알고리즘 1주차

자료구조 복습

MMC 연구실

박사 과정 문희찬

조교 소개

- 문희찬
- 컴퓨터공학과 대학원 석사과정
- MMC연구실 (A1406)
- HCMoon@hallym.ac.kr

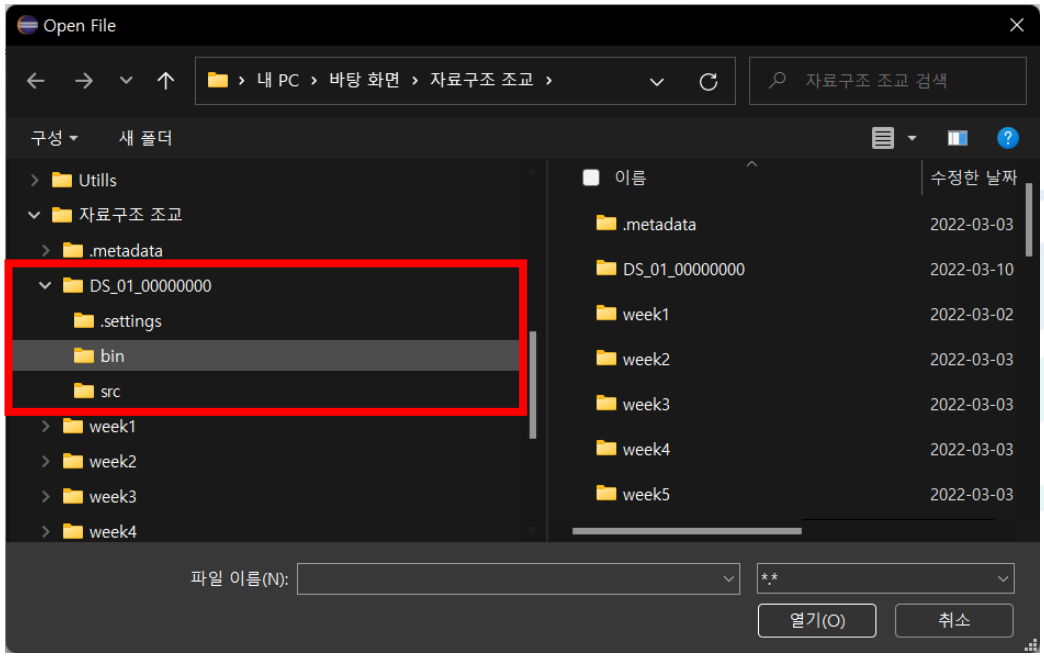
실습 수업 진행 방식

- 쉬는 시간 없이 1시간 30분 수업 (화장실 자유롭게 다녀오세요)
- 출석체크 : 수업 시작, 수업 끝날 때 체크
- 수업 시작 30분 뒤부터, 확인 문제를 해결한 학생은 검사 받고 퇴실
- 과제 진행 중 모르는 부분은 메일로 질문

과제 설명

- 알고리즘 수업은 Eclipse를 사용하여 코드를 작성합니다.
- 확인 문제 및 과제를 전부 해결하여 제출해주세요.
- 과제 제출 시 프로젝트 폴더를 압축해서 제출합니다.
- 과제의 채점은 프로젝트의 실행 결과를 기준으로 점수를 매깁니다.
- 컨닝 금지, 모르는 것이 있으면 저에게 질문해주세요.
  (메일 주소 확인)

과제 제출 방법



- 프로젝트 폴더를 압축하여 제출

- 프로젝트이름 : AL_(주차)_(학번)

예) AL_01_00000000

- *.java파일만 제출하면 안됩니다.

- 제출양식을 반드시 지켜주세요!

확인문제 (factorial)

Package Name : recursion
Class Name : Factorial

```java
package recursion;

public class Factorial {

    public static void main(String[] args) {
        System.out.println(factorial1(5));
        System.out.println(factorial2(5));
    }


    public static int factorial1(int number) {
        // 재귀함수를 이용하여 구현

    }

    public static int factorial2(int number) {
        // 반복문을 이용하여 구현
    }
}
```

```
<terminated> Factorial [Java Application]
120
120
```

확인문제 (factorial)

- $n = 0$: 1
- $n \geq 1$: $n \cdot (n-1) \cdot \ldots \cdot 2 \cdot 1 = n \cdot (n-1)!$

```
factorial(n)
        if  (n ≤ 1) then return 1
         else return (n · factorial(n –1));
 end factorial()
```

