

SP 기본

1. File I/O
2. Data Structure
3. Process & Thread
4. 동기화 (Synchronization)
5. Encryption/Decryption
6. 네트워크 프로그래밍 I (Socket)
7. Json
8. 네트워크 프로그래밍 II (Http 통신)
9. 부록
 - 1) Redirection
 - 2) Java Command를 이용한 실행 방법
 - 3) Java API 모음
 - 4) C# API 모음

1. File I/O

1.1 Overview

1.2 Text File, Binary File

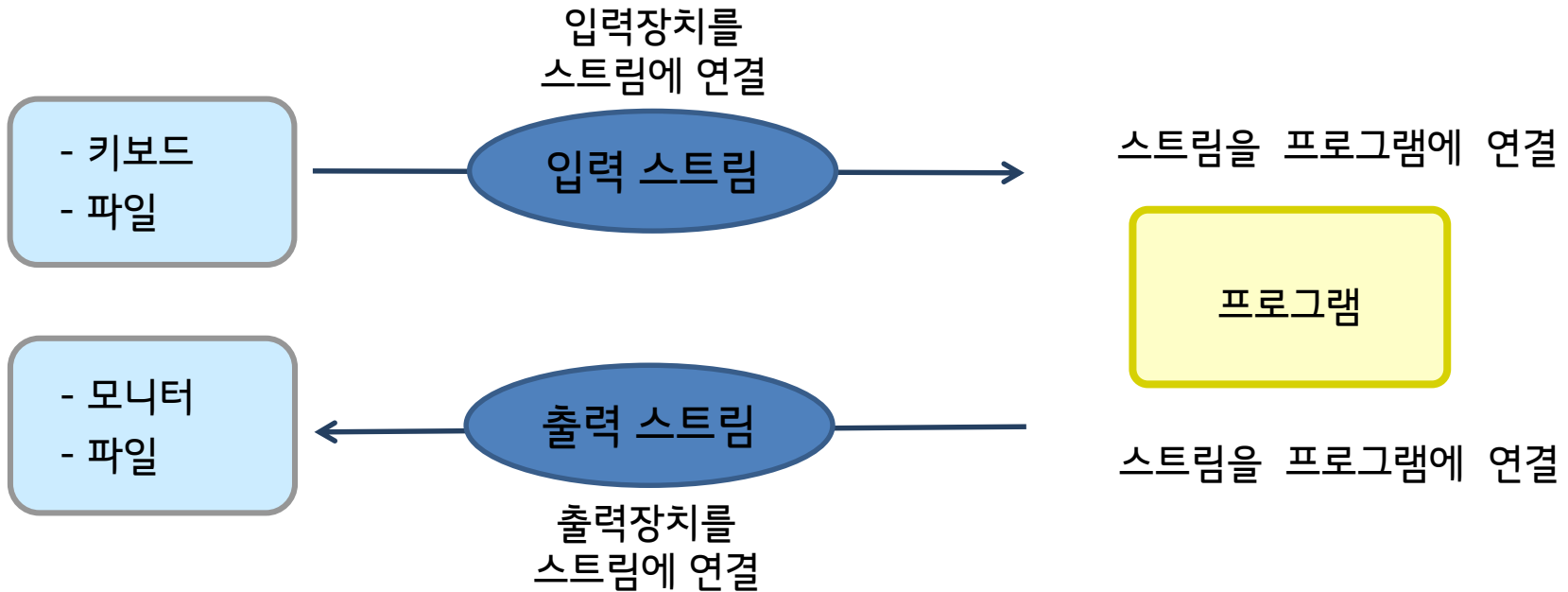
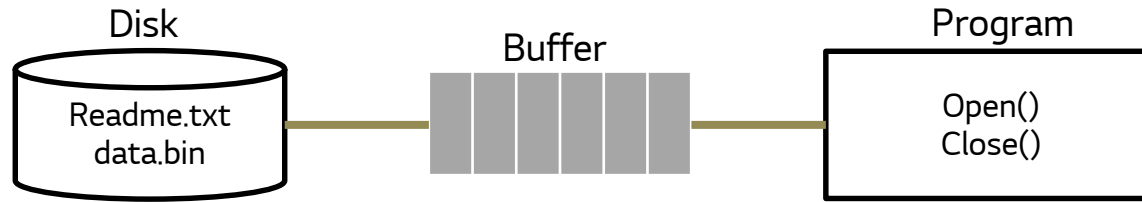
1.3 Text File Read & Print

