[제 9 주 실습]

허프만 코드를 이용한 파일 압축

강 지 훈
jhkang@cnu.ac.kr

충남대학교 컴퓨터공학과

[문제 9]
Huffman Code

문제 개요

- 허프만 알고리즘을 구현하고, 압축을 실행하여 본다.

Huffman 알고리즘

- 무손실 데이터 압축 알고리즘
- 20%~90%로 데이터를 효율적으로 압축하는데 사용
- 데이터를 일련의 문자로 간주
- 문자가 발생하는 빈도 테이블을 이용하여 최적의 방법을 구축

Huffman 코드

- 허프만의 알고리즘을 통해 나온 최적 이진 문자 코드
  - 가장 많이 나오는 코드를 짧은 코드로 대체
  - 디코딩 과정이 필요 (디코딩 테이블)

Character	Frequency	C1(Fixed Length)	C2	C3(Huffman)
a	16	000	10	00
b	5	001	11110	1110
c	12	010	1110	110
d	17	011	110	01
e	10	100	11111	1111
f	25	101	0	10

- **Example 4.7: C3 is the best of three**
  - **Bits(C1) = 16(3) + 5(3) + 12(3) + 17(3) + 10(3) + 25(3) = 255**
  - **Bits(C2) = 16(2) + 5(5) + 12(4) + 17(3) + 10(5) + 25(1) = 231**
  - **Bits(C3) = 16(2) + 5(4) + 12(3) + 17(2) + 10(4) + 25(2) = 212**

60

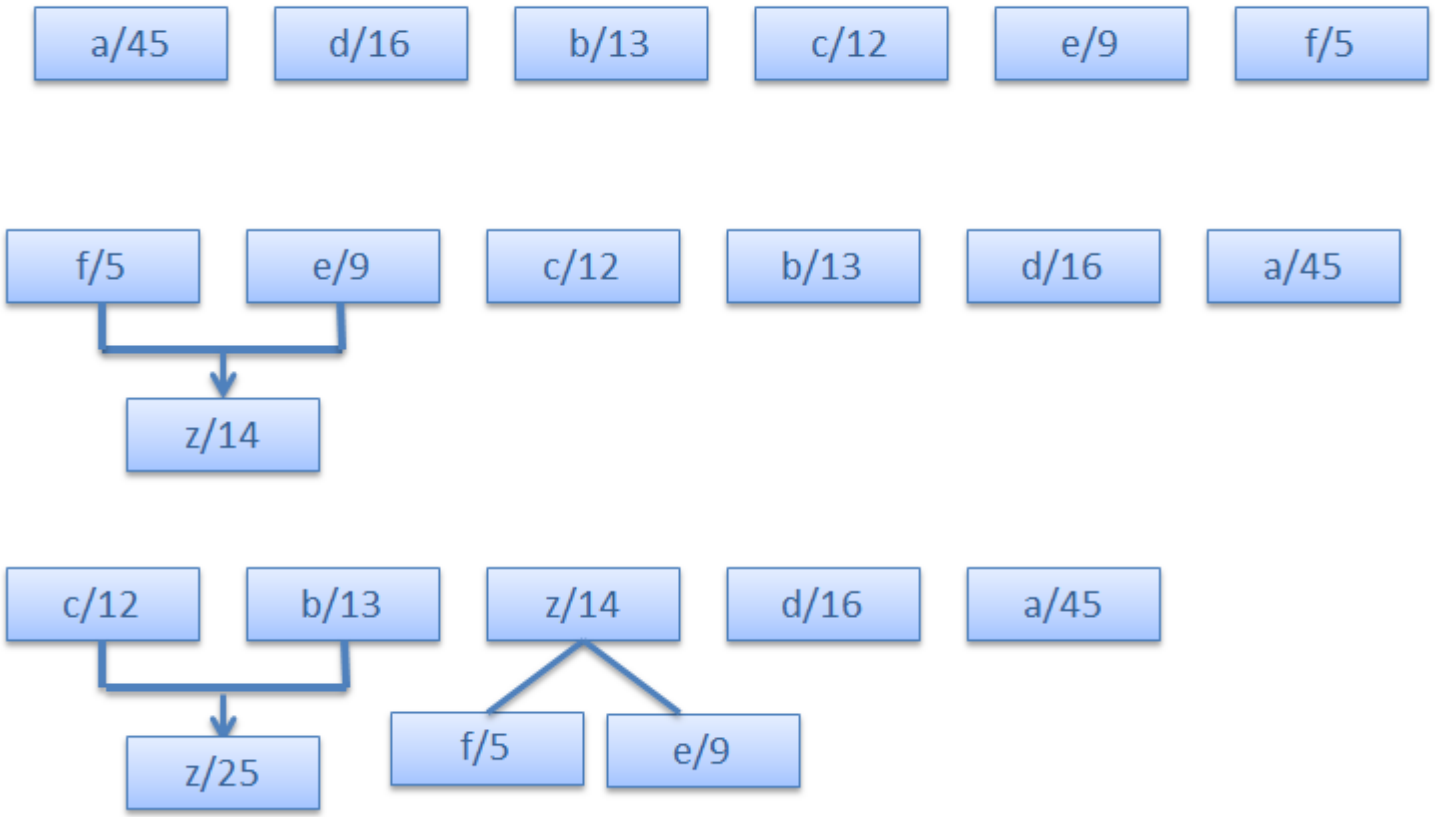
Huffman Tree

- http://en.wikipedia.org/wiki/File:Huffman_huff_demo.gif

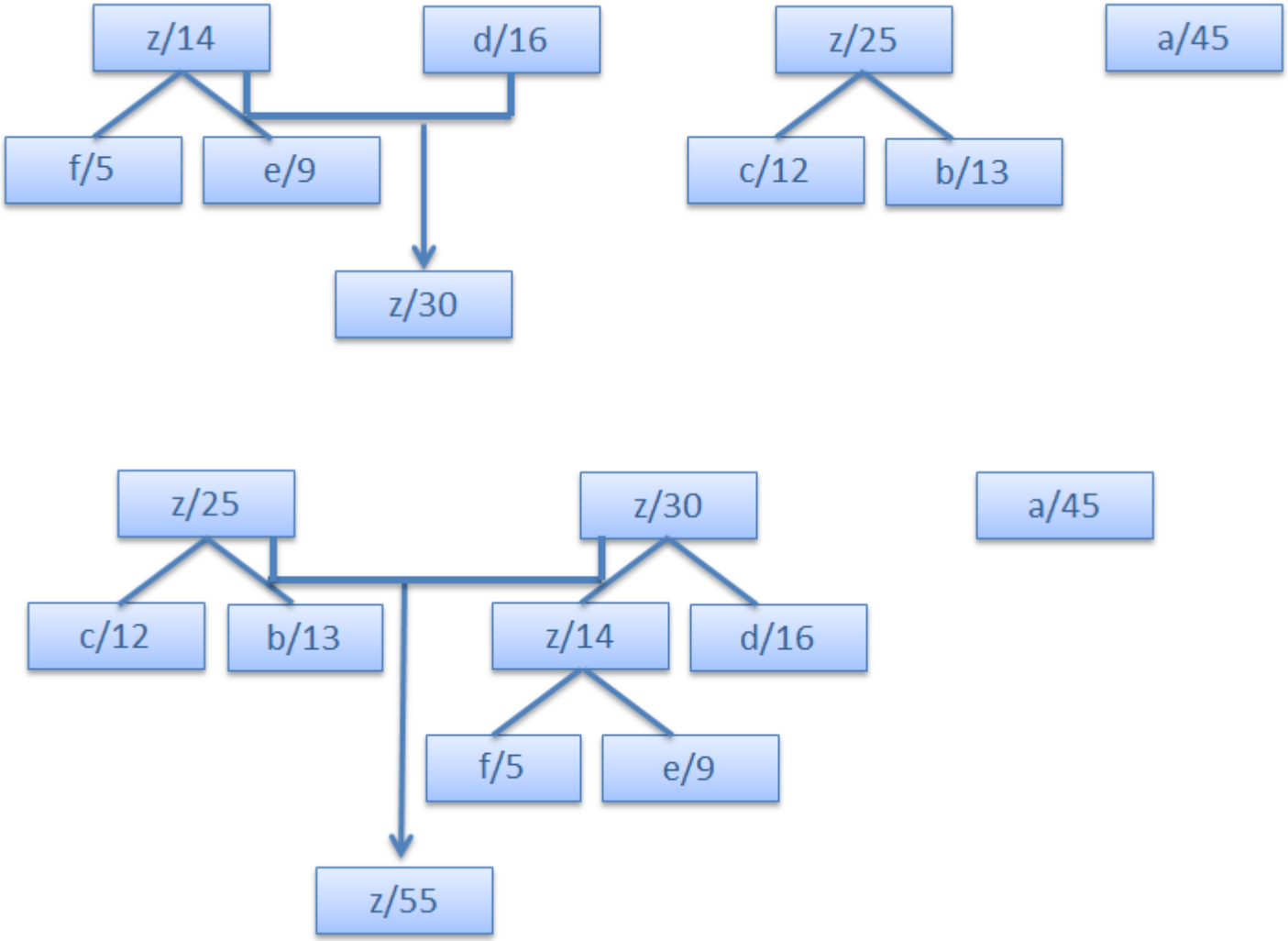
File :