확인문제 (Binarysearch)

Package Name : recursion
Class Name : Binarysearch

```java
public class BinarySearch {

    public static void main(String[] args) {
        int array1 [] = {1, 6, 13, 41, 45, 68, 70, 74, 81, 100};
        int array2 [] = {100, 68, 13, 41, 45, 6, 70, 74, 81, 1};
        System.out.println("array1에서 68의 위치 : " + search(array1, 68));
        System.out.println("array2에서 68의 위치 : " + search(array2, 68));
    }

    public static int search(int a [], int key) {
        // 배열이 정렬되어있는지 확인
        // 배열이 정렬되어있지 않다면 -1 리턴

        // binarySearch 메소드 호출
        // return binarySearch(???);

    }


    private static int binarySearch(int array [], int key, int left, int right) {
        // 재귀 알고리즘을 이용해 binary search 구현
    }


}
```

```
<terminated> BinarySearch [Java Application
array1에서 68의 위치 : 5
ERROR : 배열이 정렬되어 있지 않습니다.
array2에서 68의 위치 : -1
```

확인문제 (Binarysearch)

- key = a[mid] : 탐색 성공, return mid
- key < a[mid] : a[mid]의 왼편에 대해 이진탐색
- key > a[mid] : a[mid]의 오른편에 대해 이진탐색

확인문제 (Binarysearch)

```
binsearch(a[], key, left, right)
if (left ≤ right) then {
mid ← (left + right) / 2;
            case {
                    key = a[mid] : return (mid);
                    key < a[mid] : return (binsearch(a, key, left, mid - 1));
                    key > a[mid] : return (binsearch(a, key, mid + 1, right));
            }
        }
        else return -1;
 end binsearch()
```

선형리스트

- 선형 리스트 (linear list)
  - 순서를 가진 원소들의 순열(sequence)
  - 물리적 순서가 아닌 원소의 특성에 의한 논리적 순서를 의미
  - 리스트는 기본적으로 순서 개념을 가지므로 선형 리스트라고 볼 수 있음

- 리스트 L=(e1, e2, … , en)
  - L은 리스트 이름, ei는 리스트 원소
  - 공백 리스트(empty list, 원소가 하나도 없는 리스트)의 표현 : L=( )
  - 리스트의 각 원소는 선행자(predecessor)와 후속자(successor)를 가짐
  - Ex) 알고리즘 강의 요일 = (월요일, 수요일, 금요일)
    토요일 강의 과목 = ()

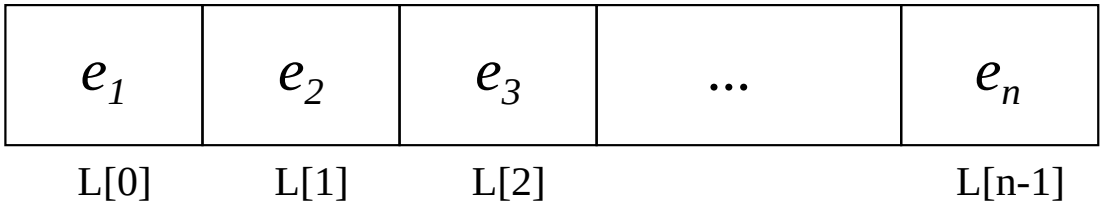
선형리스트

- **isEmpty(L):** 리스트 L이 공백인지 아닌지 결정
- **length(L):** 리스트 L의 길이를 계산함. 여기서 리스트 길이는 리스트에 포함된 원소의 수. 공백리스트의 길이는 0
- **retrieve(L, i):** 리스트 L의 i번째 원소를 검색 ($1 \le i \le$ L의 길이)
- **delete(L, x):** 공백이 아닌 리스트 L로부터 원소 x를 제거. 이 때 리스트 L의 길이는 하나 감소
- **insert(L, i, x):** 새로운 원소 x를 리스트 L의 지정된 위치 i에 삽입. 이 때 리스트 원소 e_i, e_{i+1}, …,e_n은 e_{i+1}, …, e_n, e_{n+1}로 되고, 리스트 L의 길이는 하나 증가

