[제 13 주 실습]

AVL Tree

강 지 훈
jhkang@cnu.ac.kr

충남대학교 컴퓨터공학과

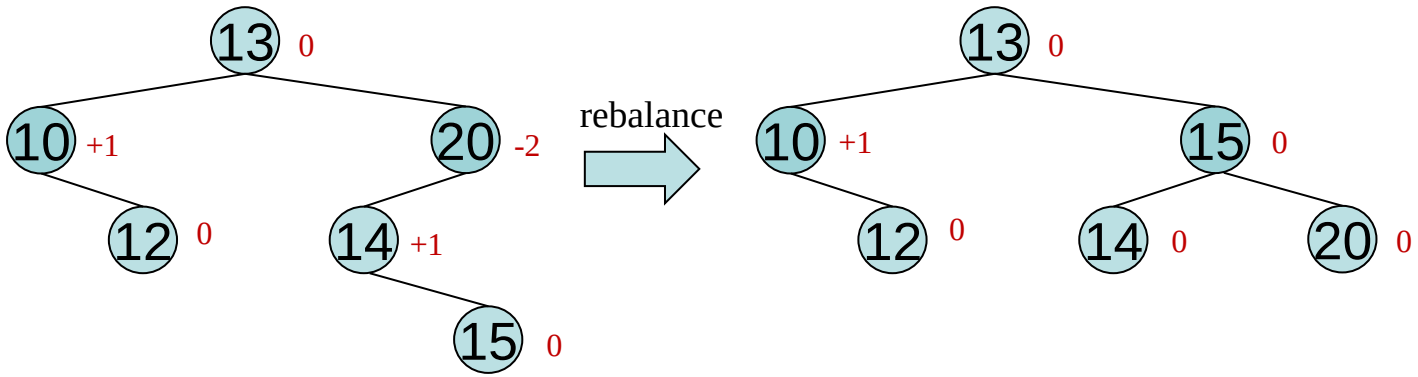
[문제 13]
AVL TREE

문제 개요

- AVL Tree를 실제로 구현하여 본 뒤 AVL Tree를 이용하여 값을 얻어 내는 방법을 추가적으로 구현한다.
- http://webdiis.unizar.es/asignaturas/EDA/AVLTree/avltree.html

