Data Structure

Fall 2019

M 16:00-18:00 W 11:00-13:00

http://smart.hallym.ac.kr

Instructor: **Jin Kim**
010-6267-8189(033-248-2318)
jinkim@hallym.ac.kr

Office Hours:

Lab(Priority Queue)

Fall 2019

http://smart.hallym.ac.kr

Instructor: Jin Kim

010-6267-8189(033-248-2318)

jinkim@hallym.ac.kr

Office Hours:

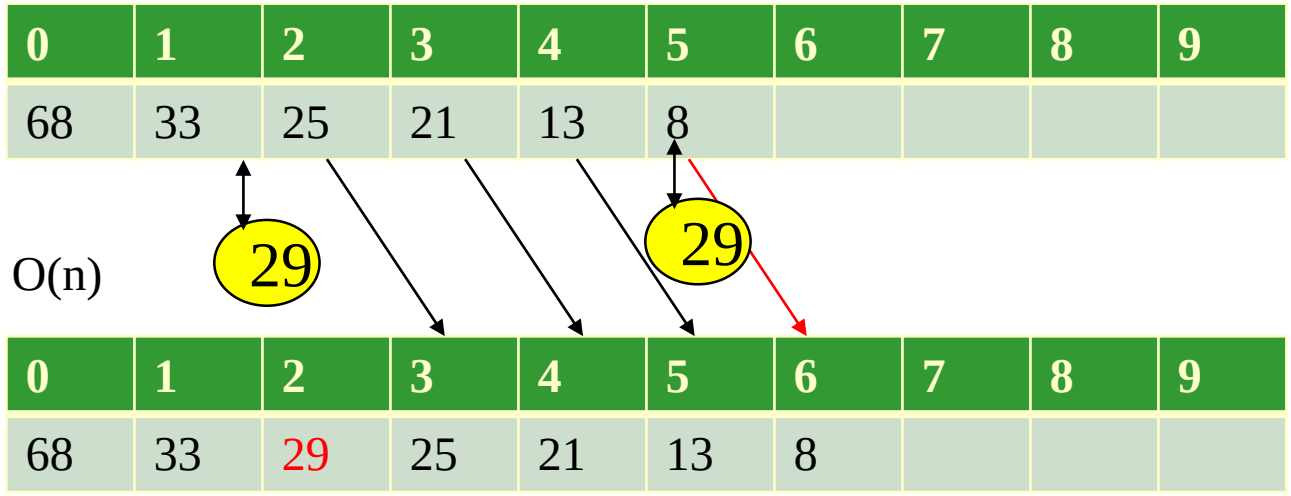
Priority Queue Implementation

- 배열과 연결리스트로 구현해보자
- 아주 다양한 구현 방법이 있다
- 우선순위큐안에 원소가 정렬되어 있을 수도 있고, 없을 수도 있지만 우리는 정렬되도록 할 것임.
- 우선순위큐에서 큐에 원소가 없을때까지 원소삭제(remove)하여 늘어놓으면 정렬이 됨.

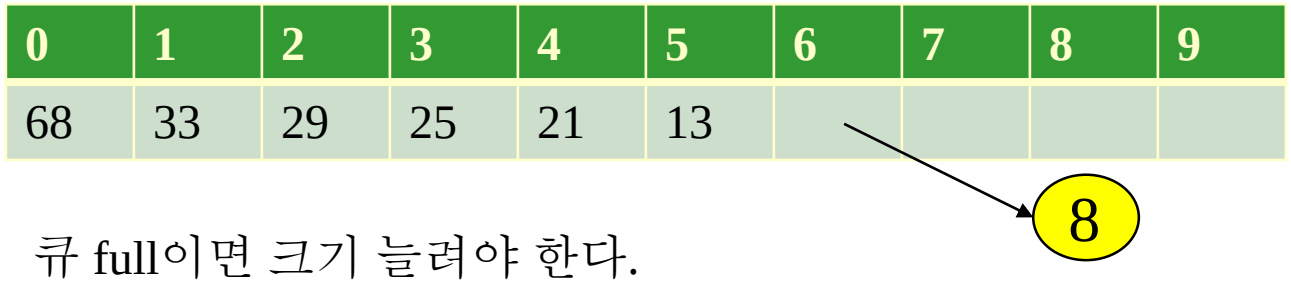
실제 구현 (배열 사용)

초기상태(작은 수가 가장 큰 우선순위를 가진다 가정)

삽입 (29) O(n)



삭제 O(1)



큐 full이면 크기 늘려야 한다.