b	p	`	m	j	o	d	a	i	r	u	l	s	e	
1	1	2	2	3	3	3	4	4	5	5	6	6	8	12

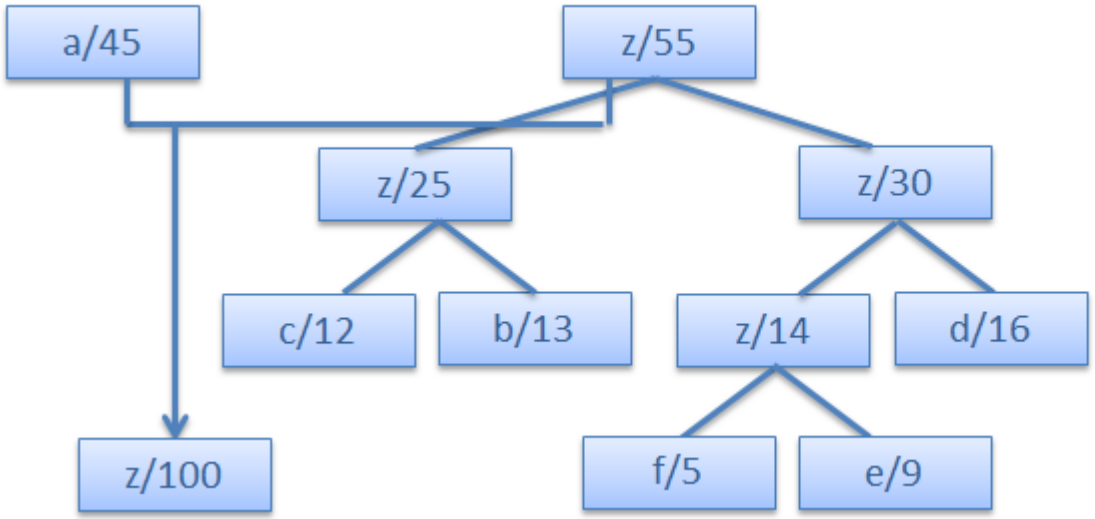
Huffman Tree



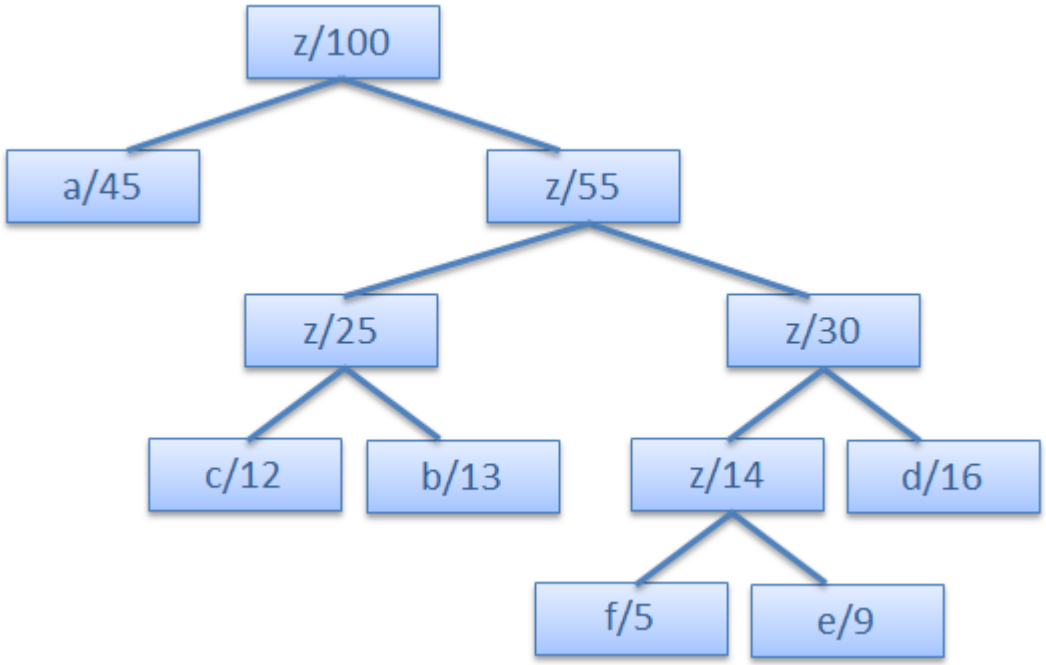
Huffman Tree



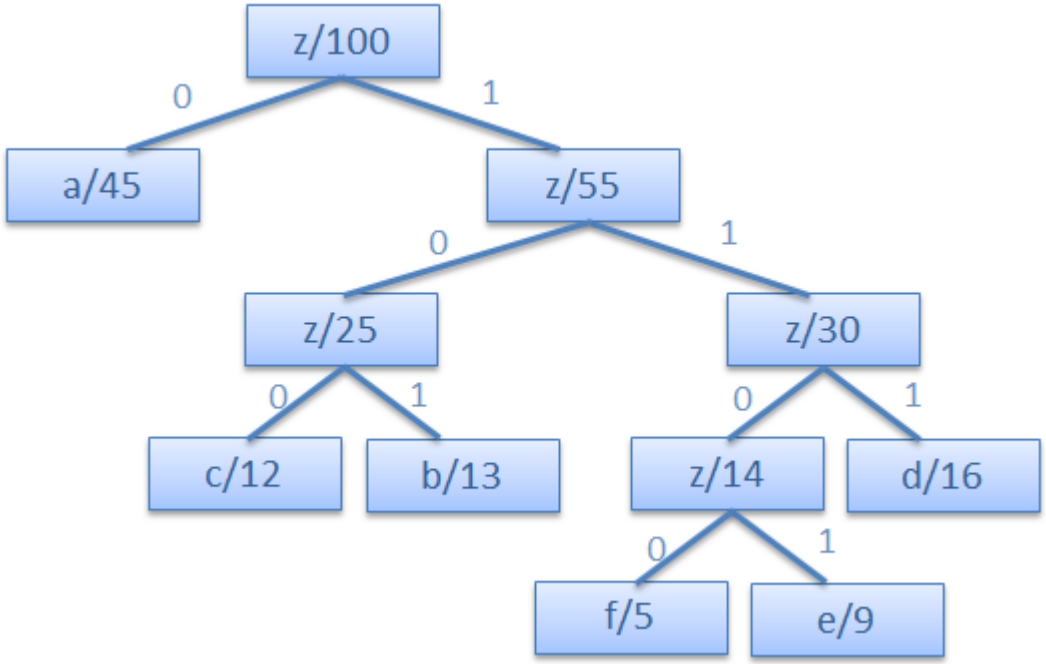
Huffman Tree



Huffman Tree



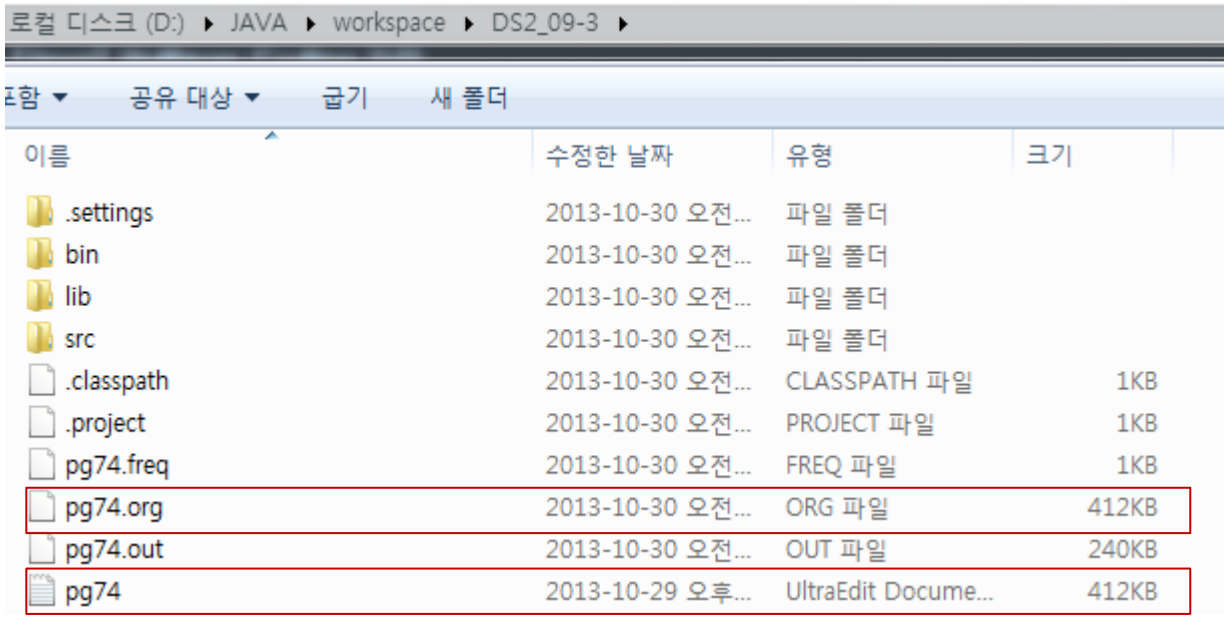
Huffman Code



Character	Variable-length codeword	Size of string in bit
a	0	1
b	101	3
c	100	3
d	111	3
e	1101	4
f	1100	4

❑출력화면

```
수행 하려고 하는 메뉴를 선택하세요(encode : 1, decode : 2, exit : -1) :1
압축 할 파일 이름을 입력하세요 : pg74.txt
빈도 파일 이름을 입력하세요 : pg74.freq
압축 파일 이름을 입력하세요 : pg74.out
Huffman Code Encoding 처리 완료 pg74.txt > pg74.out
수행 하려고 하는 메뉴를 선택하세요(encode : 1, decode : 2, exit : -1) :2
압축을 해제할 파일 이름을 입력하세요 : pg74.out
빈도 파일 이름을 입력하세요 : pg74.freq
압축 해제 후 저장할 파일 이름을 입력하세요 : pg74.org
Huffman Code decoding 처리 완료pg74.out > pg74.org
수행 하려고 하는 메뉴를 선택하세요(encode : 1, decode : 2, exit : -1) :
```



이 과제에서 필요한 객체는?

- AppView
- AppController
  - Huffman
- Huffman
  - MinPriorityQ
  - HuffmanTree
    - HuffmanTreeNode
  - HuffmanInputStream
  - HuffmanOutputStream
- HuffmanInputStream
- HuffmanOutputStream
- MinPriorityQ
- HuffmanTree
- HuffmanTreeNode

AppControllor의 공개 함수는?

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

AppView의 공개 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - public AppView()
  - public void showInputSelectMenu()
  - public void showInputEncodingOrignalFileName()
  - public void showInputEncodingResultFileName()
  - public void showInputDecodingOrignalFileName()
  - public void showInputDecodingResultFileName()
  - public void showInputFrequencyFileName()
  - public void showErrorMsg()
  - public void showResultEncoding(String originalFileName, String resultFileName)
  - public void showResultDecoding(String originalFileName, String resultFileName)
  - public String inputString()
  - public int inputInt()

Huffman의 멤버 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - public void EnCoding(String originalFileName, String frequencyFileName, String resultFileName)
  - public void Decoding(String originalFileName, String frequencyFileName, String resultFileName)

HuffmanTree의 멤버 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - public HuffmanTree()
  - public int ERROR()
  - public int NOT_A_VALUE()
  - public int EOF()
  - public void charCount(InputStream in)
  - private void buildTree()
  - public String getCode(int key)
  - public int getKey(String code)
  - public void writeFrequency(DataOutputStream out)
  - public void readFrequency(DataInputStream in)

HuffmanTreeNode의 멤버 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - public HuffmanTreeNode(int v, int c, HuffmanTreeNode p, HuffmanTreeNode l, HuffmanTreeNode r)
  - public int value()
  - public int count()
  - public HuffmanTreeNode parent()
  - public HuffmanTreeNode left()
  - public HuffmanTreeNode right()
  - public void setParent(HuffmanTreeNode aParent)
  - public int compareTo(Object receivedArg)

MinPriorityQ의 멤버 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - [실습 3]에서 사용한 MinPriorityQ를 Comparator를 사용하도록 수정
  - public MinPriorityQ()
  - public MinPriorityQ(int givenMaxsize)
  - public boolean isEmpty()
  - public boolean isFull()
  - public int size()
  - public boolean add(Comparable aComparable)
  - public Key deleteMin()

Class “AppControllor“

AppControllor– 비공개 인스턴스 변수

```
public class AppControllor {
    private AppView _appView;
    private Huffman _huffman;
```

AppControllor 의 공개 함수 run()의 구현

```
public void run() {
    int command = 0;
    this._appView = new AppView();
    this._huffman = new Huffman();

    while (command != -1) {
        try {
            this._appView.showInputSelectMenu();
            command = this._appView.inputInt();

            if(command == 1){
                this._appView.showInputEncodingOrignalFileName();
                String originalFileName = this._appView.inputString();
                this._appView.showInputFrequencyFileName();
                String frequencyFileName = this._appView.inputString();
                this._appView.showInputEncodingResultFileName();
                String resultFileName = this._appView.inputString();

                this._huffman.EnCoding(originalFileName, frequencyFileName, resultFileName);
                this._appView.showResultEncoding(originalFileName, resultFileName);
            }
```

AppControllor 의 공개 함수 run()의 구현

```
            else if(command == 2)
            {
                this._appView.showInputDecodingOrignalFileName();
                String originalFileName = this._appView.inputString();
                this._appView.showInputFrequencyFileName();
                String frequencyFileName = this._appView.inputString();
                this._appView.showInputDecodingResultFileName();
                String resultFileName = this._appView.inputString();

                this._huffman.Decoding(originalFileName, frequencyFileName, resultFileName);
                this._appView.showResultDecoding(originalFileName, resultFileName);
            }
            else if(command == -1)
                break;
            else
                this._appView.showErrorMsg();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Class “AppView“

AppView – 비공개 인스턴스 변수