선형리스트

- 배열을 사용해 표현 (순차 표현 리스트)
  - 리스트 원소 ei와 ei+1이 인덱스 i-1과 i에 대응되게 **연속적으로 저장**
  - 원소의 물리적 순서로 논리적 순서를 나타냄 (순서를 표시하기 위한 특별한 장치가 필요 없음)
  - 삽입, 삭제시에 후속 원소들을 한자리씩 밀거나 당겨야 하는 오버헤드가 치명적인 약점

$$L = (e_1, e_2, \ldots, e_n)$$



선형 리스트 주의사항

- **다음 조건을 만족하도록 구현**

- 리스트에 원소를 삽입할 때, 해당 인덱스 원소를 한 칸 뒤로 민 뒤에 새로운 원소 추가

- 리스트에서 특정 원소를 삭제할 때, 해당 원소를 삭제 후 리스트 원소를 한 칸 씩 당겨 줌

확인문제 (선형리스트)

Package Name : list
Class Name : LinearList

```
class LinearList {
    private String strArray[];
    private int size;
    public static int MAX = 100;

    public LinearList() {
        size = 0;
        strArray = new String[MAX];
    }

    public boolean isEmpty(){
        // 리스트가 비어있는지 리턴하는 메소드
        // 비어 있을 경우 true, 원소가 한 개 이상 존재할 경우 false
    }

    public int length(){
        // 리스트에 몇 개의 원소가 있는지 리턴하는 메소드
    }
```

```
    public String retrieve(int i){
        // 해당 인덱스의 원소를 리턴하는 메소드
    }

    public void delete(int i){
        // 리스트의 i 번째 원소를 삭제하는 메소드
    }

    public void insert(int i, String str){
        // 리스트에서 i 번째에 str 원소를 삽입하는 메소드
    }

    public void printArray(){
        // 리스트의 문자열을 전부 출력하는 메소드
    }
```

확인문제 (선형리스트)

Package Name : list
Class Name : Main

```
public class Main {

    public static void main(String[] args) {
        LinearList linear = new LinearList();

        System.out.println(linear.isEmpty());

        linear.insert(0, "A");
        linear.insert(0, "C");
        linear.insert(1, "AA");
        linear.insert(2, "B");

        linear.printArray();

        linear.delete(1);

        System.out.println(linear.isEmpty());
        linear.printArray();

    }

}
```

```
true
[0] : C
[1] : AA
[2] : B
[3] : A
false
[0] : C
[1] : B
[2] : A
```

실습 과제

1. 피보나치 수열 재귀, 반복문 구현

2. Palindrome

3. 연결리스트 구현

과제 1

Package Name : recursion
Class Name : Fibo

```java
public class Fibo {
    public static final int MAX_N = 10;
    public static void main(String[] args) {
        for(int i = 0; i <= MAX_N; i++)
            System.out.println(fib(i));

        System.out.println("------------------------------");

        for(int i = 0; i <= MAX_N; i++)
            System.out.println(fibIter(i));
    }

    public static long fib(int n) {
        // 재귀함수를 이용해 n번째 피보나치 수열의 값 리턴
    }

    public static long fibIter(int n) {
        // 반복문을 이용해 n번째 피보나치 수열의 값 리턴
    }


}
```

```
<terminated> Fibo [Java Application]
0
1
1
2
3
5
8
13
21
34
55
------------------------------
0
1
1
2
3
5
8
13
21
34
55
```

과제 2 (Palindrome)

- Palindrome : “eye”, “kayak”처럼 거꾸로 읽어도 제대로 읽는 것과 같은 문자열

- Palindrome인지 확인하는 방법

  - 반복문 혹은 재귀함수를 이용해 문자를 비교

  “kayak” → “kayak” → “kayak” → O

  “apple” → X

- 문자열에서 문자를 하나 가져오는 법

```
String str = “abba”;
str.charAt(0); // “a”
str.charAt(1); // “b”
```

과제 2 (Palindrome)

Package Name : recursion
Class Name : Palindrome

- **재귀함수를 이용**해 Palindrome인지 판단하는 메소드 구현

```
public class Palindrome {

    public static void main(String[] args) {
        System.out.println("abba : " + isPalin("abba"));
        System.out.println("abcba : " + isPalin("abcba"));
        System.out.println("abba : " + isPalin("accba"));
    }
    private static boolean isPalin(String s, int j, int k) {

    }

    public static boolean isPalin(String s) {
        return isPalin();
    }

}
```

```
<terminated> Palindrome [Java Application]
abba : true
abcba : true
abba : false
```