ArrayPQ.java

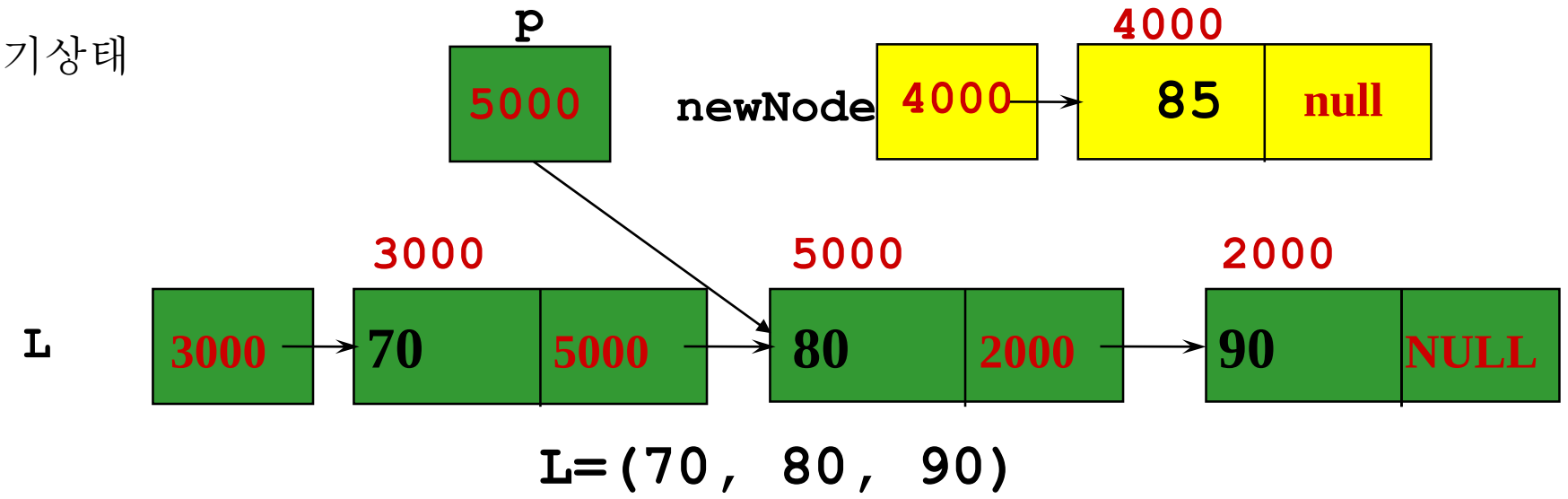
- 배열에서 원소 삽입시 pq full이면 원소크기를 +5하도록 코드를 추가하라.

  예
  priorityQ에 public void resize() 메소드추가하고 insert()에서
  if (isFull()) { resize(); }
  하면 어떨까?

- Void clear() : priority queue안의 모든 내용을 제거하는 메소드를 추가하라.
- Int size() : 현재 priority queue안의 원소 개수를 return하는 메소드를 추가하라.

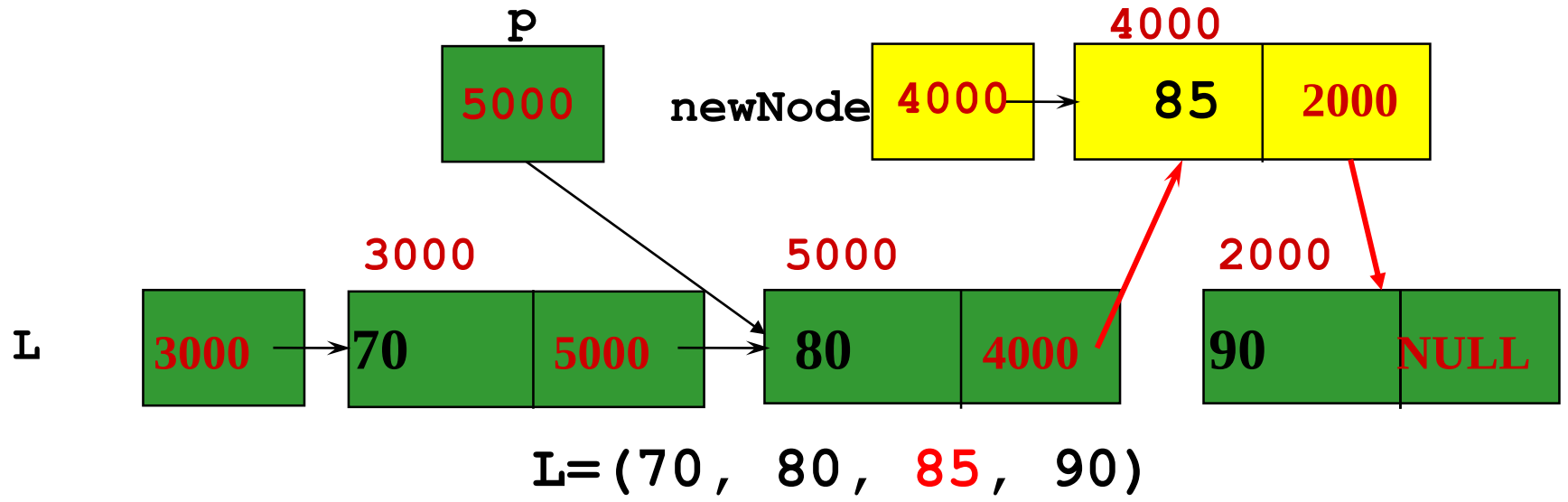
실제 구현 (연결리스트 사용)

