 0 |

## 예제6 - 역방향 탐색

Example6.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | package net.skhu.iterator;  import java.util.ArrayList;  import java.util.List;  import java.util.ListIterator;  public class Example6 {    public static void main(String[] args) {  List<String> list = new ArrayList<String>();  for (int i = 0; i < 10; ++i) {  String s = String.format("%02d", i);  list.add(s);  }    for (int i = list.size() - 1; i >= 0; --i)  System.out.print(list.get(i) + " ");    System.out.println();  ListIterator<String> iterator = list.listIterator(list.size());  while (iterator.hasPrevious()) {  String s = iterator.previous();  System.out.print(s + " ");  }  }  } |

실행 결과 출력

|  |
| --- |
| 09 08 07 06 05 04 03 02 01 00  09 08 07 06 05 04 03 02 01 00 |

## 예제7 - 탐색 중 삽입

Example7.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41 | package net.skhu.iterator;  import java.util.ArrayList;  import java.util.List;  import java.util.ListIterator;  public class Example7 {    public static void main(String[] args) {  List<Integer> list = new ArrayList<Integer>();  for (int i = 0; i < 10; ++i)  list.add(i);    for (int i : list)  System.out.print(i + " ");    System.out.println();  ListIterator<Integer> iterator = list.listIterator();  while (iterator.hasNext()) {  int i = iterator.next();  if (i % 2 == 0) iterator.remove();  }    for (int i : list)  System.out.print(i + " ");  System.out.println();    iterator = list.listIterator();  while (iterator.hasNext()) {  int i = iterator.next();  if (i % 2 == 1) iterator.add(i + 1);  }    for (int i : list)  System.out.print(i + " ");  }  } |

실행 결과 출력

|  |
| --- |
| 0 1 2 3 4 5 6 7 8 9  1 3 5 7 9  1 2 3 4 5 6 7 8 9 10 |

## 예제8 - 탐색 중 삽입

Example8.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41 | package net.skhu.iterator;  import java.util.ArrayList;  import java.util.List;  import java.util.ListIterator;  public class Example8 {    public static void main(String[] args) {  List<Integer> list = new ArrayList<Integer>();  for (int i = 0; i < 10; ++i)  list.add(i);    for (int i : list)  System.out.print(i + " ");    System.out.println();  ListIterator<Integer> iterator = list.listIterator();  while (iterator.hasNext()) {  int i = iterator.next();  if (i % 2 == 0) iterator.remove();  }    for (int i : list)  System.out.print(i + " ");  System.out.println();    iterator = list.listIterator(list.size());  while (iterator.hasPrevious()) {  int i = iterator.previous();  if (i % 2 == 1) iterator.add(i - 1);  }    for (int i : list)  System.out.print(i + " ");  }  } |

실행 결과 출력

|  |
| --- |
| 0 1 2 3 4 5 6 7 8 9  1 3 5 7 9  0 1 2 3 4 5 6 7 8 9 |