Stack과 Queue

작성일시	@2023년 4월 15일 오후 4:16
강의 번호	자바의정석 Chapter11
유형	
자료	
복습	☐
Spring Framwork	

▼ 큐(Queue)

- FIFO(First In First Out)
- 데이터를 꺼낼 때 항상 첫 번째 저장된 데이터를 삭제하므로, ArrayList보다 LinkedList를 사용한다.

```
//Stack
boolean empty() : stack이 비어있는지 알려준다.
Object peek() : 스택의 맨 위에 저장된 객체를 반환. pop()와 달리 스택에서 객체를 꺼내지는 않음
                비어있을 때는 EmptryStackException발생
Object pop() : 스택의 맨 위에 저장된 객체를 꺼낸다. peek와 동일하게 비어있을때 예외 발생
Object push(Object item) : 스택에 객체를 저장한다.
int search(Object o) : 스택에서 주어진 객체를 찾아서 그 위치를 반환 못 찾으면 -1 반환.
                        배열과 달리 위치는0이 아닌 1부터 시작한다.


//Queue
boolean add(Object o) : 지정된 객체를 큐에 추가한다. 성공하믄 true반환 저장공간이 부족하면 illegalState예외 발생
Object remove() 큐에서 객체를 꺼내 반환. 비어있으면 NoSuchElement발생
Object element() : 삭제없이 요소를 읽어온다. peek와 달리 Queue가 비었을 때 NoSuchElemnet예외발생
boolean offer(Object o) : 큐객체를 저장. 성공하면 true
Object poll() : 큐에서 객체를 꺼내서 반환 비어있으면 null
Object peek() 삭제없이 요소를 읽어온다. 큐가 비어있으면 null반환
```

- stack을 직접 만들어본 코드

```
package ch11;

import java.util.EmptyStackException;
import java.util.Vector;

class MyStack extends Vector{
  public Object push(Object item) {
    addElement(item);
```

```
    return item;
  }
  public Object pop() {
    Object obj = peek();
    removeElementAt((size()-1);
      return obj;
  }
  public Object peek() {
    int len= size();

    if (len==0)
      throw new EmptyStackException();
    return elementAt(len-1);
  }
  public boolean empty() {
    return size()==0;
  }
  public int search(Object o) {
    int i= lastIndexOf(o);

    if(i>=0) {
      return size()-1;

    }
    return -1;
  }

}
```

▼ 스택과 큐의 활용

- 스택 활용 : 수식계산, 수식괄호 검사, 워드프로세서의 undo/redo, 웹브라우저의 뒤로/앞으로
- 큐 활용 : 최근사용문서, 인쇄작업 대기목록, 버퍼
- undo/redo

```
package ch11;
import java.util.*;

public class StackEx1 {
  public static Stack back = new Stack();
  public static Stack forward = new Stack();

  public static void main(String[] args) {
    goUrl("1.네이트");
    goUrl("2.야후");
    goUrl("3. 네이버");
    goUrl("4.다음");

    printStatus();

    goBack();
    System.out.println("= '뒤로' 버튼을 누른 후 =");
    printStatus();

    goBack();
```

```
    System.out.println("= '뒤로' 버튼을 누른 후 =");
    printStatus();

    goForward();
    System.out.println("= '앞으로' 버튼을 누른 후 =");
    printStatus();

    goUrl("codechobo.com");
    System.out.println("=새로운 주소로 이동 후 =");
    printStatus();
  }

    public static void printStatus() {
      System.out.println("back:"+back);
      System.out.println("forward:"+forward);
      System.out.println("현재 화면은 '" + back.peek()+"' 입니다.");
      System.out.println();
    }

    public static void goUrl(String url) {
      back.push(url);
      if(!forward.empty())
        forward.clear();
    }

    public static void goForward() {
      if(!forward.empty())
        back.push(forward.pop());
    }
    public static void goBack() {
      if(!back.empty())
        forward.push(back.pop());
    }
  }
```

- history기능 구현

```
package ch11;

import java.util.*;

class QueueEx1 {
  static Queue q = new LinkedList();
  static final int MAX_SIZE = 5;

  public static void main(String[] args) {
    while (true) {
      System.out.println(">>");
      try {
        Scanner sc = new Scanner(System.in);
        String input = sc.nextLine().trim();

        if ("".equals(input))
          continue;

        if (input.equalsIgnoreCase("q")) {
          System.exit(0);
        } else if (input.equalsIgnoreCase("help")) {
          System.out.println("help - 도움말을 보여줍니다.");
          System.out.println("q또는 Q - 프로그램 종료");
```

```
          System.out.println("history - 최근 입력한 명령어를" + MAX_SIZE + "개 보여줍니다.");
        } else if (input.equalsIgnoreCase("history")) {
          int i = 0;
          save(input);

          LinkedList tmp = (LinkedList) q;
          ListIterator it = tmp.listIterator();

          while (it.hasNext()) {
            System.out.println(++i + "." + it.next());
          }
        } else {
          save(input);
          System.out.println(input);
        }
      } catch (Exception e) {
        System.out.println("입력 오류입니다");

      }

    }
  }// main

  public static void save(String input) {
    if (!"".equals(input))
      q.offer(input);

    if (q.size() > MAX_SIZE)
      q.remove();
  }

}// class
```