자료구조 및 실습
보고서

[제06주] DS_06_201502273_김현종

2018.04.21.

201502273/김현종

1.내용

```
public class CircularArrayQueue {
        private static final int DEFAULT_MAX_SIZE = 10;//기본 최대 크기
    private int maxSize;//Queue의 최대 크기
    private int front;//Queue의 front
    private int rear;//Queue의 rear
    private String[] elements;

    /*
     * CircularArrayQueue의 기본 Constructor
     * this(maxSize)를 호출한다.
     */
    CircularArrayQueue() {
        this(CircularArrayQueue.DEFAULT_MAX_SIZE);
    }

    /*
     * CircularArrayQueue의 Constructor
     * 받은 값으로 Queue의 최대 크기를 지정하고 front, rear를 0으로 초기화 하고
     * maxSize의 크기로 String Array의 객체를 만든다.
     */
    public CircularArrayQueue(int maxSize) {
        this.maxSize = maxSize;
        this.front = 0;
        this.rear = 0;
        this.elements = new String[this.maxSize];
        }

    /*
     * Queue에 넣는 함수
     * Queue가 꽉 차 있는지 확인하고
     * 꽉 차있으면 return false
     * 그렇지 않으면 rear가 가리키는 부분에 받은 값을 넣어주고 rear를 1 올린뒤
     * maxSize로 Mod연산 해준다.
     */
        public boolean enQueue(String string) {
                if(isFull()){
                        System.out.println("ERROR : 큐가 꽉 차서 삽입이 불가능합니다.");
                        return false;
                }else{
                        this.elements[this.rear] = string;
                        this.rear = (this.rear+1)%this.maxSize;
                        return true;
                }

    }

        /*
     * Queue에서 빼는 함수
     * Queue가 비었는지 확인하고
     * 꽉 비어있으면 return false
     * 그렇지 않으면 front가 가리키는 부분의 값을 반환할 변수에 저장하고
     * front가 가리키는 부분을 null로 초기화해준다.
     * 그리고 front를 1 올리고 maxSize로 Mod연산 해 준뒤
     * 삭제했던 값을 반환해준다.
     */
    public String deQueue() {
        if(isEmpty()){
                System.out.println("ERROR : 큐에 원소가 없습니다.");
                return "";
        }else{
                String temp = this.elements[this.front];
                this.elements[this.front] = null;
                this.front = (this.front+1)%this.maxSize;
                return temp;
        }
    }

    /*
     * 받은 Integer값의 크기만큼의 개수를 deQueue해 주는 함수
```

```
	 * for문으로 입력받은 만큼 반복하며 deQueue해준다.
	 */
	public void removes(int i) {
		String temp;
		for(int j = 0; j < i; j++){
			temp = this.deQueue();
			if(temp != "")
				System.out.println("[DeQueue] The Deleted Element is '" + temp
+ "'.");
		}
	}

	/*
	 * 현재 Queue의 모습을 출력하는 함수
	 * Queue에 있는 값을 모두 저장하기 위한 StringBuilder를 선언하고
	 * for문으로 size만큼 반복하며 front에서 i만큼의 offset이 있는 곳의 값을 append해준다.
	 * 반복문이 끝나면 만들어진 String을 반환해준다.
	 */
	public String printQueue() {
		StringBuilder temp = new StringBuilder();
		temp.append("[ ");
		for(int i = 0; i < this.size(); i++){
			temp.append(this.elements[(this.front + i)%this.maxSize] + " ");
		}
		temp.append("]");

		return temp.toString();
	}

	/*
	 * Queue의 front에 있는 값을 반환해주는 함수
	 */
	public String front() {
		return this.elements[this.front];
	}

	/*
	 * Queue의 현재 size를 반환하는 함수
	 * rear가 front보다 크거나 같으면 rear-front를 반환하고
	 * 그 반대이면 maxSize + rear - front를 반환한다.
	 */
	public int size() {
		if(this.front <= this.rear){
			return this.rear - this.front;
		}else{
			return this.maxSize + this.rear - this.front;
		}
	}

	/*
	 * 현재 Queue가 비었는지 안비었는지 반환하는 함수
	 * 비었으면 true를 반환한다.
	 */
	public boolean isEmpty() {
		return this.rear == this.front;
	}

	/*
	 * 현재 Queue가 꽉 찼는지 꽉 차지 않았는지 반환하는 함수
	 * 꽉 찼으면 true를 반환한다.
	 */
	public boolean isFull() {
		return (this.rear + 1)%this.maxSize == this.front;
	}

}
```

