CECS 277 – Lecture 12 – Priority Queues

Priority Queues are heaps that sort elements based on a pre-specified ordering. Elements with higher priority are kept at the top, and those with lower priority are moved to the bottom. A priority queue can only sort objects that implement the Comparable interface and have overridden the compareTo() method. Some standard classes, such as String, Integer, and Double, already override the compareTo() method, but if you need to add your own objects to a PriortiyQueue, your class will need to implement the Comparable interface.

Priority Queues have several methods used to manipulate the queue:

1. offer(e) – inserts the element e into the queue.
2. peek() – accesses but doesn't remove the next element in the queue.
3. poll() – accesses and removes the next element in the queue.
4. size() – returns the number of elements in the queue.
5. clear() – removes all of the elements in the queue.

Example: Priority Queue of Strings

```
PriortyQueue<String> q = new PriortyQueue<String>();
q.offer("banana");
q.offer("tomato");
q.offer("zucchini");
q.offer("apple");

System.out.println(q.poll());  //outputs apple
```

The compareTo() method compares the specified comparable part of the two elements and returns a negative value if the object has higher priority than the specified object, a zero if they have equal priority, and a positive value otherwise. Strings prioritize in alphabetical order, and Integers prioritize on lower values.

```
String a = "cat";
String b = "dog";
System.out.println(a.compareTo(b)); //outputs -1


Integer a = new Integer (2);
Integer b = new Integer (1);
System.out.println(a.compareTo(b)); //outputs 1
```

```java
public class Student implements Comparable<Student>{
	private String name;
	private int grade;
	public Student(String n, int g){
		name = n;
		grade = g;
	}
	public String getName(){ return name; }
	public int getGrade(){ return grade; }
	public void setName(String n){ name = n; }
	public void setGrade(int g){ grade = g; }
	@Override
	public int compareTo(Student s) {
		if(this.getGrade()<s.getGrade()){
			return 1;
		}else if(this.getGrade()>s.getGrade()){
			return -1;
		}else{
			return 0;
		}
	}
}
import java.util.PriorityQueue;
public class TestPQ {
	public static void main(String[] args) {
		PriorityQueue<Student> q=new PriorityQueue<Student>();
		q.offer(new Student("Mary", 77));
		q.offer(new Student("George", 45));
		q.offer(new Student("Ethel", 89));
		q.offer(new Student("John", 88));
		q.offer(new Student("Tom", 90));
		while (!q.isEmpty()){
			System.out.println(q.poll().getName());
		}
	}
}
/*Output
Tom
Ethel
John
Mary
George
*/
```