자료구조 실습 보고서

[제 9-1주]

제출일: 19.05.22
학번/이름: 201603867/조성환

1. 프로그램 설명서

1) 주요 알고리즘/ 자료구조/ 기타

Queue라는 자료구조를 사용함. queue는 선입선출로 입력을 위한 마지막위치, 출력을 위한 처음 위치를 기억하는 식으로 자료구조를 이용함.

2) 함수 설명서

.main()
AppController의 객체를 생성함.
appcontroller.run()을 통해 프로그램을 실행함.

AppController
생성자
queue를 set하고, input, added, ignore char를 0으로 셋함.

run()
프로그램 시작의 출력문과 함께 시작함.
inputChar()로 char을 입력받고 이를 input에 넣음
입력받은 input이 '!'이 될 때까지 while을 통해 계속 입력받음.
input에 따라 다른 함수를 실행함.
문자면 addToQueue() 숫자면 removeN() -면 removeOne() # 이면 showQueueSize()
/ 면 showAllFromFront \면 show AllFromRear < 면 showFrontElement > 면 showRearElement. 만약 이외의 문자면 의미없는 문자가 입력되었음을 출력 후 countIgnoredChar()을 실행함
마지막으로 input에 inputChar()을 통해 입력을 받음
만약 !를 입력받아 while문을 나오면 quitQueueProcessing()을 실행
showStatistics()을 실행한 후 종료한다는 출력문과 함께 종료함.

count Input Ignore Added Char()
입력받은, 무시한, 추가한 문자를 세줌. queue가 정상적으로 동작했는지를 알아보기 위함.

inputChar()
AppView.intputChar()을 통해 입력 받음

addToQueue()
queue가 꽉 차있는지 확인 후 꽉 차있으면 못넣는다는 출력문. 공간이 있다면 queue().enQueue로 입력을 한 후 추가된 원소를 출력한 후 countAddedChar()을 해줌.

removeOne()
queue가 empty인지 확인함. 만약 비어있으면 비어있다는 출력문을 냄.
비어있지 않다면 Chararter로 removedChar을 선언한 후. deQueue()를 실행한 리턴값을 받음.

만약 removedChar이 null이면 오류 메시지를 내보내고, 아니라면 삭제된 원소를 출력함.

removeN()
입력받은 숫자가 0 이하인지 확인함. 먼저 removeN에 숫자를 넣는 방법은 Character.getNumericValue(input)을 통해 숫자인지 확인이 가능함.
만약 0보다 크면 queue가 비어있는지를 확인함. 비어있으면 더 이상 삭제할 원소가없다고 출력함. 비어있지 않다면 removeOne()과 같이 실행을 하는데, 이를 n번 반복함.

showAllFromFront()
front임을 나타내는 출력문과 함께 iterator를 선언한 후 hasNExt를 통해 계속 다음 요소를 받아옴. 만약 hasNext가 없다면 while문을 마치고 rear을 출력함.

showAllFromRear()
rear를 출력함
size만큼 (한개를 빼야함) order를 설정한 후 order가 0보다 작을 때 까지 줄임. 해당 위치의 요소를 toString으로 가져와서 출력함
front를 출력함

showFrontElement()
queue가 비었는지를 확인한 후 비어있으면 비어있다는 출력을 함. 비어있지 않다면 front()를 통해 앞의 요소를 출력함.

showRearElement()
queue가 비었는지를 확인한 후 비어있으면 비어있다는 출력을 함. 비어있지 않다면 rear()를 통해 앞의 요소를 출력함.

showQueueSize()
queue.size()를 통해 size를 출력함.

quitQueueProcessing()
queue를 끝내기 위해 실행하는 함수임.
showAllFromFront()를 통해 앞에서부터 보여준 후 removeN으로 size만큼 삭제함.

showStatistics()
사용 통계를 보여준다는 출력문과 함께
입력된 문자의 수를 inputChars()로 정상처리된 문자를 inpuChars()-ignoredChars()
무시된 문자를 ignoredChars() queue에 삽입된 문자를 addedChars()로 나타냄.