1.4 Binary File Read & Write

1.5 File/Directory List 출력

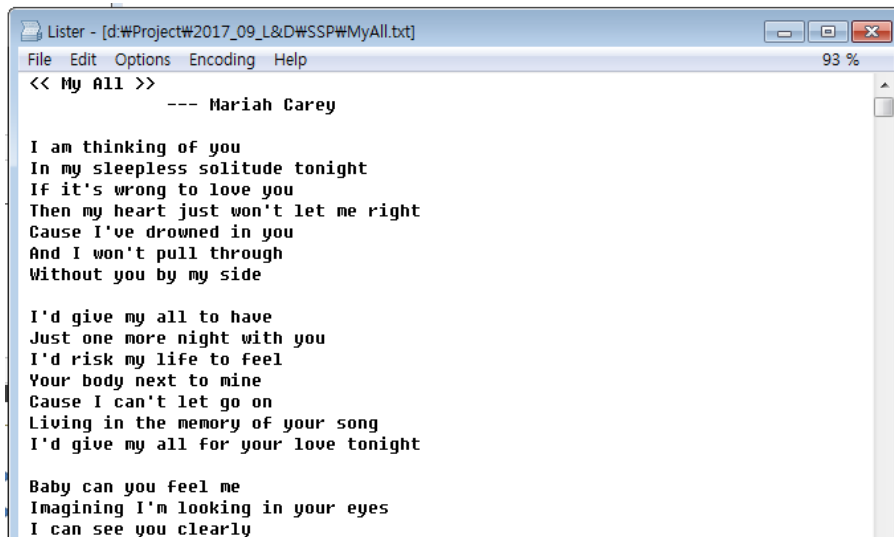
1.6 실습

1.7 참고



스트림 (Stream)	<ul style="list-style-type: none"> ➡ 파일로부터 데이터 입출력을 위해 필요한 것 ➡ 데이터를 입력 받거나 출력할 때 입출력 장치로부터 데이터가 흘러가는 것을 뜻함 ➡ 운영체제에 의해 형성되는 소프트웨어적인 상태
-----------------	--

- 파일은 기본적으로 모두 바이너리 파일
- 텍스트 파일은 파일 안의 데이터가 오직 텍스트(문자,숫자,기호들)만 들어있고 여러 행들로 구성되어 있는 것



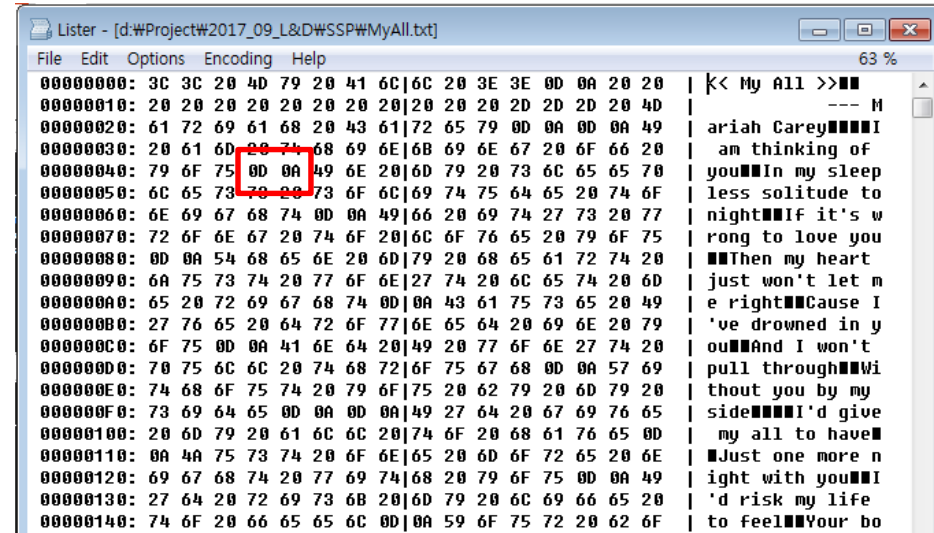
File Edit Options Encoding Help 93 %

```
<< My All >>
--- Mariah Carey

I am thinking of you
In my sleepless solitude tonight
If it's wrong to love you
Then my heart just won't let me right
Cause I've drowned in you
And I won't pull through
Without you by my side

I'd give my all to have
Just one more night with you
I'd risk my life to feel
Your body next to mine
Cause I can't let go on
Living in the memory of your song
I'd give my all for your love tonight

Baby can you feel me
Imagining I'm looking in your eyes
I can see you clearly
```