```
import java.util.Scanner;

public class AppView {
    private Scanner _scanner;
```

AppView의 Public Method

- AppView 의 Public Member function의 사용법과 구현
  - public AppView()
    - 생성자
  - public void showInputSelectMenu()
    - "수행 하려고 하는 메뉴를 선택하세요(encode : 1, decode : 2, exit : -1) :" 를 화면에 내보낸다.
  - public void showInputEncodingOrignalFileName()
    - "압축 할 파일 이름을 입력하세요 : " 를 화면에 내보낸다.
  - public void showInputEncodingResultFileName()
    - "압축 파일 이름을 입력하세요 : " 를 화면에 내보낸다.
  - public void showInputDecodingOrignalFileName()
    - "압축을 해제할 파일 이름을 입력하세요 : " 를 화면에 내보낸다.
  - public void showInputDecodingResultFileName()
    - "압축 해제 후 저장할 파일 이름을 입력하세요 : " 를 화면에 내보낸다.

AppView의 Public Method

- AppView 의 Public Member function의 사용법과 구현
  - public void showInputFrequencyFileName()
    - “빈도 파일 이름을 입력하세요 : ” 를 화면에 내보낸다.
  - public void showErrorMsg()
    - “메뉴에 맞는 입력을 해주세요.” 를 화면에 내보낸다.
  - public void showResultEncoding(String originalFileName, String resultFileName)
    - “Huffman Code Encoding 처리 완료 " + originalFileName + " > " + resultFileName를 화면에 내보낸다.
  - public void showResultDecoding(String originalFileName, String resultFileName)
    - “Huffman Code decoding 처리 완료" + originalFileName + " > " + resultFileName를 화면에 내보낸다.

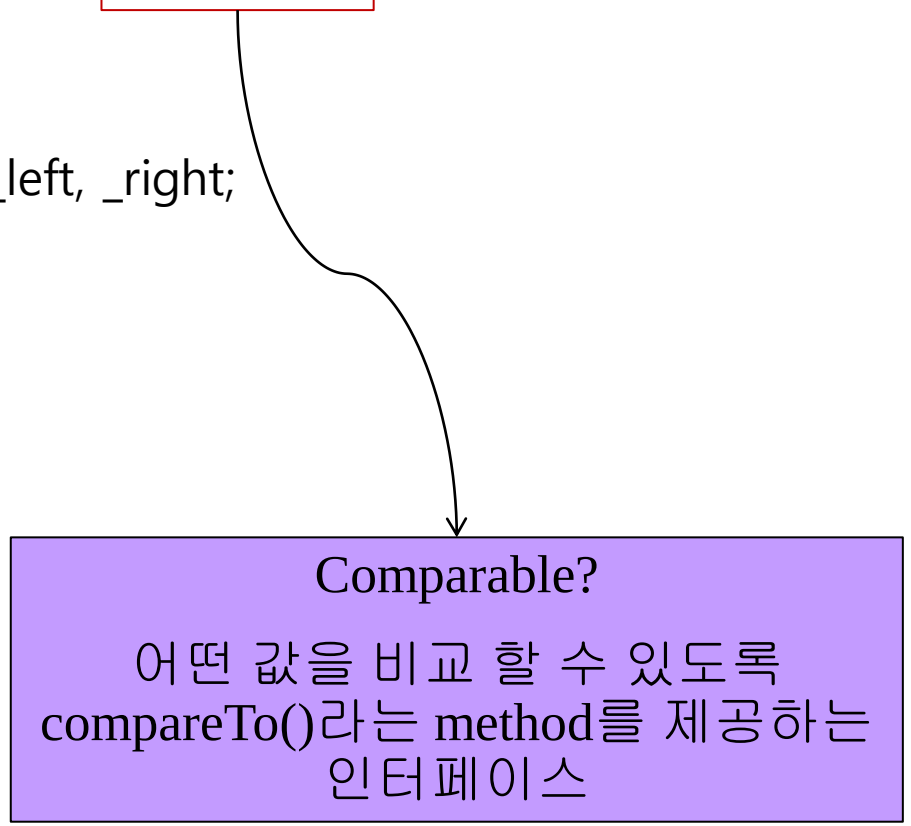
AppView의 Public Method

- AppView 의 Public Member function의 사용법과 구현
  - public String inputString()
    - 문자열을 입력 받아 반환한다.
    - _scanner의 nextLine()등을 이용한다.
  - public int inputInt()
    - 정수를 입력 받아 반환한다.
    - _scanner의 nextLine()를 이용한다. > why?

Class “HuffmanTreeNode”

비공개 인스턴스 변수

```
public class HuffmanTreeNode implements Comparable {
    private int _value;
    private int _count;
    private HuffmanTreeNode _parent, _left, _right;
```



"Comparable" interface

- compareTo() 메소드를 재정의
  - 어떤 값들의 비교 결과를 크다(1), 같다(0), 작다(-1)와 같은 정수형 값을 반환
- 배열로 사용될 때, Arrays.sort() 메소드를 이용하여, 자동 정렬이 가능
  - Arrays.sort() : 배열을 자동 정렬 시켜주는 기능
- 이번 실습에서는 compareTo가 구현 되어 있는 HuffmanTreeNode를 사용

HuffmanTreeNode 의 Public Method

- HuffmanTreeNode 의 Public Member function의 구현
  - public HuffmanTreeNode(int v, int c, HuffmanTreeNode p, HuffmanTreeNode l, HuffmanTreeNode r)
    - 생성자
  - public int value()
    - _value를 반환
  - public int count()
    - _count를 반환
  - public HuffmanTreeNode parent()
    - _parent를 반환
  - public HuffmanTreeNode left()
    - _left를 반환
  - public HuffmanTreeNode right()
    - _right를 반환

HuffmanTreeNode 의 Public Method

- HuffmanTreeNode 의 Public Member function의 구현
  - public void setParent(HuffmanTreeNode aParent)
    - aParent를 _parent에 저장
  - public int compareTo(Object receivedArg)
    - 오브젝트끼리 비교하는 함수
    - HuffmanTreeNode의 count끼리 비교한다

```
public int compareTo(Object receivedArg)
{
    return _count - ((HuffmanTreeNode) receivedArg)._count;
}
```

Class “MinPriorityQ”

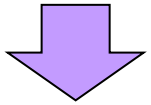
MinPriorityQ– 비공개 인스턴스 변수

```
public class MinPriorityQ {
    private int        _size;
    private int        _maxSize;
    private Comparable[]    _heap;
```

MinPriorityQ 의 Public Method

- MinPriorityQ 의 Public Member function의 수정
  - 기존의 소스에서 비교하는 부분을 compareTo를 사용하도록 수정

```
if ( (smallerChild < this._size) &&
        (this._heap[smallerChild].cost() > this._heap[smallerChild+1].cost())) {
    // right child가 존재하고, 그 cost 값이 더 작으므로,
    // right child를 smallerChild로 한다.
    smallerChild++ ;
}
```



```
if ( (smallerChild < _size) &&
        (this._heap[smallerChild].compareTo(this._heap[smallerChild + 1]) > 0) )
    // right child가 존재하고, 그 cost 값이 더 작으므로,
    // right child를 smallerChild로 한다.
    smallerChild++;
```

Class “HuffmanTree”

Member Functions의 구현

- HuffmanTree의 Public Member function의 사용법
  - public HuffmanTree()
    - 생성자
  - public void charCount(InputStream in)
    - 글자수를 카운트 하여 트리를 생성
  - public int ERROR()
    - ERROR로 정의된 값을 반환
  - public int NOT_A_VALUE()
    - NOT_A_VALUE로 정의 된 값을 반환
  - public int EOF()
    - EOF로 정의 된 값을 반환

Member Functions의 구현

- HuffmanTree의 Public Member function의 사용법
  - public String getCode(int key)
    - Key를 전달 받아 Code를 찾음
  - public int getKey(String code)
    - Code를 전달 받아 Key를 찾음
  - public void writeFrequency(DataOutputStream out)
    - Frequency Tree를 Frequency 파일에 write
  - public void readFrequency(DataInputStream in)
    - Frequency파일을 읽어 Frequency Tree로 생성

HuffmanTree – 비공개 인스턴스 변수