CircularArrayQueue

CircularArrayQueue()를 통해 maxLength, frontposition과 rearposition을 -1로, element를 maxLength만큼 배열을 만듬.

elementAt()
order를 집어넣으면 자동으로 frontposition+1+order로 사용자가 계산할 필요없이 queue에서의 order만 집어넣으면 알아서 배열에서의 index로 바꿔줌

front()
비어있다면 null을 리턴함.
비어있지 않다면 frontElement를 frontposition+1의 위치를(ElementAt이라면 0만하면 됨)가져옴. 해당 Element를 리턴함

rear()
비어있다면 null을 리턴함. 비어있지 않다면 this.rearPostion()의 위치의 요소를 가져옴(elementAt이라면 size만큼만 하면 됨)

enQueue()
꽉차있다면 false를 리턴함. 여유가 있다면 rearPosition을 1더해준 후(maxLength가 넘지 않도록 %로 계산해줘야함)rearPosition에 요소를 더한 후 해당 요소를 리턴함. 원래 size를 더해주는데, 더하지 않는 이유는 알아서 rearposition-frontposition을 통해 계산하기 때문임.

deQueue()
만약 queue가 비어있다면 null을 리턴함.
아니라면 frontPosition을 +1한 후 해당 위치의 요소를 가져옴(elementAt 이기 때문에 -1을 넣어줌) frontPosition의 요소를 null로 해줌. 뽑아온 element를 리턴함.

clear()
모든 position을 0으로 해준 후 0번째부터 maxLength만큼 돌면서 모든 요소를 null로 해줌.

circularArratQueueIterator
order와 nextOrder를 입력받음.
hasNext를 통해 다음 요소가있는지 확인, next를 통해 다음 요소를 가져옴.

3) 종합 설명서

Array를 이용해 queue를 구현함. 배열의 index를 front와 rear로 저장해 입력과 출력을 제어함. 맨처음 front, rear를 -1로 설정해 계속 돌림.

2. 프로그램 장단점/ 특이점 분석

프로그램의 장점으로는 잘 모르겠지만, 단점이 매우 크다.
queue자체가 선입 선출이기 때문에 앞과 뒤의 위치를 계속 파악해야함. array기 때문에 전체 숫자가 정해져있고, 앞과 끝을 계속 파악해야하는데, array의 끝이있기 때문에 %capacity를 통해 계속 돌려줘야함. 또한 front rear을 이용해 모든 연산을 해야함. 계산을 해야하기 때문에 시간이 많이 걸림. 이런 불편한점 때문에 array로 queue를 구현하는 것은 좋지 않음.

3. 실행 결과 분석

1) 입력과 출력

입력은 주로 문자, 숫자, 사전에 입력된 특수한 문자로 이루어짐. 이를 통해 추가, 나열 삭제 후 추가, 오류처리가 되도록 삭제 하는 식으로 진행함.

```
<<< 큐 기능 확인 프로그램을 시작합니다 >>>

? 문자를 입력하시오: a
[EnQ] 추가된 원소는'a' 입니다.
? 문자를 입력하시오: s
[EnQ] 추가된 원소는's' 입니다.
? 문자를 입력하시오: d
[EnQ] 추가된 원소는'd' 입니다.
? 문자를 입력하시오: /
[Queue] <Front>a s d <Rear>
? 문자를 입력하시오: -
[DeQ] 삭제된 원소는'a' 입니다.
? 문자를 입력하시오: 4
[DeQs] 삭제된 원소는 's' 입니다.
 (오류) 큐에서 삭제하는 동안에 오류가 발생하였습니다.
[DeQs.Empty] 큐에 더 이상 삭제할 원소가 없습니다.
[DeQs.Empty] 큐에 더 이상 삭제할 원소가 없습니다.
? 문자를 입력하시오: a
[EnQ] 추가된 원소는'a' 입니다.
? 문자를 입력하시오: g
[EnQ] 추가된 원소는'g' 입니다.
? 문자를 입력하시오: r
[EnQ] 추가된 원소는'r' 입니다.
? 문자를 입력하시오: ]
[Ignore] 의미 없는 문자가 입력되었습니다.
? 문자를 입력하시오: W
[Queue] <Rear>r g a <Front>
? 문자를 입력하시오: #
[개수]3개 입니다.
? 문자를 입력하시오: !
[Queue] <Front>a g r <Rear>
[DeQs] 삭제된 원소는 'a' 입니다.
[DeQs] 삭제된 원소는 'g' 입니다.
```

```
? 문자를 입력하시오: r
[EnQ] 추가된 원소는'r' 입니다.
? 문자를 입력하시오: ]
[Ignore] 의미 없는 문자가 입력되었습니다.
? 문자를 입력하시오: w
[Queue] <Rear>r g a <Front>
? 문자를 입력하시오: #
[개수]3개 입니다.
? 문자를 입력하시오: !
[Queue] <Front>a g r <Rear>
[DeQs] 삭제된 원소는 'a' 입니다.
[DeQs] 삭제된 원소는 'g' 입니다.
 (오류) 큐에서 삭제하는 동안에 오류가 발생하였습니다.

<큐 사용 통계>
- 입력된 문자는 12 개 입니다.
- 정상 처리된 문자는 11 개 입니다.
- 무시된 문자는 1 개 입니다.
- 삽입된 문자는 6 개 입니다.

<<< 큐 기능 확인 프로그램을 종료합니다 >>>

Process finished with exit code 0
```