File Edit Options Encoding Help 63 %

00000000:	3C 3C 20 4D 79 20 41 6C	6C 20 3E 3E 0D 0A 20 20		< My All >>---
00000010:	20 20 20 20 20 20 20 20	20 20 20 20 20 20 20 4D		--- M
00000020:	61 72 69 61 68 20 43 61	72 65 79 0D 0A 0D 0A 49		ariah Carey
00000030:	20 61 6D 20 74 68 69 6E	6B 69 6E 67 20 6F 66 20		am thinking of
00000040:	79 6F 75 0D 0A 49 6E	20 6D 79 20 73 6C 65 65 70		you
00000050:	6C 65 73 70 20 73 6F 6C	69 74 75 64 65 20 74 6F		In my sleep
00000060:	6E 69 67 68 74 0D 0A 49	66 20 69 74 27 73 20 77		less solitude to
00000070:	72 6F 6E 67 20 74 6F 20	6C 6F 76 65 20 79 6F 75		night
00000080:	0D 0A 54 68 65 6E 20 6D	79 20 68 65 61 72 74 20		If it's w
00000090:	6A 75 73 74 20 77 6F 6E	27 74 20 6C 65 74 20 6D		rong to love you
000000A0:	65 20 72 69 67 68 74 0D	0A 43 61 75 73 65 20 49		Then my heart
000000B0:	27 76 65 20 64 72 6F 77	6E 65 64 20 69 6E 20 79		just won't let m
000000C0:	6F 75 0D 0A 41 6E 64 20	49 20 77 6F 6E 27 74 20		e right
000000D0:	70 75 6C 6C 20 74 68 72	6F 75 67 68 0D 0A 57 69		Cause I
000000E0:	74 68 6F 75 74 20 79 6F	75 20 62 79 20 6D 79 20		've drowned in y
000000F0:	73 69 64 65 0D 0A 0D 0A	49 27 64 20 67 69 76 65		ou
00000100:	20 6D 79 20 61 6C 6C 20	74 6F 20 68 61 76 65 0D		And I won't
00000110:	0A 4A 75 73 74 20 6F 6E	65 20 6D 6F 72 65 20 6E		pull through
00000120:	69 67 68 74 20 77 69 74	68 20 79 6F 75 0D 0A 49		Wi
00000130:	27 64 20 72 69 73 68 20	6D 79 20 6C 69 66 65 20		thout you by my
00000140:	74 6F 20 66 65 65 6C 0D	0A 59 6F 75 72 20 62 6F		side

- 개행문자 : CR(Carriage Return, 0x0D), LF(Line Feed, 0x0A)
 - ➔ 유닉스/리눅스 시스템과 맥 OSX에서의 개행은 LF(라인 피드)만 사용

```
Void PrintFile(String fileName)
{
```

```
    String line = null;
```

```
    try {
```

```
        FileReader fileReader = new FileReader(fileName);
```

```
        BufferedReader bufferedReader = new BufferedReader(fileReader);
```

```
        while((line = bufferedReader.readLine()) != null) {
```

```
            System.out.println(line);
```

```
        }
```

```
        bufferedReader.close();
```

```
    }
```

```
    catch (FileNotFoundException e) {
```

```
        e.printStackTrace();
```

```
    }
```

```
    catch (IOException e) {
```

```
        e.printStackTrace();
```

```
    }
```

```
}
```