CircularArrayQueue.java

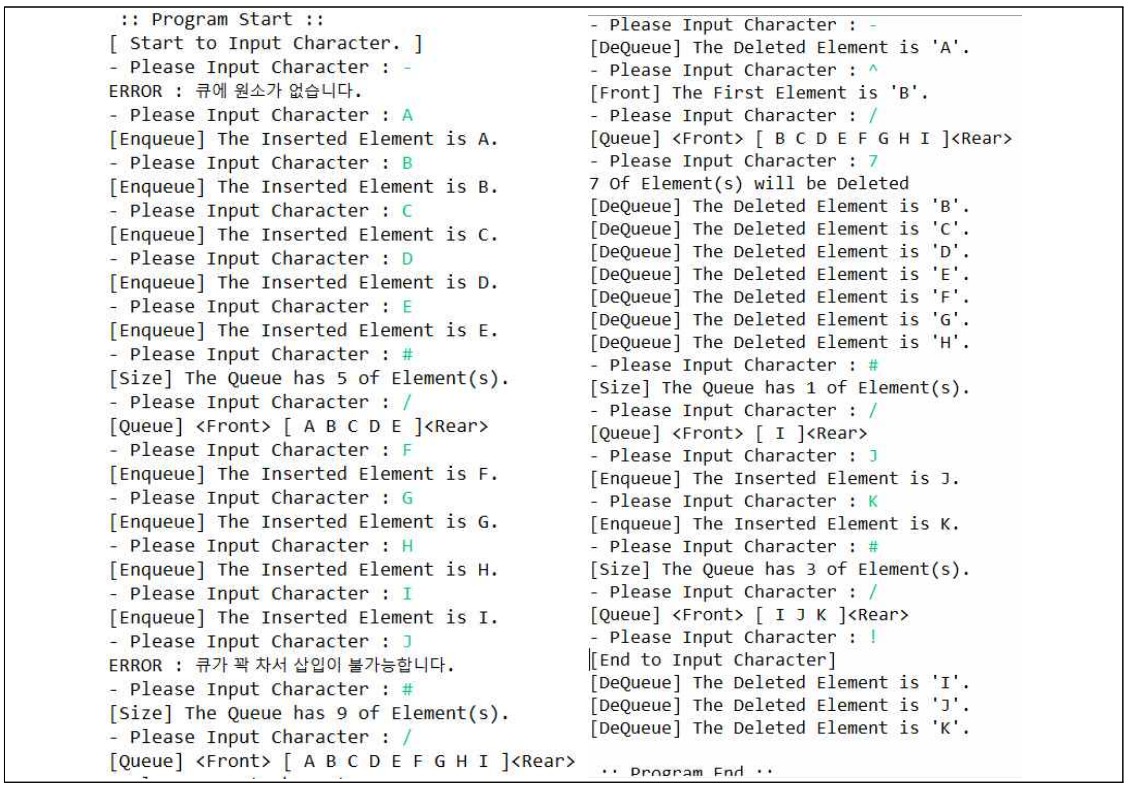
```
import java.util.Scanner;

public class MainClass_06_201502273 {
    public static void main(String args[]) {
        Scanner scanner = new Scanner(System.in);
        CircularArrayQueue queue = new CircularArrayQueue();
        System.out.println(" :: Program Start :: ");
        System.out.println("[ Start to Input Character. ]");
        String string;
        while (true) {
            System.out.print("- Please Input Character : ");
            string = scanner.nextLine();
            if ("!".equals(string)) {
                System.out.println("[End to Input Character]");
                queue.removes(queue.size());
                break;
            }
            switch (string) {
                case "#":
                    System.out.println("[Size] The Queue has " + queue.size() + " of
Element(s).");
                    break;
                case "/":
                    System.out.println("[Queue] <Front> " + queue.printQueue() + "<Rear>
");
                    break;
                case "^":
                    System.out.println("[Front] The First Element is '" + queue.front() + "'.
");
                    break;
                case "-":
                    String dequeue = queue.deQueue();
                    if(dequeue != "")
                                    System.out.println("[DeQueue] The Deleted Element is
'" + dequeue + "'.");
                    break;
                case "0":
                case "1":
                case "2":
                case "3":
                case "4":
                case "5":
                case "6":
                case "7":
                case "8":
                case "9":
                    System.out.println(string + " Of Element(s) will be Deleted");
                    queue.removes(Integer.parseInt(string));
                    break;
                default:
                    if (queue.enQueue(string))
                        System.out.println("[Enqueue] The Inserted Element is " + string +
".");
                    break;
            }

        }
        System.out.println("\n :: Program End :: ");
    }
}
```