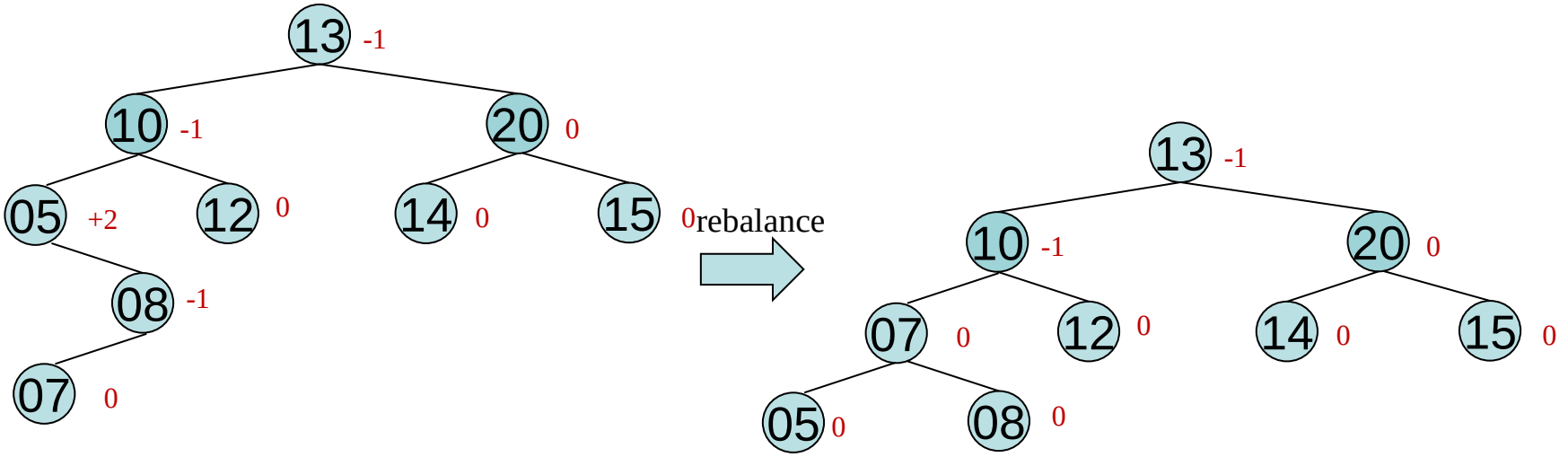
❑출력화면 [1]

```
<<프로그램을 시작합니다>>
수행 하려고 하는 메뉴를 선택하세요(add : 1, remove : 2, debug : 3, exit : -1) : 1
Key를 입력하세요 : 13
수행 하려고 하는 메뉴를 선택하세요(add : 1, remove : 2, debug : 3, exit : -1) : 1
Key를 입력하세요 : 10
수행 하려고 하는 메뉴를 선택하세요(add : 1, remove : 2, debug : 3, exit : -1) : 1
Key를 입력하세요 : 20
수행 하려고 하는 메뉴를 선택하세요(add : 1, remove : 2, debug : 3, exit : -1) : 1
Key를 입력하세요 : 12
수행 하려고 하는 메뉴를 선택하세요(add : 1, remove : 2, debug : 3, exit : -1) : 1
Key를 입력하세요 : 14
수행 하려고 하는 메뉴를 선택하세요(add : 1, remove : 2, debug : 3, exit : -1) : 1
Key를 입력하세요 : 15
수행 하려고 하는 메뉴를 선택하세요(add : 1, remove : 2, debug : 3, exit : -1) : 3
[DEBUG] PRINT AVL TREE
Left:  (10,2)  Key: (13,1) Right:  (15,6)  Parent: null Balance: 0
Left: null Key: (10,2) Right:  (12,4)  Parent:  (13,1)  Balance: 1
Left: null Key: (12,4) Right: null Parent:  (10,2)  Balance: 0
Left:  (14,5)  Key: (15,6) Right:  (20,3)  Parent:  (13,1)  Balance: 0
Left: null Key: (14,5) Right: null Parent:  (15,6)  Balance: 0
Left: null Key: (20,3) Right: null Parent:  (15,6)  Balance: 0
```



❑출력화면 [2]

```
수행 하려고 하는 메뉴를 선택하세요(add : 1, remove : 2, debug : 3, exit : -1) : 1
Key를 입력하세요 : 5
수행 하려고 하는 메뉴를 선택하세요(add : 1, remove : 2, debug : 3, exit : -1) : 1
Key를 입력하세요 : 8
수행 하려고 하는 메뉴를 선택하세요(add : 1, remove : 2, debug : 3, exit : -1) : 1
Key를 입력하세요 : 7
수행 하려고 하는 메뉴를 선택하세요(add : 1, remove : 2, debug : 3, exit : -1) : 3
[DEBUG] PRINT AVL TREE
Left:  (10,2)  Key: (13,1) Right:  (15,6)  Parent: null Balance: -1
Left:  (7,9)  Key: (10,2) Right:  (12,4)  Parent:  (13,1)  Balance: -1
Left:  (5,7)  Key: (7,9) Right:  (8,8)  Parent:  (10,2)  Balance: 0
Left: null Key: (5,7) Right: null Parent:  (7,9)  Balance: 0
Left: null Key: (8,8) Right: null Parent:  (7,9)  Balance: 0
Left: null Key: (12,4) Right: null Parent:  (10,2)  Balance: 0
Left:  (14,5)  Key: (15,6) Right:  (20,3)  Parent:  (13,1)  Balance: 0
Left: null Key: (14,5) Right: null Parent:  (15,6)  Balance: 0
Left: null Key: (20,3) Right: null Parent:  (15,6)  Balance: 0
```