```
import java.io.*;

public class HuffmanTree {
    private int NOT_A_VALUE = -2;
    private int ERROR = -3;
    private int EOF = 255;
    private HuffmanTreeNode _root = null;
    private int[] _count;
    private HuffmanTreeNode[] _huffmanTreeNodes;
```

Member Functions의 구현

- HuffmanTree의 Private Member function의 사용법
  - private void buildTree()
    - 트리를 생성

자바의 예외처리

- 프로그램 실행도중 예외가 발생하여 프로그램이 중단되는 것을 미리 예방하는 것
- 예외가 발생한 부분은 어쩔 수 없어도 이외의 영역은 끝까지 수행 될 수 있도록 함

Object - Throwable - Error
- Exception - IOException
- ClassNotFoundException
- .....
- RuntimeException
- ArithmeticException
- ClassCastException
- NullPointerException
....
- IndexOutOfBoundsException

자바의 예외처리

Throws

- 호출한 곳에 예외처리를 전달 할 때 사용하는 방법
- 함수 내에서는 처리하지 않고 실제 이러한 함수를 출력한 부분에서 try-catch구문을 통해 예외처리를 함

```
public void charCount(InputStream in) throws IOException {

    int buf = in.read(); // 한 글자를 읽어서
    while (buf != -1) {
        _count[buf]++; // 그 글자의 카운트값 1 플러스
        buf = in.read();
    }

    buildTree(); // 카운트를 가지고 트리를 만듬
}
```

Member Functions의 구현

- HuffmanTree의 Public Member function의 구현
  - public HuffmanTree()
    - _count를 int형 배열 256개 저장 할 수 있도록 생성 > Why?
  - public void charCount(InputStream in) throws IOException
    - 값이 -1이 아닐 경우(파일의 끝일 경우 -1이 반환)까지 계속 읽어서
    - _count에 있는 해당 값의 위치를 증가
  - public String getCode(int key)
    - _huffmanTreeNodes의 Key 번째의 값을 current로 저장
    - current가 null이면 null을 반환
    - 값을 저장할 String형 code를 선언
    - Null이 아닐 동안 계속 확인
      - current의 parent부터 parent를 확인하며
      - Left일 경우 0, right일 경우 1을 code에 저장
    - code를 반환

Member Functions의 구현

- HuffmanTree의 Public Member function의 구현
  - public int getKey(String code)
    - 맨 위 node인 _root를 임시 HuffmanTreeNode searchNode에 저장
    - code의 length만큼 돌면서
      - searchNode가 null이면 Error를 반환
      - Code의 값이 0이면 왼쪽
      - 1이면 오른쪽으로 이동
    - 결과값으로 얻어진 searchNode의 value를 반환

Member Functions의 구현

HuffmanTree의 Public Member function의 구현