2) 결과 분석

다 구현하려고했던대로 구현을 했으나, ArrayQueue가 너무 헷갈려서 삭제 부분에서 마지막 삭제를 할 때 삭제는 되나, 오류로 처리가 됨. 이 부분을 제외하고 모든 부분을 구현함.

4. 소스 코드

Appcontroller

```java
package controller;

import model.CircularArrayQueue;
import model.Iterator;
import model.Queue;
import view.AppView;

public class AppController {
    //constants
    private static final int QUEUE_CAPACITY= 10;

    //private var
    private Queue<Character> _queue;
    private int _inputChars;
    private int _addedChars;
    private int _ignoredChars;

    //getter setter
    public Queue<Character> queue() { return _queue; }
    public void setQueue(Queue<Character> newQueue) { this._queue = newQueue; }

    public int inputChars() { return _inputChars; }
    public void setInputChars(int newInputChars) { this._inputChars = newInputChars; }

    public int addedChars() { return _addedChars; }
    public void setAddedChars(int newAddedChars) { this._addedChars = newAddedChars; }

    public int ignoredChars() { return _ignoredChars; }
    public void setIgnoredChars(int newIgnoredChars) { this._ignoredChars = newIgnoredChars; }

    //constructors
    public AppController(){
        this.setQueue(
                new CircularArrayQueue<Character>(AppController.QUEUE_CAPACITY));
        this.setInputChars(0);
```

```
                new CircularArrayQueue<Character>(AppController.QUEUE_CAPACITY));
        this.setInputChars(0);
        this.setAddedChars(0);
        this.setIgnoredChars(0);
    }

    //private methods
    //회수 계산
    private void countInputChar(){ this.setInputChars(this.inputChars()+1);}
    private void countIgnoredChar(){ this.setIgnoredChars(this.ignoredChars()+1);}
    private void countAddedChar(){ this .setAddedChars(this.addedChars()+1);}

    //큐 수행 관련
    private void addToQueue(char aCharForAdd){
        if (this.queue().isFull()){
            AppView.outputLine( aMessage: "[EnQ.Full] 큐가 꽉 차서 더 이상 넣을 수가 없습니다.");
        } else {
            this.queue().enQueue((Character)aCharForAdd);
            AppView.outputLine( aMessage: "[EnQ] 추가된 원소는'"+ aCharForAdd+ "' 입니다.");
            this.countAddedChar();
        }
    }
    private void removeOne(){
        if (this.queue().isEmpty()){
            AppView.outputLine( aMessage: "[DeQ.Empty] 큐에 삭제할 원소가 없습니다.");
        } else {
            Character removedChar= this.queue().deQueue();
            if (removedChar== null)
                AppView.outputLine( aMessage: "(오류) 큐에서 삭제하는 동안에 오류가 발생하였습니다.");
            else
                AppView.outputLine( aMessage: "[DeQ] 삭제된 원소는'"+removedChar+ "' 입니다.");
        }
    }
    private void removeN(int numberOfCharsToBeRemoved){
        if (numberOfCharsToBeRemoved<= 0) AppView.outputLine(
```

```
67      private void removeN(int numberOfCharsToBeRemoved){
68          if (numberOfCharsToBeRemoved<= 0) AppView.outputLine(
69                  aMessage: "[DeQs] 삭제할 원소의 개수가 0개 이하 입니다.");
70          else {
71              for(int i= 0; i< numberOfCharsToBeRemoved; i++){
72                  if (this.queue().isEmpty()) AppView.outputLine( aMessage: "" +
73                          "[DeQs.Empty] 큐에 더 이상 삭제할 원소가 없습니다.");
74                  else {
75                      Character removedeCharacter= null;
76                      removedeCharacter= this.queue().deQueue();
77                      if (removedeCharacter== null) AppView.outputLine
78                              ( aMessage: " (오류) 큐에서 삭제하는 동안에 오류가 발생하였습니다.");
79                      else AppView.outputLine( aMessage: "[DeQs] 삭제된 원소는 '"+removedeCharacter+ "' 입니다.");
80                  }
81              }
82          }
83      }
84      private void quitQueueProcessing(){
85          this.showAllFromFront();
86          this.removeN(this.queue().size());
87      }
88
89      //출력 관련
90      private void showAllFromFront(){