❑출력화면 [3]

```
수행 하려고 하는 메뉴를 선택하세요(add : 1, remove : 2, debug : 3, exit : -1) : 2
Key를 입력하세요 : 12
수행 하려고 하는 메뉴를 선택하세요(add : 1, remove : 2, debug : 3, exit : -1) : 3
[DEBUG] PRINT AVL TREE
Left:  (7,9)  Key: (13,1) Right:  (15,6)  Parent: null Balance: -1
Left:  (5,7)  Key: (7,9) Right:  (10,2)  Parent:  (13,1)  Balance: 1
Left: null Key: (5,7) Right: null Parent:  (7,9)  Balance: 0
Left:  (8,8)  Key: (10,2) Right: null Parent:  (7,9)  Balance: -1
Left: null Key: (8,8) Right: null Parent:  (10,2)  Balance: 0
Left:  (14,5)  Key: (15,6) Right:  (20,3)  Parent:  (13,1)  Balance: 0
Left: null Key: (14,5) Right: null Parent:  (15,6)  Balance: 0
Left: null Key: (20,3) Right: null Parent:  (15,6)  Balance: 0
수행 하려고 하는 메뉴를 선택하세요(add : 1, remove : 2, debug : 3, exit : -1) : 2
Key를 입력하세요 : 10
수행 하려고 하는 메뉴를 선택하세요(add : 1, remove : 2, debug : 3, exit : -1) : 2
Key를 입력하세요 : 7
수행 하려고 하는 메뉴를 선택하세요(add : 1, remove : 2, debug : 3, exit : -1) : 2
Key를 입력하세요 : 20
수행 하려고 하는 메뉴를 선택하세요(add : 1, remove : 2, debug : 3, exit : -1) : 3
[DEBUG] PRINT AVL TREE
Left:  (8,9)  Key: (13,1) Right:  (15,6)  Parent: null Balance: 0
Left:  (5,7)  Key: (8,9) Right: null Parent:  (13,1)  Balance: -1
Left: null Key: (5,7) Right: null Parent:  (8,9)  Balance: 0
Left:  (14,5)  Key: (15,6) Right: null Parent:  (13,1)  Balance: -1
Left: null Key: (14,5) Right: null Parent:  (15,6)  Balance: 0
수행 하려고 하는 메뉴를 선택하세요(add : 1, remove : 2, debug : 3, exit : -1) : -1
<<프로그램을 종료합니다>>
```

이 과제에서 필요한 객체는?

- AppView
- AppController
  - AVLTree
- AVLTree
  - AVLNode
- AVLNode

AppControllor의 공개 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - public void run()

AppView의 공개 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - public AppView()
  - public void outputMsg(String aString)
  - public String inputString()
  - public int inputInt()

AVLTree의 공개 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - public AVLNode<Key, Obj> root()
  - public void addKeyandObejct(Key aKey, Obj aObj)
  - public AVLNode<Key, Obj> removeObjectForKey(Key aKey)

AVLNode의 공개 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - public AVLNode(Key givenKey, Obj givenObj)
  - public AVLNode<Key, Obj> left()
  - public void setLeft(AVLNode<Key, Obj> aNode)
  - public AVLNode<Key, Obj> right()
  - public void setRight(AVLNode<Key, Obj> aNode)
  - public AVLNode<Key, Obj> parent()
  - public void setParent(AVLNode<Key, Obj> aNode)
  - public Key key()
  - public void setKey(Key aKey)
  - public Obj obj()
  - public void setObj(Obj aObj)
  - public int balance()
  - public void setBalance(int aBalance)
  - public int compareTo(Key givenKey)