입력 스트림

입력 스트림으로부터
문자를 읽을 때 버퍼링

버퍼로부터 한 줄씩 읽기

화면에 한 줄씩 출력

버퍼 리더 닫기

```
void CopyFile(String inputFile, String outputFile)
{
    final int BUFFER_SIZE = 4096;
    int readLen;
    try {
        InputStream inputStream = new FileInputStream(inputFile);
        OutputStream outputStream = new FileOutputStream(outputFile);

        byte[] buffer = new byte[BUFFER_SIZE];

        while ((readLen = inputStream.read(buffer)) != -1) {
            outputStream.write(buffer, 0, readLen);
        }

        inputStream.close();
        outputStream.close();

    }
    catch (FileNotFoundException e) {
        e.printStackTrace();
    }
    catch (IOException ex) {
        ex.printStackTrace();
    }
}
```

바이트 단위 입출력 스트림

바이트 배열 버퍼 생성

파일로부터 읽어서 버퍼에 담기

버퍼의 내용을 파일에 쓰기

입출력 스트림 닫기

```
static void PrintFile(string filename)
{
    string line;

    StreamReader file = new StreamReader(filename);
    while ((line = file.ReadLine()) != null)
    {
        System.Console.WriteLine(line);
    }
    file.Close();
}
```

입력 스트림

한 줄씩 읽기

화면에 한 줄씩 출력

스트림 닫기

```
static void CopyFile(string InputFilename, string OutputFilename)
{
```

```
    const int BUF_SIZE = 4096;
```

```
    byte[] buffer = new byte[BUF_SIZE];
```

바이트 배열 버퍼 생성

```
    int nReadLen;
```

```
    FileStream fs_in =
```

```
        new FileStream(InputFilename, FileMode.Open, FileAccess.Read);
```

```
    FileStream fs_out =
```

```
        new FileStream(OutputFilename, FileMode.Create, FileAccess.Write);
```

바이트 단위
입출력 스트림

```
    while ((nReadLen = fs_in.Read(buffer, 0, BUF_SIZE)) > 0)
```

파일로부터 읽어서 버퍼에 담기

```
    {
```

```
        fs_out.Write(buffer, 0, nReadLen);
```

버퍼의 내용을 파일에 쓰기

```
    }
```

```
    fs_in.Close();
```

```
    fs_out.Close();
```

입출력 스트림 닫기

```
}
```



```
void PrintFile(const char * filename)
{
```

```
    char buf[4096];
```

```
    FILE * fp;
```

```
    fp = fopen(filename, "rt");
```

텍스트 파일 읽기 오픈

```
    if (!fp)
```

```
    {
```

```
        printf("File Open Error\n");
```

```
        return;
```

```
    }
```

파일로부터 한 줄씩 읽어서 버퍼에 담기

```
    while (fgets(buf, 4096, fp) != NULL)
```

```
    {
```

```
        printf("%s", buf);
```

화면에 한 줄씩 출력

```
    }
```

```
    fclose(fp);
```

파일 닫기

```
}
```

```
void CopyFile(const char * inputFilename, const char * outputFilename)
{
    char buf[4096];
    FILE *rfp, *wfp;
    int readLen;

    rfp = fopen(inputFilename, "rb");
    wfp = fopen(outputFilename, "wb");
    if (!rfp || !wfp)
    {
        printf("File Open Error\n");
        return;
    }

    while ((readLen = fread(buf, 1, 4096, rfp)) != 0)
    {
        fwrite(buf, 1, readLen, wfp);
    }

    fclose(rfp);
    fclose(wfp);
}
```

char 배열 버퍼 생성

바이너리 파일 오픈

파일로부터 읽어서 버퍼에 담기

버퍼의 내용을 파일에 쓰기

파일 닫기

```
void printfile(const char * filename)
{
    string line;
    fstream fs;
    fs.open(filename, fstream::in);

    while (getline(fs, line))
    {
        cout << line << endl;
    }

    fs.close();
}
```