과제 3

Package Name : linked
Class Name : ListNode

```
public class ListNode {
    private String name;
    private ListNode link;

    public ListNode() {
        link = null;
    }

    public ListNode(String name) {
        this.name = name;
        link = null;
    }

    public ListNode(String name, ListNode link) {
        this.name = name;
        this.link = link;
    }

    public void setName(String name) { this.name = name; }

    public void setLink(ListNode link) { this.link = link; }

    public String getName() { return name; }

    public ListNode getLink() { return link; }

}
```

과제 3

Package Name : linked
Class Name : LinkedList

```java
public class LinkedList {
    private int length;
    private ListNode first;

    public LinkedList() {
        length = 0;
        first = null;
    }

    public int length() {
        // 현재 연결 리스트의 노드 개수를 반환
    }

    public void addFirst(String name) {
        // 연결 리스트의 맨 앞에 노드를 추가
    }

    public void insert(String name, ListNode target) {
        // target 노드 뒤에 새로운 노드를 추가
    }

    public ListNode searchNode(String name) {
        // 전달인자로 받은 name을 연결리스트에서 탐색하여, 해당 노드를 반환
        // 찾지 못했을 경우 null을 반환
    }

    public void delete(ListNode p) {
        // 전달 받은 ListNode p 뒤에 있는 노드를 삭제
    }

    public void print() {
        // 연결리스트에 담겨있는 모든 노드의 name을 순서대로 출력
    }

}
```

과제 3

Package Name : linked
Class Name : Main

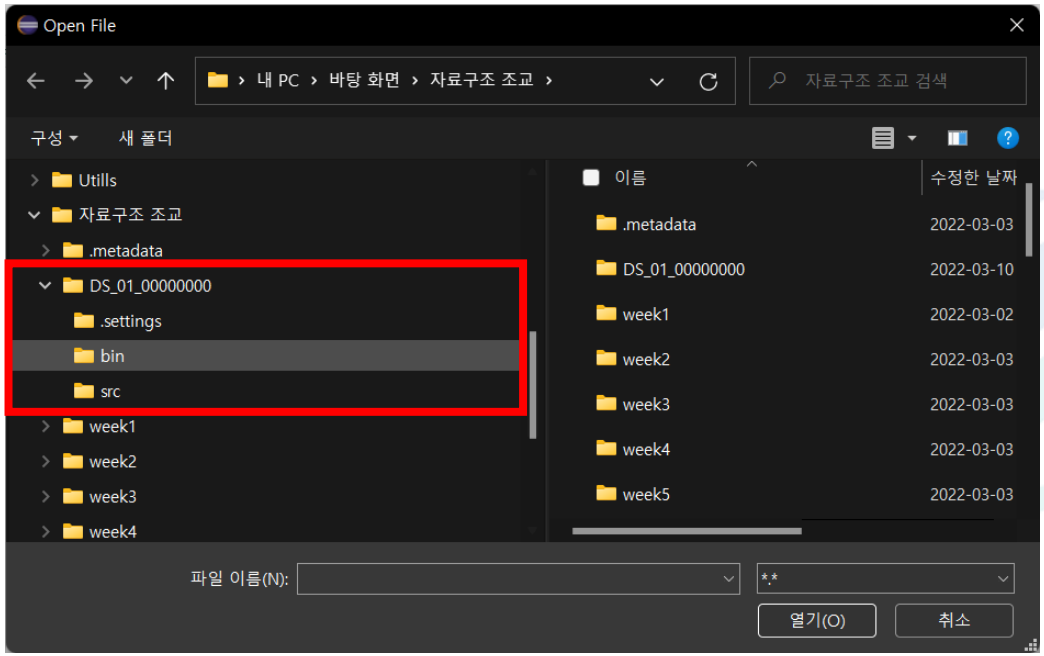
```
public class LinkedListMain {
    public static void main(String[] args) {
        LinkedList list1 = new LinkedList();
        list1.addFirst("Kim");
        list1.addFirst("Choi");
        list1.insert("Moon", list1.searchNode("Kim"));
        list1.print();

        list1.delete( list1.searchNode("Kim"));
        list1.print();


    }
}
```

```
---------------
Choi
Kim
Moon
---------------
---------------
Choi
Kim
---------------
```