Class “AppControllor“

AppControllor– 비공개 인스턴스 변수

```
public class AppControllor {
    private AppView _appView;
    private AVLTree<Integer, Integer> _dic;
```

AppControllor 의 공개 함수 run()의 구현

```
public void run() {
        _appView = new AppView();
        _dic = new AVLTree<Integer, Integer>();
        _appView.outputMsg(_appView.MSG_StartProgram);

        int command = 0;
        int aObject = 0;

        _appView.outputMsg(_appView.MSG_Menu);
        command = _appView.inputInt();
        while(command != -1)        {
                if(command == 1)      {
                        _appView.outputMsg(_appView.MSG_InputKey);
                        int aKey = _appView.inputInt();
                        aObject++;
                        _dic.addKeyandObejct(aKey, aObject);
                }
                else if(command == 2)       {
                        _appView.outputMsg(_appView.MSG_InputKey);
                        int aKey = _appView.inputInt();
                        _dic.removeObjectForKey(aKey);
                }
                else if(command == 3)       {
                        _appView.outputMsg(_appView.MSG_Debug);
                        _dic.showAVLTree(_dic.root());
                }
                _appView.outputMsg(_appView.MSG_Menu);
                command = _appView.inputInt();
            }
            _appView.outputMsg(_appView.MSG_EndProgram);
        }
```

Class “AppView“

AppView – 비공개 인스턴스 변수

```
import java.util.Scanner;

public class AppView {
    private Scanner _scanner;
```

AppView의 Public Method

- AppView 의 Public Member의 선언
  - public String MSG_StartProgram = "<<프로그램을 시작합니다>>₩n";
  - public String MSG_EndProgram = "<<프로그램을 종료합니다>>₩n";
  - public String MSG_Menu = "수행 하려고 하는 메뉴를 선택하세요(add : 1, remove : 2, debug : 3, exit : -1) : ";
  - public String MSG_InputKey = "Key를 입력하세요 : ";
  - public String MSG_Delete = "Key를 삭제합니다.₩n";
  - public String MSG_Debug = "[DEBUG] PRINT AVL TREE₩n";
  - public String MSG_Error = "[Error]₩n";

AppView의 Public Method

- AppView 의 Public Member function의 사용법과 구현
  - public AppView()
    - 생성자
  - public void outputMsg(String aString)
    - aString을 출력
  - public String inputString()
    - String을 하나 입력 받아 반환
  - public int inputInt()
    - Int를 하나 입력 받아 반환

Class “AVLTree”

비공개 인스턴스 변수

```
public class AVLTree<Key, Obj>{
    private AVLNode<Key, Obj> _root; // the root node
```

Public Method

- AVLTree 의 Public Member function의 사용법
  - public AVLNode<Key, Obj> root()
    - 현재 Tree의 Root를 전달 받는다.
  - public void addKeyandObejct(Key aKey, Obj aObj)
    - aKey와 aObj를 삽입
  - public AVLNode<Key, Obj> removeObjectForKey(Key aKey)
    - aKey를 삭제
  - public void showAVLTree(AVLNode aNode)
    - Debug용 AVL Tree 출력

Private Method

- AVLTree 의 Private Member function의 사용법
  - private void add(AVLNode<Key, Obj> aStartNode, AVLNode<Key, Obj> aNewNode)
    - aStartNode에 aNewNode를 AVLTree형으로 삽입
  - private void checkBalance(AVLNode<Key, Obj> aNode)
    - 밸런스를 체크하여 수정
  - private AVLNode<Key, Obj> searchRemoveNode(AVLNode aCurrentRoot, Key aKey)
    - aCurrentRoot에서 aKey를 검색하여 삭제 실행
  - private AVLNode<Key, Obj> remove(AVLNode aNode)
    - aRemoveNode 삭제
  - private void rebalance(AVLNode aNode)
    - Balance를 다시 계산하여 저장