```
public void writeFrequency(DataOutputStream out) throws IOException
{
    for (int i = 0; i < _count.length; i++){
        if (_count[i] > 0)
        {
            out.writeByte(i);
            // 글자값을 byte로 쓰기
            out.writeInt(_count[i]);
            // 그 글자의 count 값을 int(4byte)로 쓰기
        }
    }
    out.writeByte(0);
    out.writeInt(0); // 마지막 확인용
}
public void readFrequency(DataInputStream in) throws IOException
{
    int key = in.readUnsignedByte();
    int c = in.readInt();
    while (c != 0)
    {
        _count[key] = c;
        // 현재 글자의 카운트 값 저장
        key = in.readUnsignedByte();
        c = in.readInt();
    }
    buildTree();
}
```

Why?

Member Functions의 구현

- HuffmanTree의 private Member function의 구현
  - private void buildTree() [1]
    - _huffmanTreeNodes 를 256개의 HuffmanTreeNode배열로 생성
    - MinPriorityQ인 _minPriorityQ를 생성
    - _count.length만큼 확인하며 _count[i]가 0보다 큰 경우
      - 새로운 HuffmanTreeNode를 생성(데이터만 존재-없는 링크는 null)하여 _huffmanTreeNodes[i]에 저장
      - _huffmanTreeNodes[i]를 _minPriorityQ 에 삽입
    - EOF도 처리하기 위하여 아래의 문장을 추가
      - _minPriorityQ.insert(_huffmanTreeNodes[EOF] = new HuffmanTreeNode(EOF, 1, null, null, null));

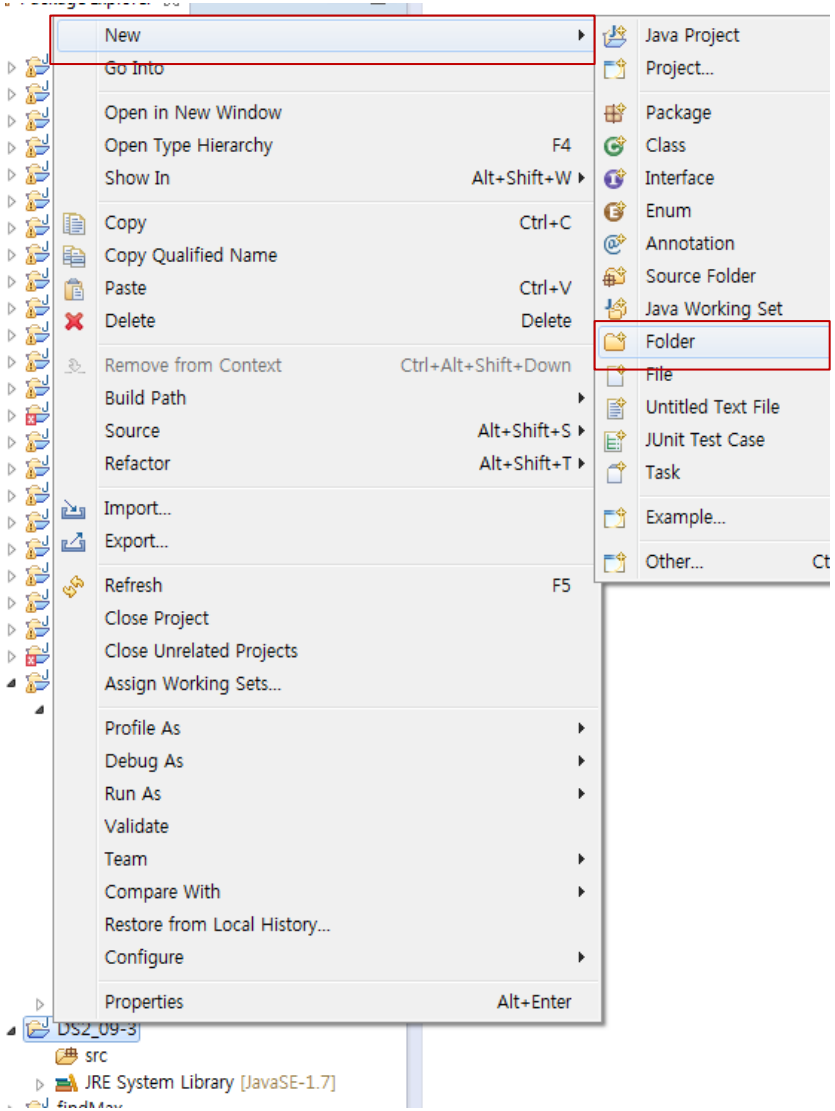
Member Functions의 구현

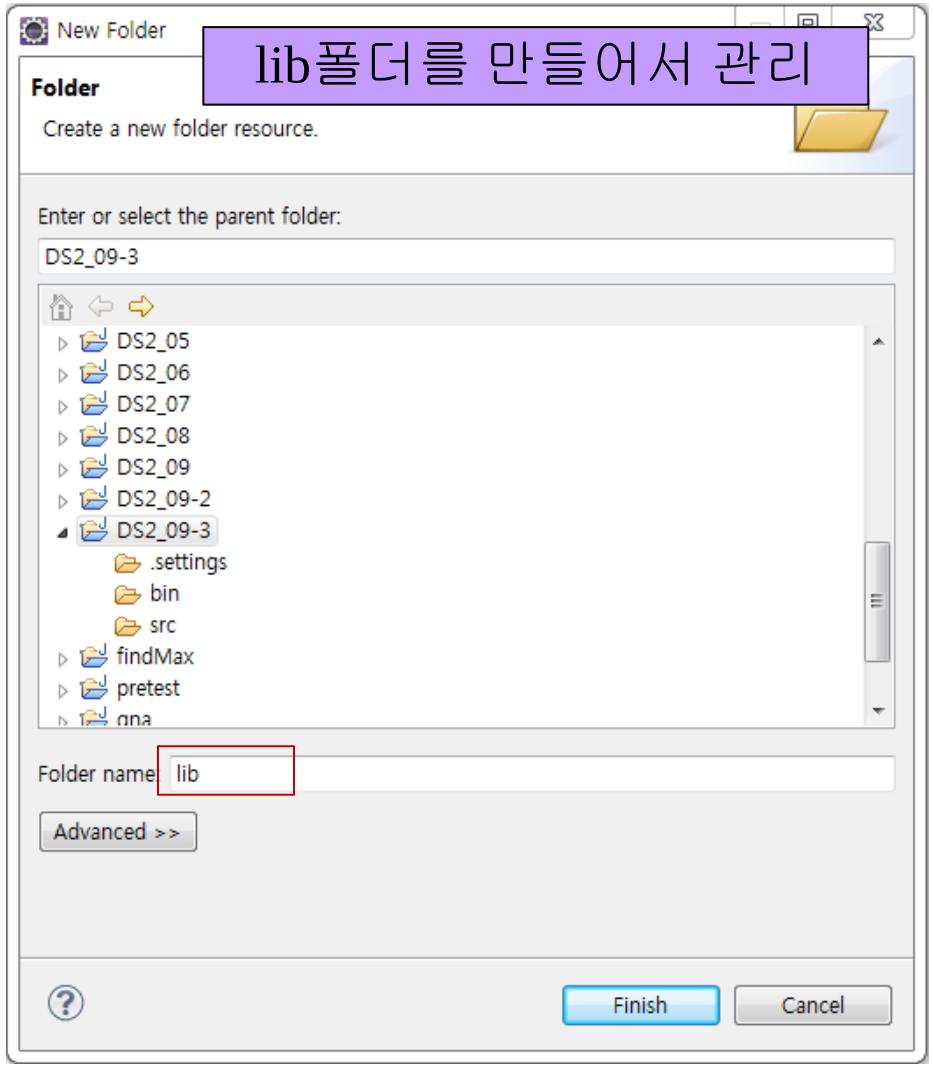
- HuffmanTree의 private Member function의 구현