91          AppView.output( aMessage: "[Queue] <Front>");
92          Iterator<Character> queueIterator= this.queue().iterator();
93          while (queueIterator.hasNext()){
94              Character element= queueIterator.next();
95              AppView.output( aMessage: element.toString()+ " ");
96          }
97          AppView.outputLine( aMessage: "<Rear>");
98      }
99      private void showAllFromRear(){
100         AppView.output( aMessage: "[Queue] <Rear>");
101         for (int order= this.queue().size()-1; order>= 0; order--){
```

```
        AppView.output( aMessage: "[Queue] <Rear>");
        for (int order= this.queue().size()-1; order>= 0; order--){
            AppView.output( aMessage: this.queue().elementAt(order).toString()+ " ");
        }
        AppView.outputLine( aMessage: "<Front>");
    }
    private void showFrontElement(){
        if (this.queue().isEmpty()){
            AppView.outputLine( aMessage: "비어있음");
        } else {
            AppView.outputLine( aMessage: "[Queue]"+this.queue().front());
        }
    }
    private void showRearElement(){
        if (this.queue().isEmpty()){
            AppView.outputLine( aMessage: "비어있음");
        } else {
            AppView.outputLine( aMessage: "[Queue]"+ this.queue().rear());
        }
    }
    private void showQueueSize() { AppView.outputLine( aMessage: "[개수]"+ this.queue().size()+ "개 입니다."); }
    private void showStatistics(){
        AppView.outputLine( aMessage: "");
        AppView.outputLine( aMessage: "<큐 사용 통계>");
        AppView.outputLine( aMessage: "- 입력된 문자는 "+ this.inputChars()+ " 개 입니다.");
        AppView.outputLine
                ( aMessage: "- 정상 처리된 문자는 "+ (this.inputChars()- this.ignoredChars())+ " 개 입니다.");
        AppView.outputLine( aMessage: "- 무시된 문자는 "+ this.ignoredChars()+ " 개 입니다.");
        AppView.outputLine( aMessage: "- 삽입된 문자는 "+ this.addedChars()+ " 개 입니다.");
    }

    //입력 관련
    private char inputChar(){
        AppView.output( aMessage: "? 문자를 입력하시오: ");
```

```
135             AppView.output( aMessage: "? 문자를 입력하시오: ");
136             return AppView.inputChar();
137         }
138
139         public void run(){
140             AppView.outputLine( aMessage: "<<< 큐 기능 확인 프로그램을 시작합니다 >>>");
141             AppView.outputLine( aMessage: "");
142
143             char input= this.inputChar();
144             while (input!= '!'){
145                 this.countInputChar();
146                 if ((Character.isAlphabetic(input))) {
147                     this.addToQueue( Character.valueOf(input));
148                 } else if (Character.isDigit(input)){
149                     this.removeN(Character.getNumericValue(input));
150                 } else if (input== '-'){
151                     this.removeOne();
152                 } else if (input== '#'){
153                     this.showQueueSize();
154                 } else if (input== '/'){
155                     this.showAllFromFront();
156                 } else if (input== '##'){
157                     this.showAllFromRear();
158                 } else if (input== '<'){
159                     this.showFrontElement();
160                 } else if (input== '>'){
161                     this.showRearElement();
162                 } else {
163                     AppView.outputLine(
164                             aMessage: "[Ignore] 의미 없는 문자가 입력되었습니다.");
165                     this.countIgnoredChar();
166                 }
167                 input= this.inputChar();
168             }
```

```
166                 }
167                 input= this.inputChar();
168             }
169             this.quitQueueProcessing();
170
171             this.showStatistics();
172             AppView.outputLine( aMessage: "");
173             AppView.outputLine( aMessage: "<<< 큐 기능 확인 프로그램을 종료합니다 >>>");
174         }
175
176     }
177
```