Private Method

- AVLTree 의 Private Member function의 사용법
  - private AVLNode rotateLeft(AVLNode aNode)
    - aNode를 Left rotation
  - private AVLNode rotateRight(AVLNode aNode)
    - aNode를 right rotation
  - private AVLNode rotateLeftRight(AVLNode aNode)
    - leftNode를 rotation 실행 한 뒤 rightRotation을 실행
  - private AVLNode rotateRightLeft(AVLNode aNode)
    - rightNode를 rotation 실행 한 뒤 leftRotation을 실행
  - private AVLNode successor(AVLNode aNode)
    - 트리에서 주어진 노드의 successor를 반환
    - Successor : 트리를 어떤 순서로 순회할 때 한 노드를 방문한 후 바로 다음에 방문할 노드
  - private int size(AVLNode aNode)
    - Balance를 설정

Public Method

- AVLTree 의 Public Member function의 구현
  - public AvlNode<Key, Obj> root()
    - _root를 반환
  - public void addKeyandObejct(Key aKey, Obj aObj)
    - aKey와 aObj를 가지는 삽입하려는 AVLNode newNode를 생성
    - Recursive로 삽입을 하는 add를 호출
  - public AvlNode<Key, Obj> removeObjectForKey(Key aKey)
    - aKey를 가진 Node를 찾고 삭제하는 함수를 호출하는 searchRemoveNode함수를 호출

Public Method

- AVLTree 의 Public Member function의 구현

```
public void showAVLTree(AVLNode aNode) {
    System.out.print("Left: " );
    if (aNode.left() != null) {
        System.out.print(" (" + aNode.left().key() + "," + aNode.left().obj() + ") ");
    }
    else
        System.out.print("null");
    System.out.print(" Key: (" + aNode.key() +"," + aNode.obj() + ") Right: ");
    if (aNode.right() != null) {
        System.out.print(" (" + aNode.right().key() + "," + aNode.right().obj() + ") ");
    }
    else
        System.out.print("null");
    System.out.print(" Parent: ");
    if (aNode.parent() != null) {
        System.out.print(" (" + aNode.parent().key() + "," + aNode.parent().obj() + ") ");
    }
    else
        System.out.print("null");
    System.out.println(" Balance: " + aNode.balance());
    if (aNode.left() != null) {
        showAVLTree(aNode.left());
    }
    if (aNode.right() != null) {
        showAVLTree(aNode.right());
    }
}
```

Private Method

- AVLTree 의 Private Member function의 구현

```
private void add(AVLNode<Key, Obj> aStartNode, AVLNode<Key, Obj> aNewNode) {
    if (aStartNode == null) {
        this._root = aNewNode;
    } else {
        if (aNewNode.compareTo(aStartNode.key()) < 0) {
            if (aStartNode.left() == null) {
                aStartNode.setLeft(aNewNode);
                aNewNode.setParent(aStartNode);
                checkBalance(aStartNode);
            } else {
                add(aStartNode.left(), aNewNode);
            }
        } else if (aNewNode.compareTo(aStartNode.key()) > 0) {
            if (aStartNode.right() == null) {
                aStartNode.setRight(aNewNode);
                aNewNode.setParent(aStartNode);
                checkBalance(aStartNode);
            } else {
                add(aStartNode.right(), aNewNode);
            }
        } else {
            // do nothing: This node already exists
        }
    }
}
```

Private Method

- AVLTree 의 Private Member function의 구현

```
private void checkBalance(AVLNode<Key, Obj> aNode) {
    rebalance(aNode);
    int balance = aNode.balance();
    if (balance == -2) {

        if (size(aNode.left().left()) >= size(aNode.left().right())) {
            aNode = rotateRight(aNode);
        } else {
            aNode = rotateLeftRight(aNode);
        }
    } else if (balance == 2) {
        if (size(aNode.right().right()) >= size(aNode.right().left())) {
            aNode = rotateLeft(aNode);
        } else {
            aNode = rotateRightLeft(aNode);
        }
    }
    if (aNode.parent() != null) {
        checkBalance(aNode.parent());
    } else {
        this._root = aNode;
    }
}
```