파일 입력 스트림

한 줄씩 읽기

화면에 한 줄씩 출력

스트림 닫기

```
void copyfile(const char * InputFilename, const char * OutputFilename)
{
    const int BUF_LEN = 4096;
    char buffer[BUF_LEN];

    fstream fs_in, fs_out;
    fs_in.open(InputFilename, fstream::in | fstream::binary);
    fs_out.open(OutputFilename, fstream::out | fstream::binary);

    while (!fs_in.eof())
    {
        fs_in.read(buffer, BUF_LEN);

        fs_out.write(buffer, fs_in.gcount());

    }

    fs_in.close();
    fs_out.close();
}
```

char 배열 버퍼 생성

파일 입출력 스트림

파일로부터 읽어서 버퍼에 담기

버퍼의 내용을 파일에 쓰기

입출력 스트림 닫기

```
void FileDirList()  
{
```

디렉토리 설정

```
    File directory = new File(".");  
    File[] fList = directory.listFiles();
```

설정된 디렉토리 내의 파일과
디렉토리를 리스트에 저장

```
    for (File file : fList) {
```

디렉토리인 경우

```
        if(file.isDirectory()) {  
            System.out.println "["+file.getName()+"]");  
        }
```

디렉토리 이름 출력

```
    else {  
        System.out.println(file.getName());  
    }
```

디렉토리가 아닌 경우

파일 이름 출력

```
    }
```

```
}
```

➤ 재귀호출을 이용한 하위 폴더 파일 탐색

```
void FileSearchAll(String path)
{
    File directory = new File(path);
    File[] fList = directory.listFiles();

    for (File file : fList) {
        if (file.isDirectory()) {
            FileSearchAll(file.getPath());
        }
        else {
            System.out.println(file.getName());
        }
    }
}
```

```
void FileDirList()  
{
```

```
    string[] subdirectoryEntries = Directory.GetDirectories(".");
```

현재 디렉토리 내의 디렉토리들을
리스트에 저장

```
    foreach (string subdirectory in subdirectoryEntries)  
        Console.WriteLine("{0}", subdirectory);
```

디렉토리 리스트 출력

```
    string[] fileEntries = Directory.GetFiles(".");
```

현재 디렉토리 내의 파일들을
리스트에 저장

```
    foreach (string fileName in fileEntries)  
        Console.WriteLine(fileName);
```

파일 리스트 출력

```
}
```

```
DirectoryInfo di = new DirectoryInfo("./INPUT");  
FileInfo[] fiArr = di.GetFiles("*.*", SearchOption.AllDirectories);  
foreach (var f in fiArr)  
{  
    Console.WriteLine(f.Name);  
}
```

하위 폴더들까지 모두 탐색

```
void FileDirList()  
{
```

```
    WIN32_FIND_DATA ffd;  
    HANDLE hFind;
```

```
    hFind = FindFirstFile(TEXT("./*./*"), &ffd);
```

디렉토리 설정 후 핸들 가져오기
& 첫 파일 혹은 디렉토리 찾기

```
    do  
    {
```

```
        if (ffd.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
```

디렉토리인 경우

```
        {  
            _tprintf(TEXT("[%s]Wn"), ffd.cFileName);
```

디렉토리 이름 출력

```
        }
```

```
    else
```

디렉토리가 아닌 경우

```
    {
```

```
        _tprintf(TEXT("%sWn"), ffd.cFileName);
```

파일 이름 출력

```
    }
```

```
    } while (FindNextFile(hFind, &ffd) != 0);
```

다음 파일 혹은 디렉토리 찾기

```
    FindClose(hFind);
```

핸들 닫기

```
}
```



```
void FileDirList()
```

```
{
```

```
    struct dirent *de;
```

파일 혹은 디렉토리 정보를 담을 구조체

```
    DIR *dr = opendir(".");
```

디렉토리 열고 포인터 가져오기

```
    while ((de = readdir(dr)) != NULL)
```

```
    {
```

디렉토리 혹은 파일 찾기

```
        if (de->d_type != DT_REG)
```

일반 파일이 아닌 경우

```
        {
```

```
            printf("[%s]\n", de->d_name);
```