  - private void buildTree() [2]
    - _minPriorityQ에서 최소 값(removeMin())을 읽음
    - _minPriorityQ이 비어 있지 않는 동안
      - _minPriorityQ에서 최소값(_minPriorityQ)을 얻어와서
      - 이러한 최소값을 가지는 좌,우 링크를 가지는 새 노드(value는 NOT_A_VALUE로 저장)를 만듬
      - 좌,우 링크의 parent를 지정(setParent())
      - 새 노드를 _minPriorityQ에 삽입
      - _minPriorityQ에서 최소값(removeMin())을 읽음
    - 가장 마지막에 있는 node를 _root에 저장

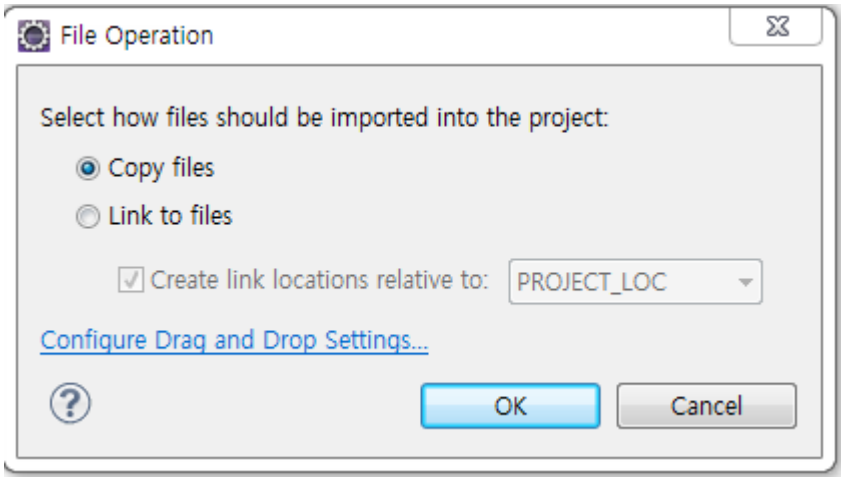
Jar “HuffmanIOStream”

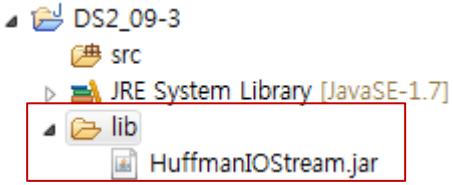
Jar file의 사용

- Class “HuffmanInputStream”,
  Class “HuffmanOutputStream”은 Jar file로 제공

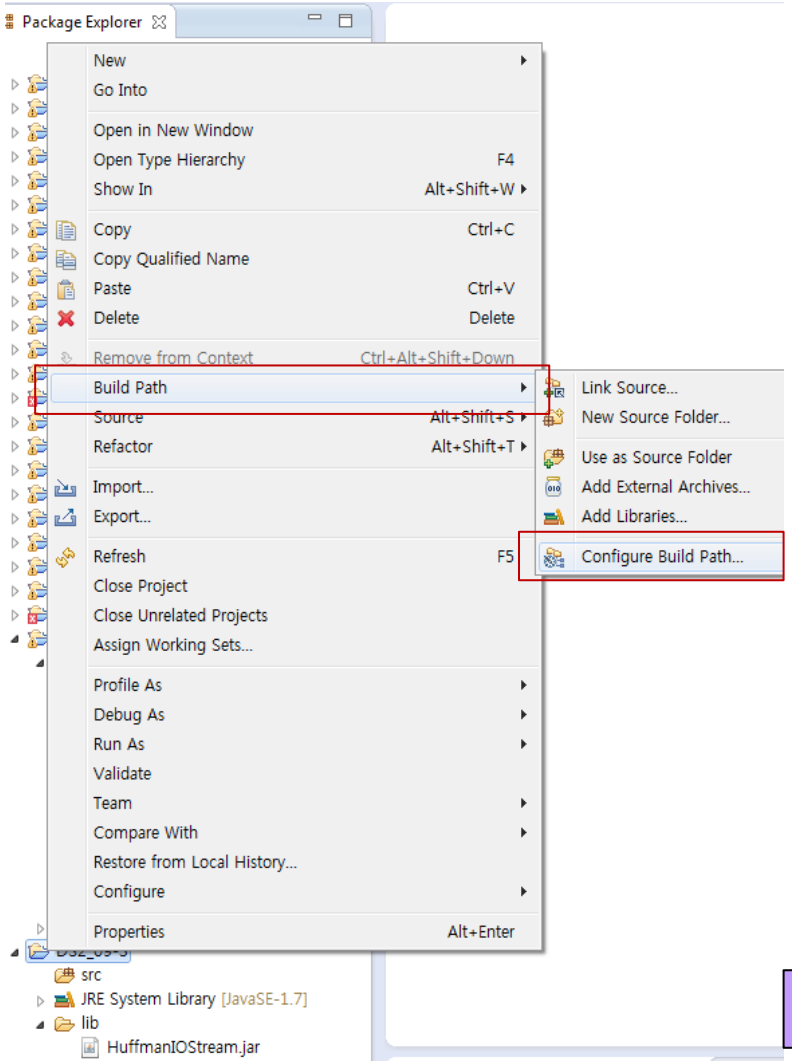

New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Refresh F5
Close Project
Close Unrelated Projects
Assign Working Sets...
Profile As
Debug As
Run As
Validate
Team
Compare With
Restore from Local History...
Configure
Properties Alt+Enter
Java Project
Project...
Package
Class
Interface
Enum
Annotation
Source Folder
Java Working Set
Folder
File
Untitled Text File
JUnit Test Case
Task
Example...
Other...
DS2_09-3
src
JRE System Library [JavaSE-1.7]

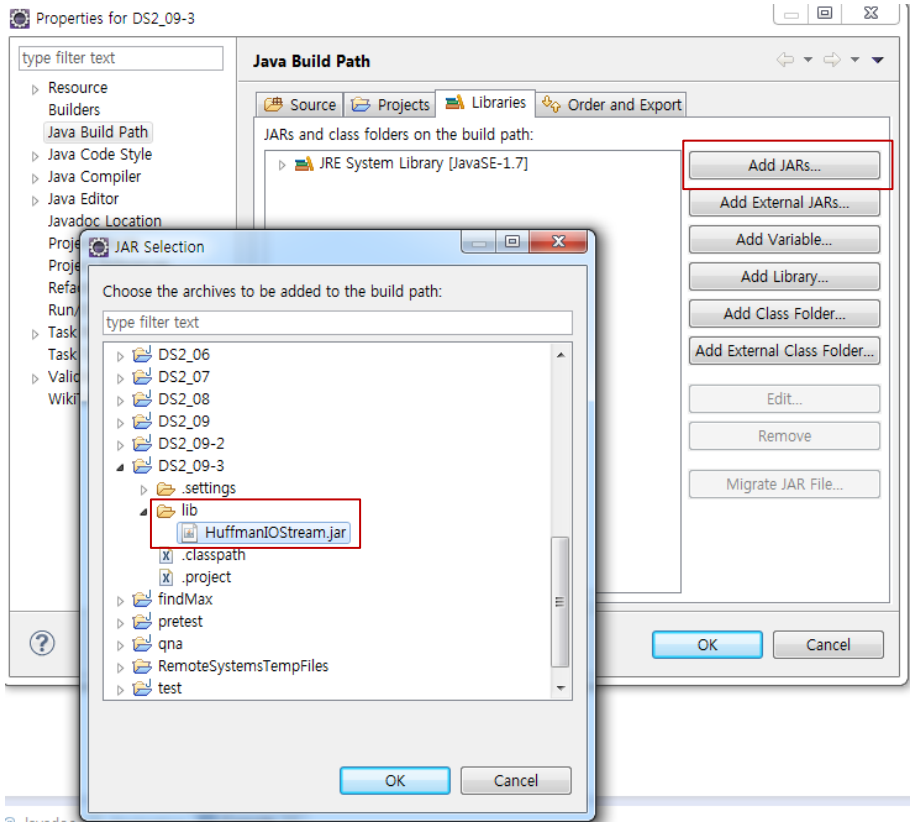

New Folder
lib폴더를 만들어서 관리
Folder
Create a new folder resource.
Enter or select the parent folder:
DS2_09-3
DS2_05
DS2_06
DS2_07
DS2_08
DS2_09
DS2_09-2
DS2_09-3
.settings
bin
src
findMax
pretest
Folder name: lib