Private Method

- AVLTree 의 Private Member function의 구현

```
private AVLNode<Key, Obj> searchRemoveNode(AVLNode aCurrentRoot, Key aKey) {
    if (aCurrentRoot == null) {
        return null;
    } else {
        if (aCurrentRoot.compareTo(aKey) > 0) {
            return searchRemoveNode(aCurrentRoot.left(), aKey);
        } else if (aCurrentRoot.compareTo(aKey) < 0) {
            return searchRemoveNode(aCurrentRoot.right(), aKey);
        } else if (aCurrentRoot.compareTo(aKey) == 0) {
            return remove(aCurrentRoot);
        }
    }
    return null;
}
```

Private Method

- AVLTree 의 Private Member function의 구현

```
private AVLNode rotateLeft(AVLNode aNode) {

    AVLNode v = aNode.right();
    v.setParent(aNode.parent());

    aNode.setRight(v.left());

    if (aNode.right() != null) {
        AVLNode aR = aNode.right();
        aR.setParent(aNode);
    }

    v.setLeft(aNode);
    aNode.setParent(v);

    if (v.parent() != null) {
        AVLNode vP = v.parent();
        if (vP.right() == aNode) {
            vP.setRight(v);
        } else if (vP.left() == aNode) {
            vP.setLeft(v);
        }
    }

    rebalance(aNode);
    rebalance(v);

    return v;
}
```

Private Method

- AVLTree 의 Private Member function의 구현

```
private AVLNode successor(AVLNode aNode) {
    if (aNode.right() != null) {
        AVLNode r = aNode.right();
        while (r.left() != null) {
            r = r.left();
        }
        return r;
    } else {
        AVLNode p = aNode.parent();
        while (p != null && aNode == p.right()) {
            aNode = p;
            p = aNode.parent();
        }
        return p;
    }
}
```

Private Method

- AVLTree 의 Private Member function의 구현

```
private int size(AVLNode aNode) {
    if (aNode == null) {
        return -1;
    }
    if (aNode.left() == null && aNode.right() == null) {
        return 0;
    } else if (aNode.left() == null) {
        return 1 + size(aNode.right());
    } else if (aNode.right() == null) {
        return 1 + size(aNode.left());
    } else {
        return 1 + Math.max(size(aNode.left()), size(aNode.right()));
    }
}
```

Private Method

- AVLTree의 Member function의 구현
  - 아래의 함수들은 직접 구현해보도록 한다.
  - private void rebalance(AVLNode aNode)
    - aNode의 right의 size에서 aNode의 left의 size를 뺀 값을 aNode의 balance에 저장
  - private AVLNode<Key, Obj> remove(AVLNode aNode)
    - aRemoveNode 삭제

Private Method

- AVLTree의 Member function의 구현
  - 아래의 함수들은 직접 구현해보도록 한다.
  - private AVLNode rotateRight(AVLNode aNode)
    - aNode를 right rotation
  - private AVLNode rotateLeftRight(AVLNode aNode)
    - leftNode를 rotation 실행 한 뒤 rightRotation을 실행
  - private AVLNode rotateRightLeft(AVLNode aNode)
    - rightNode를 rotation 실행 한 뒤 leftRotation을 실행
  - public AVLTree getLeft()
  - public AVLTree getRight()

Class “AVLNode”

비공개 인스턴스 변수

```
public class AVLNode <Key, Obj>{
    private AVLNode<Key, Obj> _left;
    private AVLNode<Key, Obj> _right;
    private AVLNode<Key, Obj> _parent;
    private Key _key;
    private Obj _obj;
    private int _balance;
```