디렉토리 이름 출력

```
        }
```

```
    else
```

일반 파일인 경우

```
    {
```

```
        printf("%s\n", de->d_name);
```

파일 이름 출력

```
    }
```

```
}
```

```
    closedir(dr);
```

디렉토리 닫기

```
}
```

◆ File/Directory List 출력 참고 사이트

JAVA

<https://dzone.com/articles/java-example-list-all-files>

C언어

<http://www.geeksforgeeks.org/c-program-list-files-sub-directories-directory/>

C++

[https://msdn.microsoft.com/en-us/library/windows/desktop/aa365200\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/aa365200(v=vs.85).aspx)

C#

[https://msdn.microsoft.com/en-us/library/07wt70x2\(v=vs.110\).aspx?cs-save-lang=1&cs-lang=csharp#code-snippet-1](https://msdn.microsoft.com/en-us/library/07wt70x2(v=vs.110).aspx?cs-save-lang=1&cs-lang=csharp#code-snippet-1)

◆ Directory 생성 API

JAVA

```
File destFolder = new File("./OUTPUT");
if(!destFolder.exists()) {
    destFolder.mkdirs();
}
```

C#

```
System.IO.Directory.CreateDirectory("./OUTPUT");
```

C++

```
_mkdir("./OUTPUT");
```

C언어

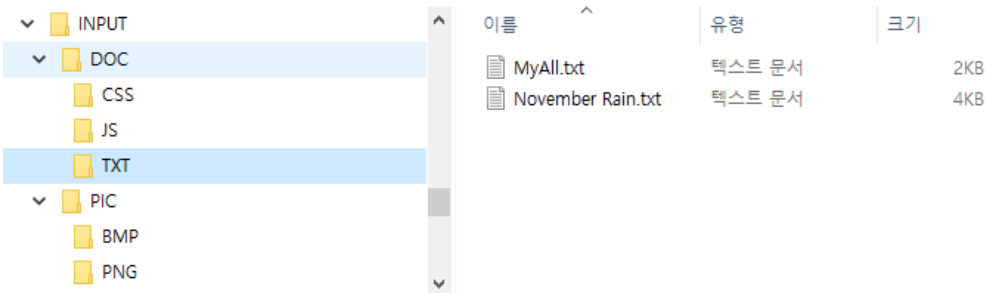
```
mkdir("./OUTPUT",0777);
```

1. INPUT 폴더 하위에 위치한 파일들의 파일명(상대경로 포함), 크기를 Console화면에 출력하시오.
2. INPUT 폴더 하위에 위치한 파일들 중 크기가 3Kbyte가 넘는 파일들은 모두 OUTPUT 폴더로 복사하시오. (OUTPUT 폴더 및 서브 폴더 생성)
단, 파일 복사 시 바이너리 파일을 버퍼에 읽고 쓰는 방식으로 구현하고, 버퍼의 크기는 512Byte로 설정하시오.

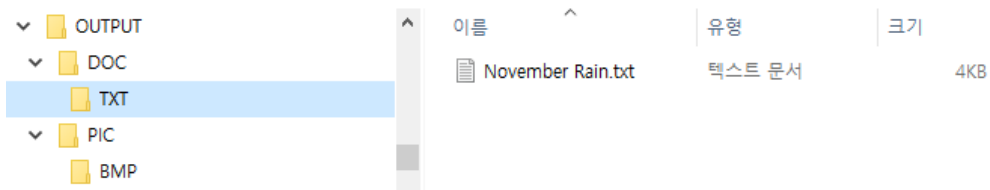
[Console 출력]

```
.#test.exe: 4608bytes.  
.#DOC#CSS#main-cef-mac.css: 2538bytes.  
.#DOC#CSS#main-cef-ui-theme.css: 2515bytes.  
.#DOC#JS#config.js: 1427bytes.  
.#DOC#JS#desktop.js: 1032bytes.  
.#DOC#TXT#MyAll.txt: 1086bytes.  
.#DOC#TXT#November Rain.txt: 3518bytes.  
.#PIC#BMP#progress_bg_center.bmp: 5670bytes.  
.#PIC#BMP#progress_bg_left.bmp: 834bytes.  
.#PIC#BMP#progress_bg_right.bmp: 834bytes.  
.#PIC#BMP#progress_center.bmp: 834bytes.  
.#PIC#BMP#progress_left.bmp: 106bytes.  
.#PIC#BMP#progress_right.bmp: 106bytes.  
.#PIC#PNG#close_x.png: 306bytes.  
.#PIC#PNG#japanese_over.png: 568bytes.
```