초기상태



L=(70, 80, 90)

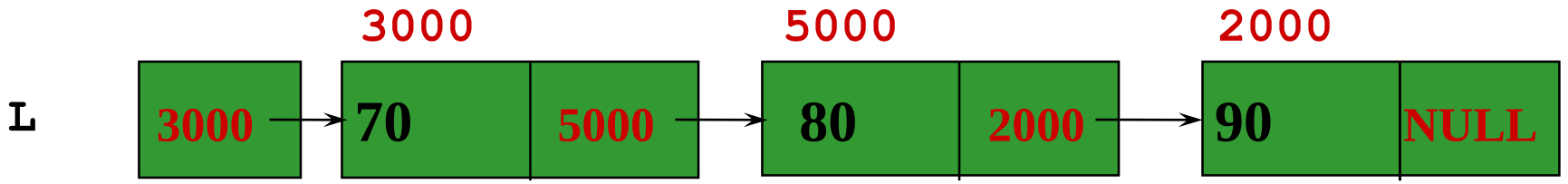
삽입 (75) O(n)



L=(70, 80, 85, 90)

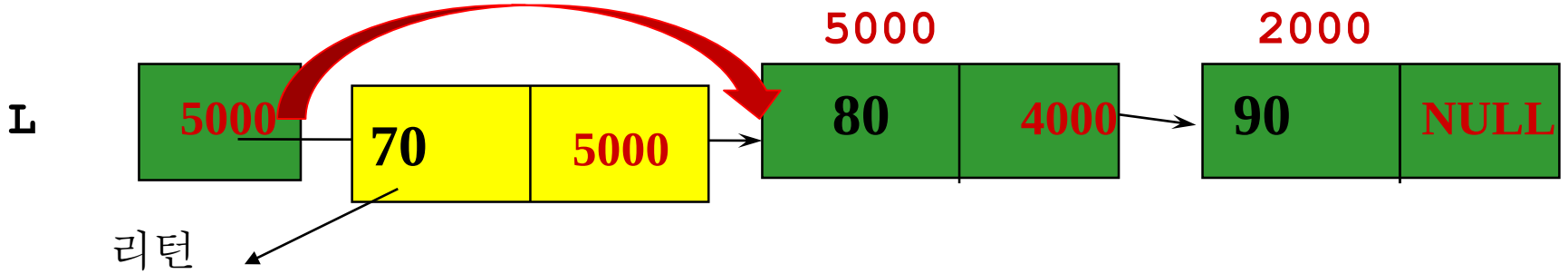
실제 구현 (연결리스트 사용)

초기상태



L=(70, 80, 90)

삭제 O(1)



L=(80, 90)

listPQ.java

- NumberofItem(): 현재 priority queue안의 원소 개수를 return하는 메소드를 추가하라.
- Peek() 채워라.
- Void clear() : priority queue안의 모든 내용을 제거하는 메소드를 추가하라.

  이때 원소의 개수를 0으로 하라.

PQCompObj.java

- Java.util.PriorityQueue를 사용하라.
- MyClass (학번, “이름“, 점수)의 객체 10개를 우선순위큐에 삽입하라.
- 우선순위큐에서 원소들을 제거하라. 이때 학번이 먼저 출력하도록하라.
- 점수가 먼저 출력되도록 해보라.