Public Method

- AVLTree 의 Public Member function의 사용법
  - public AVLNode(Key givenKey, Obj givenObj)
  - public AVLNode<Key, Obj> left()
  - public void setLeft(AVLNode<Key, Obj> aNode)
  - public AVLNode<Key, Obj> right()
  - public void setRight(AVLNode<Key, Obj> aNode)
  - public AVLNode<Key, Obj> parent()
  - public void setParent(AVLNode<Key, Obj> aNode)
  - public Key key()
  - public void setKey(Key aKey)
  - public Obj obj()
  - public void setObj(Obj aObj)
  - public int balance()
  - public void setBalance(int aBalance)
  - public int compareTo(Key givenKey)

[문제 13] 요약

- AVL Tree를 구현해보고 실제 Tree를 위한 메소드들을 추가하여 테스트해본다.

- 각 요약에 대한 내용을 보고서에 작성하여 제출하세요.

과제 제출

과제 제출

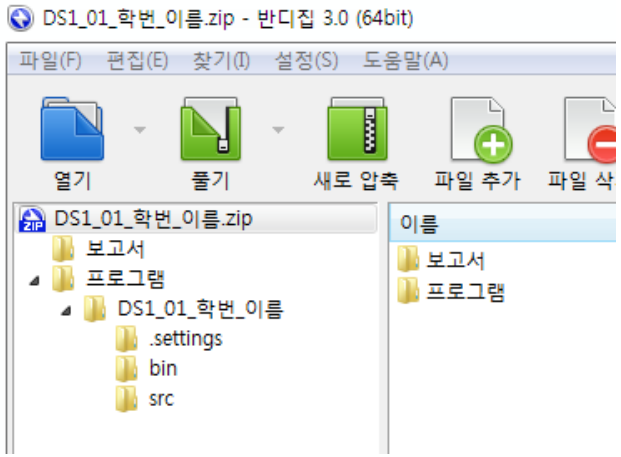
- pineai@cnu.ac.kr
  - 메일 제목 : [0X]DS2_13_학번_이름
    - 양식에 맞지 않는 메일 제목은 미제출로 간주됨
    - 앞의 0X는 분반명 (오전10시 : 00반 / 오후4시 : 01반)

- 제출 기한
  - 12월 15일(일) 23시59분까지
  - 시간 내 제출 엄수
  - 모든 실습 과제는 12월 22일까지 제출되어야함

  - 제출을 하지 않을 경우 0점 처리하고, 숙제를 50% 이상 제출하지 않으면 F 학점 처리하며, 2번 이상 제출하지 않으면 A 학점을 받을 수 없다.

과제 제출

- 파일 이름 작명 방법
  - DS2_13_학번_이름.zip
  - 폴더의 구성
    - DS2_13_학번_이름
      - 프로그램
        - 프로젝트 폴더 / 소스
        - 메인 클래스 이름 : DS2_13_학번_이름.java
      - 보고서
        - 이곳에 보고서 문서 파일을 저장한다.
        - 입력과 실행 결과는 화면 image로 문서에 포함시킨다.
        - 문서는 pdf 파일로 만들어 제출한다.



보고서 작성 방법

- 겉장
  - 제목: 자료구조 실습 보고서
  - [제xx주] 숙제명
  - 제출일
  - 학번/이름
- 내용
  1. 프로그램 설명서
     1. 주요 알고리즘 /자료구조 /기타
     2. 함수 설명서
     3. 종합 설명서 : 프로그램 사용방법 등을 기술
  2. 구현 후 느낀 점 : 요약의 내용을 포함하여 작성한다.
  3. 실행 결과 분석
     1. 입력과 출력 (화면 capture : 실습예시와 다른 예제로 할 것)
     2. 결과 분석

     -------- 표지 제외 3장 이내 작성 --------
  4. 소스코드 : 화면 capture가 아닌 소스를 붙여넣을 것

     소스는 장수 제한이 없음.

[제 13 주 실습] 끝