MainClass_06_201502273.java

2.결과



```
 :: Program Start ::
[ Start to Input Character. ]
- Please Input Character : -
ERROR : 큐에 원소가 없습니다.
- Please Input Character : A
[Enqueue] The Inserted Element is A.
- Please Input Character : B
[Enqueue] The Inserted Element is B.
- Please Input Character : C
[Enqueue] The Inserted Element is C.
- Please Input Character : D
[Enqueue] The Inserted Element is D.
- Please Input Character : E
[Enqueue] The Inserted Element is E.
- Please Input Character : #
[Size] The Queue has 5 of Element(s).
- Please Input Character : /
[Queue] <Front> [ A B C D E ]<Rear>
- Please Input Character : F
[Enqueue] The Inserted Element is F.
- Please Input Character : G
[Enqueue] The Inserted Element is G.
- Please Input Character : H
[Enqueue] The Inserted Element is H.
- Please Input Character : I
[Enqueue] The Inserted Element is I.
- Please Input Character : J
ERROR : 큐가 꽉 차서 삽입이 불가능합니다.
- Please Input Character : #
[Size] The Queue has 9 of Element(s).
- Please Input Character : /
[Queue] <Front> [ A B C D E F G H I ]<Rear>
- Please Input Character : -
[DeQueue] The Deleted Element is 'A'.
- Please Input Character : ^
[Front] The First Element is 'B'.
- Please Input Character : /
[Queue] <Front> [ B C D E F G H I ]<Rear>
- Please Input Character : 7
7 Of Element(s) will be Deleted
[DeQueue] The Deleted Element is 'B'.
[DeQueue] The Deleted Element is 'C'.
[DeQueue] The Deleted Element is 'D'.
[DeQueue] The Deleted Element is 'E'.
[DeQueue] The Deleted Element is 'F'.
[DeQueue] The Deleted Element is 'G'.
[DeQueue] The Deleted Element is 'H'.
- Please Input Character : #
[Size] The Queue has 1 of Element(s).
- Please Input Character : /
[Queue] <Front> [ I ]<Rear>
- Please Input Character : J
[Enqueue] The Inserted Element is J.
- Please Input Character : K
[Enqueue] The Inserted Element is K.
- Please Input Character : #
[Size] The Queue has 3 of Element(s).
- Please Input Character : /
[Queue] <Front> [ I J K ]<Rear>
- Please Input Character : !
[End to Input Character]
[DeQueue] The Deleted Element is 'I'.
[DeQueue] The Deleted Element is 'J'.
[DeQueue] The Deleted Element is 'K'.

 :: Program End ::
```

A, B, C, D, E의 순서로 Queue에 넣었고 넣은 순서대로 들어가 있는 것을 확인할 수 있다. 또한 넣은 개수대로 Queue의 Size를 얻을 수 있는 것을 볼 수 있었고 Queue에서 원소를 제거를 할 경우 먼저 들어온 값이 삭제되는 것을 볼 수 있어 FIFO에 대한 내용을 직접 확인할 수 있었다.

깨달은 점 및 결론

Queue는 First-in-first-out(FIFO)구조로서 먼저 추가된 요소가 먼저 처리되게 된다. 해당 구현에서는 Queue에 추가하는 기능, 빼는 기능, front에 있는 값이 무엇인지 보는 기능정도만 구현되어있지만 n번째 위치에 있는 원소를 참조할 수 있는 메소드도 구현이 될 수 있을 것이다.

또한 Queue는 처리속도가 서로 다른 두 기기간의 버퍼로 사용될 수 있다. 처리속도가 빠른 IC에서 처리속도가 느린 IC로 값을 보낼 때 보내는 값을 Queue로 되어있는 버퍼에 넣어주어서 원활하게 처리될 수 있도록 도울 수 있다.