ArrayQueue

```java
package model;

public class CircularArrayQueue<E> implements Queue<E> {
    //Constants
    private static final int DEFAULT_CAPACITY= 100;

    //instance Var
    private int _maxLength;
    private int _frontPosition;
    private int _rearPosition;
    private E[] _elements;

    //getter setter
    public int maxLength() { return _maxLength; }
    public void setMaxLength(int newMaxLength) { this._maxLength = newMaxLength; }

    public int frontPosition() { return _frontPosition; }
    public void setFrontPosition(int newFrontPosition) { this._frontPosition = newFrontPosition; }

    public int rearPosition() { return _rearPosition; }
    public void setRearPosition(int newRearPosition) { this._rearPosition = newRearPosition; }

    public E[] elements() { return _elements; }
    public void setElements(E[] newElements) { this._elements = newElements; }

    //constructors
    /unchecked/
    public CircularArrayQueue(int givenCapacity){
        this.setMaxLength(givenCapacity+1);
        this.setFrontPosition(-1);
        this.setRearPosition(-1);
        this.setElements((E[]) new Object[this.maxLength()]);
    }
    public CircularArrayQueue(){ this(CircularArrayQueue.DEFAULT_CAPACITY);}
```

```java
    public CircularArrayQueue(){ this(CircularArrayQueue.DEFAULT_CAPACITY);}

    @Override
    public Iterator<E> iterator() { return new CircularArrayQueueIterator(); }

    public int capacity(){ return (this.maxLength()-1); }
    public int size(){
        if (this.rearPosition()>= this.frontPosition()){
            return (this.rearPosition()- this.frontPosition());
        } else {
            return (this.rearPosition() + this.maxLength() - this.frontPosition());
        }
    }
    public boolean isEmpty(){
        if(this.frontPosition()== this.rearPosition()) return true;
        else return false;
    }
    public boolean isFull(){
        if (this.maxLength()+1 == this.frontPosition()) return true;
        else return false;
    }

    //검색
    public E front(){
        E frontElement= null;
        if(this.isEmpty()) return null;
        else {
            frontElement= this.elements()[frontPosition()+ 1];
            return frontElement;
        }
    }
    public E rear(){
        E rearElement= null;
        if (this.isEmpty()) return null;
```

```
67          if (this.isEmpty()) return null;
68          else {
69              rearElement= this.elements()[this.rearPosition()];
70              return rearElement;
71          }
72      }
73      public E elementAt(int anOrder){
74          if (this.isEmpty()) return null;
75          else return this.elements()[(this.frontPosition()+1+anOrder)% this.maxLength()];
76      }
77
78      //추가
79      public boolean enQueue(E anElement){
80          if (this.isFull()) return false;
81          else {
82              this.setRearPosition(this.rearPosition()+1%this.maxLength());
83              this.elements()[this.rearPosition()]= anElement;
84              return true;
85          }
86      }
87
88      //제거
89      public E deQueue(){
90          if (this.isEmpty()) return null;
91          else {
92              E anElement= null;
93              this.setFrontPosition(this.frontPosition()+1);
94              anElement= this.elementAt( anOrder: -1);
95              this.elements()[this.frontPosition()]= null;
96              return anElement;
97          }
98      }
99      public void clear(){
100         this.setFrontPosition(0); this.setRearPosition(0);
```

```
        }
    }

    private class CircularArrayQueueIterator implements Iterator<E>{
        private int _nextOrder;

        public int nextOrder() { return _nextOrder; }
        public void setNextOrder(int newNextOrder) { this._nextOrder = newNextOrder; }

        private CircularArrayQueueIterator(){ this.setNextOrder(0);}

        @Override
        public boolean hasNext() { return (this.nextOrder()< CircularArrayQueue.this.size()); }
        @Override
        public E next(){
            E nextElement= null;
            if (this.hasNext()){
                nextElement= CircularArrayQueue.this.elementAt(this.nextOrder());
                this.setNextOrder(this.nextOrder()+1);
            }
            return nextElement;
        }
    }
}

```