3개 프로그램 업로드

1. ArrayPQMain.java
2. ListPQMain.java
3. PQObject.java
4. Upload your program

*.java 파일만을 업로드하라.
다른 파일들은 필요없다.
불필요한 파일을 업로드하면 제출하지 않은 것으로 간주.

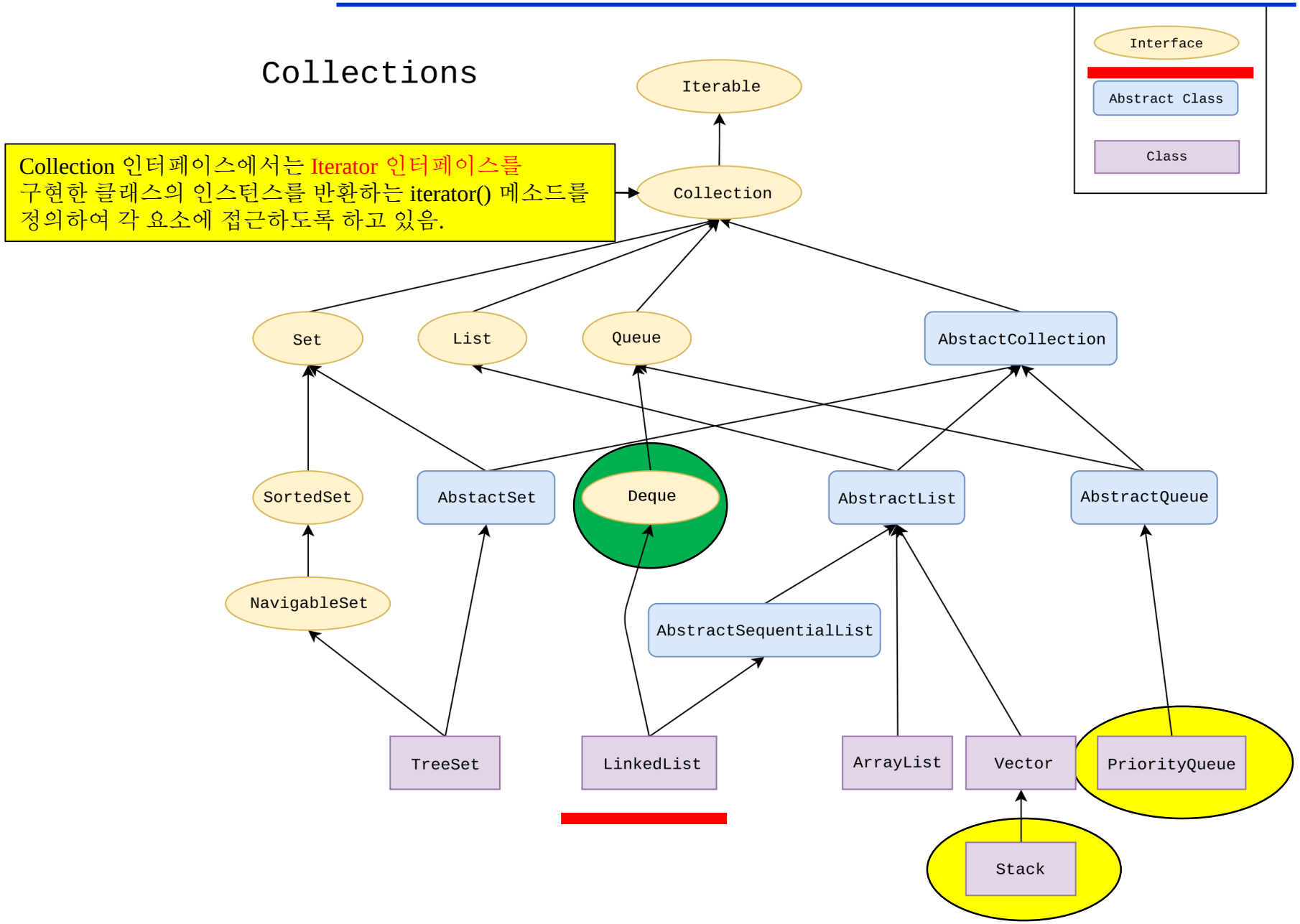
Java.util.PriorityQueue(PQUtil.java)

- Java.util.PriorityQueue를 사용하라.
- (“Kim”, 10, 99), (“Lee”, 11, 92), 등으로 (이름, 학번, 성적)순으로 원소를 입력하라. 학번순으로 출력되도록 하라. 성적순으로 출력되도록 하라. 이름순으로 출력되도록 하라.
- 기타 여러 함수들을 테스트해보라.