Advanced >>
Finish
Cancel



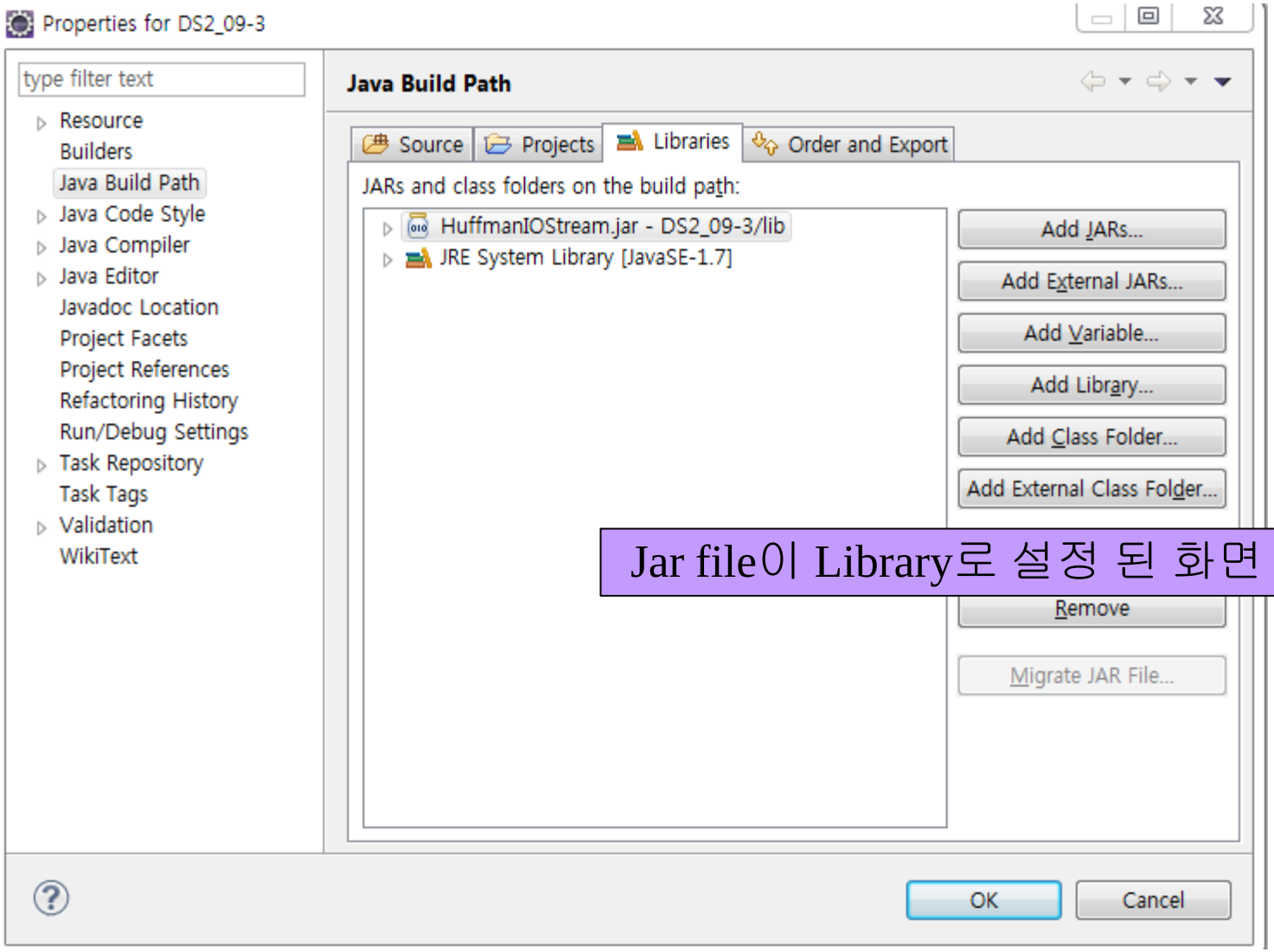


다운 받은 HuffmanIOStream.jar을 DragAndDrop하여 복사





Jar파일의 Build Path를 설정



Jar file이 Library로 설정 된 화면

Class “Huffman”

Member Functions의 구현

- Huffman의 Public Member function의 사용법
  - public void EnCoding(String originalFileName, String frequencyFileName, String resultFileName) throws IOException
    - 압축
  - public void Decoding(String originalFileName, String frequencyFileName, String resultFileName) throws IOException
    - 압축 해제

자바 입출력 확장

DataInputStream / DataOutputStream

기본 자료형 제공

메소드 / 생성자	설명
DataInputStream(InputStream in)	DataInputStream객체를 생성한다.
boolean readBoolean()	각 자료형에 맞는 데이터들을 읽어온다.
byte readByte()	
char readChar()	
short readShort()	
int readInt()	
long readLong()	
float readFloat()	
double readDouble()	
String readUTF()	UTF형식으로 쓰여진 문자를 읽는다. 더이상 데이터가 없으면 EOFException을 발생시킨다.

메소드 / 생성자	설명
DataOutputStream(OutputStream out)	DataOutputStream객체를 생성합니다.
void writeBoolean(boolean b)	각 자료형에 알맞은 데이터들을 출력한다.
void writeByte(byte b)	
void writeChar(char c)	
void writeShort(short s)	
void writeInt(int i)	
void writeLong(long l)	
void writeFloat(float f)	
void writeDouble(double b)	
void writeUTF(String s)	UTF형식의 문자를 출력한다.
void writeChars(String s)	주어진 문자열을 출력한다.

Member Functions의 구현

Huffman 의 private Member function의 구현