과제 제출 방법



- 프로젝트 폴더를 압축하여 제출

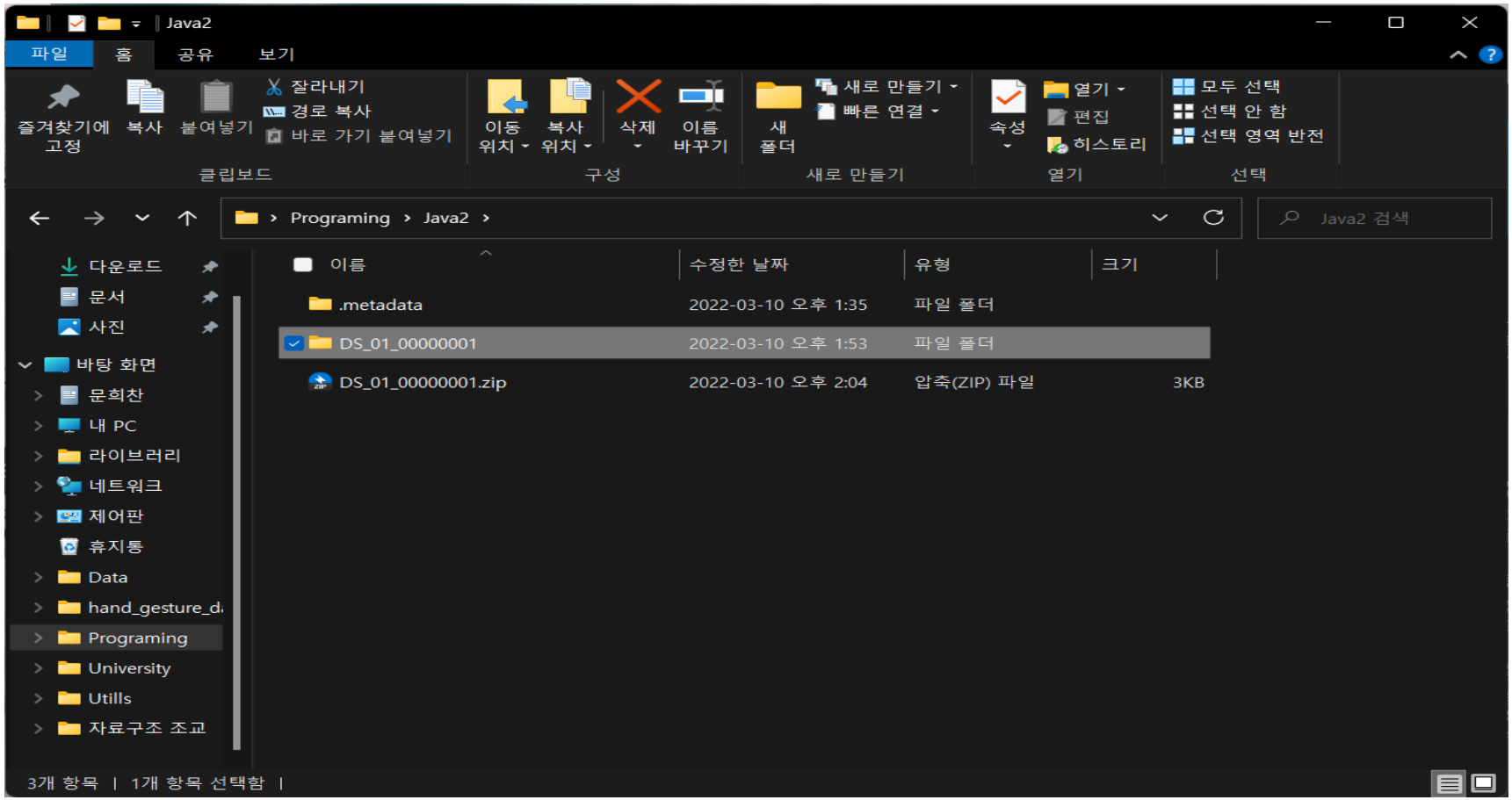
- 프로젝트이름 : AL_(주차)_(학번)

예) AL_01_00000000

- *.java파일만 제출하면 안됩니다.

제출양식을 반드시 지켜주세요

과제 제출 방법



- 반드시 **프로젝트 폴더를 압축**하여 제출