Java Priority Queue

- https://docs.oracle.com/javase/7/docs/api/java/util/PriorityQueue.html
- java.util
  - **Class PriorityQueue<E> // PriorityQueue는 class. 따라서 직접 빵을 찍을 수 있다.**

Java collections frameworks



Priority Queue in Java

Constructors of Priority Queue:

PriorityQueue(): Creates a PriorityQueue with the default initial capacity (11) that orders its elements according to their natural ordering.
PriorityQueue(Collection<E> c): Creates a PriorityQueue containing the elements in the specified collection.
PriorityQueue(int initialCapacity): Creates a PriorityQueue with the specified initial capacity that orders its elements according to their natural ordering.
PriorityQueue(int initialCapacity, Comparator<E> comparator): Creates a PriorityQueue with the specified initial capacity that orders its elements according to the specified comparator.
PriorityQueue(PriorityQueue<E> c): Creates a PriorityQueue containing the elements in the specified priority queue.
PriorityQueue(SortedSet<E> c): Creates a PriorityQueue containing the elements in the specified sorted set.

Methods in Priority Queue Class

1. boolean add(E element): This method inserts the specified element into this priority queue.
2. public remove(): This method removes a single instance of the specified element from this queue, if it is present
3. public poll(): This method retrieves and removes the head of this queue, or returns null if this queue is empty.
4. public peek(): This method retrieves, but does not remove, the head of this queue, or returns null if this queue is empty.
5. Iterator iterator(): Returns an iterator over the elements in this queue.
6. boolean contains(Object o): This method returns true if this queue contains the specified element
7. void clear(): This method is used to remove all of the contents of the priority queue.
8. boolean offer(E e): This method is used to insert a specific element into the priority queue.
9. int size(): The method is used to return the number of elements present in the set.
10. toArray(): This method is used to return an array containing all of the elements in this queue.
11. Comparator comparator(): The method is used to return the comparator that can be used to order the elements of the queue.

java.util.PriorityQueue 사용하기

Generic : 클래스 내부에서 사용할 데이터 타입을
나중에 인스턴스를 생성할 때 확정하는 것을 제네릭이라 한다

객체 사용

```
import java.util.PriorityQueue;

Public class MyPQ {
                …
        PriorityQueue<String> pQueue = new PriorityQueue<String>();
        pQueue.add("C");
        pQueue.add("C++");
        pQueue.add("Java");
        pQueue.add("Python");
             …
        pQueue.poll();
```

java.util.PriorityQueue 사용하기 (PQObj.java)

Generic : 클래스 내부에서 사용할 데이터 타입을
나중에 인스턴스를 생성할 때 확정하는 것을 제네릭이라 한다

객체 사용

```
import java.util.PriorityQueue;

class MyNode {
    int i;
    String name;
}

Public class MyDeque {
…
PriorityQueue<MyNode> pq = new PriorityQueue<>();

MyNode node1 = new MyNode();
pq.add(node1);
….
MyNode node2 = Pq.poll();
```

PQ에서 큰 수 먼저 출력하는 방법(PQascending.java)

PQ는 정수를 우선 순위 큐에 넣고 출력할 경우, 작은 수부터 출력한다. 이를 반대로 출력하도록 하는 몇가지 방법이 있다. Pqascending.java를 보고 그 방법을 이해하라.

PQ에서 객체를 비교하기(PQCompObj.java)

PQ에 객체를 입력하면, 객체들을 비교하기 위한 방법이 있어야 한다. 우선순위를 제공하는 방법을
. PQCompObj.java를 보고 이해하라.

참고 Iterator (PriorityQueue1.java)

Iterator는 자바의 컬렉션 프레임워크에서 컬렉션에 저장되어 있는 요소들을 읽어오는 방법을 표준화 하였는데 그 중 하나가 Iterator. 인터페이스이다. 모든 컬렉션클래스에서 데이터를 읽을 때 사용됨.

```
public interface Iterator {
        boolean hasNext();  // 원소가 남아있는냐?
        Object next();  //다음 원소를 가져와라.
        void remove(); // next 메소드가 호출한 원소를 제거하라.
}
```

사용법

변수 — Iterator 타입의 변수를 Iterator로 변환한다.

```
Iterator it1 = pq.iterator();
While(it1.hasNext()){
     ….
}
```