

13-2. LIFO와 FIFO 컬렉션

혼자 공부하는 자바 (신용권 저)

- 시작하기 전에
- Stack
- Queue
- 키워드로 끝내는 핵심 포인트
- 확인문제

시작하기 전에

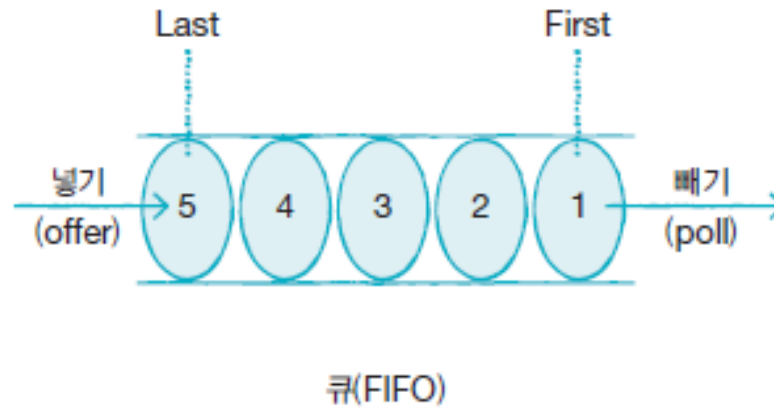
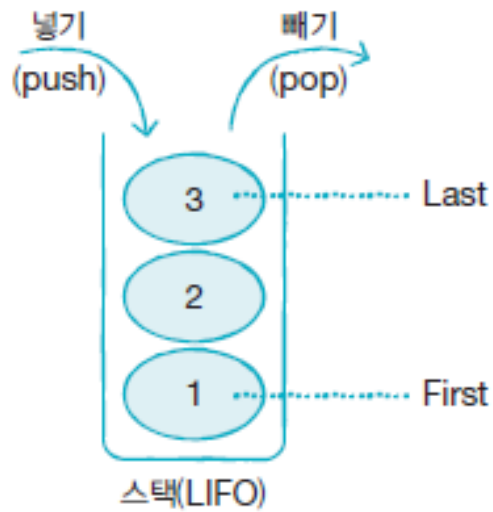
[핵심 키워드] : Stack, Queue

[핵심 포인트]

컬렉션 프레임워크에는 LIFO(후입선출) 자료구조를 제공하는 Stack 클래스와 FIFO(선입선출) 자료구조를 제공하는 Queue 인터페이스가 있습니다. 이번 절에서는 Stack 클래스와 Queue 인터페이스에 대해 살펴본다.

시작하기 전에

- ❖ **후입선출** (LIFO : Last In First Out)
 - 나중에 넣은 객체가 먼저 빠져나가는 자료구조
- ❖ **선입선출** (FIFO : First In First Out)
 - 먼저 넣은 객체가 먼저 빠져나가는 자료구조
- ❖ 컬렉션 프레임워크에는 LIFO 자료구조 제공하는 Stack 클래스와 FIFO 자료구조 제공하는 Queue 인터페이스 제공됨



❖ Stack

- LIFO 자료구조 구현한 클래스

리턴 타입	메소드	설명
E	push(E item)	주어진 객체를 스택에 넣습니다.
E	peek()	스택의 맨 위 객체를 가져옵니다. 객체를 스택에서 제거하지 않습니다.
E	pop()	스택의 맨 위 객체를 가져옵니다. 객체를 스택에서 제거합니다.