```
public void EnCoding(String originalFileName, String frequencyFileName, String resultFileName) throws IOException
{
        HuffmanTree huffmanTree = new HuffmanTree();

        InputStream OriginalFileInputStream = new BufferedInputStream(new FileInputStream(originalFileName));
        huffmanTree.charCount(OriginalFileInputStream);
        OriginalFileInputStream.close();

        DataOutputStream frequencyOutputStream = new DataOutputStream(new FileOutputStream(frequencyFileName));
        huffmanTree.writeFrequency(frequencyOutputStream);
        frequencyOutputStream.close();

        OriginalFileInputStream = new BufferedInputStream(new FileInputStream(originalFileName));
        HuffmanOutputStream huffmanOutputStream = new HuffmanOutputStream(new FileOutputStream(resultFileName));
        int buf = OriginalFileInputStream.read();
        while (buf != -1)
        {
                huffmanOutputStream.writeCode(huffmanTree.getCode(buf));
                buf = OriginalFileInputStream.read();
        }
        huffmanOutputStream.writeCode(huffmanTree.getCode(huffmanTree.EOF()));
        huffmanOutputStream.close();
}
```

Member Functions의 구현

- Huffman 의 private Member function의 구현
  - public void Decoding(String originalFileName, String frequencyFileName, String resultFileName) throws IOException
    - EnCoding함수를 참고하여 작성
    - Frequency를 먼저 읽어 Tree를 생성
    - 압축 파일을 열어서 Tree에 맞는 값을 얻어(readCode) resultFile에 저장

[문제 9] 요약

- 허프만 알고리즘
  - 허프만 알고리즘의 이해
    - 알고리즘의 수행 과정을 그림으로 그려 이해한 내용을 바탕으로 보고서를 작성
  - 허프만 코드로 압축을 하는 이유는 무엇인가?
  - 실습 자료 중간중간 Why?가 붙은 부분에 대하여 이유를 설명하시오.
    - P. 28, P.46

- 각 요약에 대한 내용을 보고서에 작성하여 제출하세요.

과제 제출

과제 제출

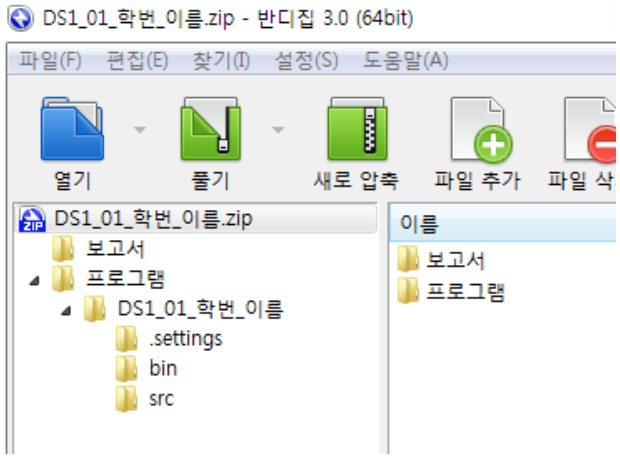
- pineai@cnu.ac.kr
  - 메일 제목 : [0X]DS2_09_학번_이름
    - 양식에 맞지 않는 메일 제목은 미제출로 간주됨
    - 앞의 0X는 분반명 (오전10시 : 00반 / 오후4시 : 01반)

- 제출 기한
  - 11월 5일(화) 23시59분까지
  - 시간 내 제출 엄수

  - 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_09_학번_이름.zip
  - 폴더의 구성
    - DS2_08_학번_이름
      - 프로그램
        - 프로젝트 폴더 / 소스
        - 메인 클래스 이름 : DS2_09_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 9 주 실습] 끝

부록 - 추가구현

- HuffmanInputStream와 HuffmanOutputStream을 구현할 경우 추가점수 부여

Class “HuffmanInputStream”

HuffmanInputStream의 멤버 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - public HuffmanInputStream(InputStream inputStream)
  - public int readCode(HuffmanTree huffmanTree)
  - public void close()

비공개 인스턴스 변수

```
import java.io.DataInputStream;
import java.io.IOException;
import java.io.InputStream;

class HuffmanInputStream {
    private DataInputStream _in;
    private int _buffer;
    private int _position;
```

Member Functions의 구현

- HuffmanInputStream 의 private Member function의 구현
  - public HuffmanInputStream(InputStream is) throws IOException
    - 비트는 8비트 기준으로 position을 지정
    - 생성자
  - public int readCode(HuffmanTree huffmanTree) throws IOException
    - 한 비트씩 읽어서 스트링에 추가하고 생성된 스트링을 decode
    - Decode 결과에 따라 Error, NOT_A_VALUE, 결과 값 등으로 다음 작업을 결정
  - private int readBit() throws IOException
    - 한 bit를 읽어온다
    - 8비트 기준으로 남은 bit가 없으면 끝인 경우 -1을
    - 아닌 경우 position을 재설정
    - 중간 bit인 경우 적정한 값 만큼 shift하여 현재 bit를 반환
  - public void close() throws IOException
    - 현재 open되어 있는 Stream을 close

Class “HuffmanOutputStream”

HuffmanOutputStream의 멤버 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - public HuffmanOutputStream(OutputStream os)
  - public void writeCode(String code)
  - private void writeBit(char c)
  - public void flush()
  - public void close()

비공개 인스턴스 변수

```
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.OutputStream;

class HuffmanOutputStream
{
    private DataOutputStream _out;
    private int _buffer;
    private int _position;
```

Member Functions의 구현

- HuffmanOutputStream의 private Member function의 구현
  - public HuffmanOutputStream(OutputStream os) throws IOException
    - 생성자
  - public void writeCode(String code) throws IOException
    - 0과 1로 된 Stream code를 한 비트씩 출력
  - private void writeBit(char c) throws IOException
    - 0또는 1의 한 비트를 출력
    - 적정한 값을 왼쪽으로 shift 시켜서 버퍼에 OR시키면 된다
  - public void flush() throws IOException
    - 한 바이트가 되면 파일에 쓴다.
  - public void close() throws IOException
    - 현재까지의 비트들을 파일에 작성하고
    - Output Stream을 close