- Stack 객체 생성하려면 저장할 객체 타입을 E 타입 파라미터 자리에 표기하고 기본 생성자를 호출

```
Stack<E> stack = new Stack<E>();  
Stack<E> stack = new Stack<>();
```

Stack의 E 타입 파라미터를 생략하면
← 왼쪽 Stack에 지정된 타입을 따라 감

■ 예시 - 동전 클래스

```
01 package sec02.exam01;
02
03 public class Coin {
04     private int value;
05
06     public Coin(int value) {
07         this.value = value;
08     }
09
10     public int getValue() {
11         return value;
12     }
13 }
```



■ 예시 - Stack을 이용한 동전 케이스

```
01 package sec02.exam01;
02
03 import java.util.*;
04
05 public class StackExample {
06     public static void main(String[] args) {
07         Stack<Coin> coinBox = new Stack<Coin>();
08
09         coinBox.push(new Coin(100));
10         coinBox.push(new Coin(50));
11         coinBox.push(new Coin(500));
12         coinBox.push(new Coin(10));
13
14         while(!coinBox.isEmpty()) {
15             Coin coin = coinBox.pop();
16             System.out.println("꺼내온 동전 : " + coin.getValue() + "원");
17         }
18     }
19 }
```

← 동전을 끼움

← 동전 케이스가 비었는지 확인

← 동전 케이스에서 제일 위의 동전을 꺼냄

실행결과	
꺼내온 동전 :	10원
꺼내온 동전 :	500원
꺼내온 동전 :	50원
꺼내온 동전 :	100원

❖ Queue

- FIFO 자료구조에서 사용되는 메소드 정의

리턴 타입	메소드	설명
boolean	offer(E e)	주어진 객체를 넣습니다.
E	peek()	객체 하나를 가져옵니다. 객체를 큐에서 제거하지 않습니다.
E	poll()	객체 하나를 가져옵니다. 객체를 큐에서 제거합니다.

- LinkedList 클래스

```
Queue<E> queue = new LinkedList<E>();
```

```
Queue<E> queue = new LinkedList<>();
```

LinkedList의 E 타입 파라미터를 생략하면
왼쪽 Queue에 지정된 타입을 따라 감

■ 예시 – Message 클래스

```
01  package sec02.exam02;
02
03  public class Message {
04      public String command;
05      public String to;
06
07      public Message(String command, String to) {
08          this.command = command;
09          this.to = to;
10      }
11  }
```

Queue

■ 예시 – Queue를 이용한 메시지 큐

```
01 package sec02.exam02;
02
03 import java.util.LinkedList;
04 import java.util.Queue;
05
06 public class QueueExample {
07     public static void main(String[] args) {
08         Queue<Message> messageQueue = new LinkedList<Message>();
09
10         messageQueue.offer(new Message("sendMail", "홍길동"));
11         messageQueue.offer(new Message("sendSMS", "신용권"));
12         messageQueue.offer(new Message("sendKakaotalk", "홍두깨"));
13
14         while(!messageQueue.isEmpty()) {
15             Message message = messageQueue.poll();
16             switch(message.command) {
17                 case "sendMail":
```


← 메시지 저장

← 메시지 큐가 비었는지 확인

← 메시지 큐에서 1개의 메시지 꺼냄

Queue

```
18         System.out.println(message.to + "님에게 메일을 보냅니다.");
19         break;
20     case "sendSMS":
21         System.out.println(message.to + "님에게 SMS를 보냅니다.");
22         break;
23     case "sendKakaotalk":
24         System.out.println(message.to + "님에게 카카오톡을 보냅니다.");
25         break;
26     }
27 }
28 }
29 }
```

 실행결과 ×

홍길동님에게 메일을 보냅니다.
신용권님에게 SMS를 보냅니다.
홍두께님에게 카카오톡을 보냅니다.

키워드로 끝내는 핵심 포인트

- **Stack** : 후입선출을 구현한 클래스.
- **Queue** : 선입선출에 필요한 메소드를 정의한 인터페이스. 구현 클래스로 LinkedList가 있음

확인문제

- ❖ Stack과 Queue에 대한 설명으로 맞는 것에 O, 틀린 것에 X하세요
- Stack은 후입선출을 구현한 클래스이다 ()
 - Queue는 선입선출을 위한 인터페이스이다 ()
 - Stack의 push()는 객체를 넣을 때, pop()은 객체를 뺄 때 사용한다 ()
 - Queue의 poll()은 객체를 넣을 때, offer()은 객체를 뺄 때 사용한다 ()

Thank You !

혼자 공부하는 자바 (신용권 저)