[INPUT 폴더]



[OUTPUT 폴더]



Ch 2. Data Structure

2.1 List

2.2 Map

2.3 Queue

2.4 실습

- 데이터를 순차적으로 처리하는 구조
- List 에 데이터를 추가하는 순서대로 인덱스 번호가 매겨짐
- 데이터의 중복을 허용
- 배열과 매우 유사한 구조를 갖고 있음
- 정렬 기능

JAVA

```
ArrayList<T> L = new ArrayList<T>();  
  
L.add(t);  
  
Collections.sort(L, ...);
```

C#

```
List<T> L = new List<T>();  
  
L.add(t);  
  
L.sort(...);
```

C++

```
std::list<T> L;  
  
L.push_back(t);  
  
L.sort(...);
```

C언어 (Glib)

```
GArray* a = g_array_new(FALSE, FALSE, sizeof(T));  
  
g_array_append_val(L, t);  
  
g_array_sort(L, ...);
```

기본 입출력

```
ArrayList<String> al = new ArrayList<String>();
```

ArrayList 인스턴스 생성

```
al.add("Michael Knight");  
al.add("Mac Guyver");  
al.add("Clark Kent");  
al.add("Bruce Wayne");  
al.add("Tony Stark");
```

데이터 입력

```
for (String name : al) {  
    System.out.println(name);  
}  
System.out.println();
```

출력(방식 1)

```
al.remove("Clark Kent");
```

데이터 삭제

```
for (int i=0; i<al.size(); i++) {  
    System.out.println(al.get(i));  
}  
System.out.println();
```

출력(방식 2)

```
al.remove(al.get(0));
```

```
Iterator<String> itr = al.iterator();  
while (itr.hasNext()) {  
    System.out.println(itr.next());  
}
```

출력(방식 3)

➤ 출력 결과

```
Michael Knight  
Mac Guyver  
Clark Kent  
Bruce Wayne  
Tony Stark
```

```
Michael Knight  
Mac Guyver  
Bruce Wayne  
Tony Stark
```

```
Mac Guyver  
Bruce Wayne  
Tony Stark
```

정렬

```
ArrayList<String> al = new ArrayList<String>();
```

```
al.add("Michael Knight"); al.add("Mac Guyver"); al.add("Clark Kent");  
al.add("Bruce Wayne"); al.add("Tony Stark");
```

```
Collections.sort(al);
```

정렬 (오름차순)

```
for (String name : al) {  
    System.out.println(name);  
}  
System.out.println();
```

정렬 Comparator

```
Comparator<String> co = new Comparator<String>() {  
    public int compare(String o1, String o2) {  
        return (o2.compareTo(o1));  
    }  
};
```

내림차순

→ o2, o1 자리변경 시 오름차순

```
Collections.sort(al, co);  
for (String name : al) {  
    System.out.println(name);  
}  
System.out.println();
```

정렬 (Comparator 사용)

정렬 (람다식 사용)
- JAVA8부터 가능

```
Collections.sort(al, (g1, g2) -> g1.compareTo(g2));  
for (String name : al) {  
    System.out.println(name);  
}
```

➤ 출력 결과

```
Bruce Wayne  
Clark Kent  
Mac Guyver  
Michael Knight  
Tony Stark
```

```
Tony Stark  
Michael Knight  
Mac Guyver  
Clark Kent  
Bruce Wayne
```

```
Bruce Wayne  
Clark Kent  
Mac Guyver  
Michael Knight  
Tony Stark
```

기본 입출력

```
List<string> al = new List<string>();
```

List 인스턴스 생성

```
al.Add("Michael Knight");  
al.Add("Mac Guyver");  
al.Add("Clark Kent");  
al.Add("Bruce Wayne");  
al.Add("Tony Stark");
```

데이터 입력

```
foreach (string name in al) {  
    Console.WriteLine(name);  
}
```

출력(방식 1)

```
al.Remove("Clark Kent");
```

데이터 삭제

```
for (int i=0; i<al.Count; i++) {  
    Console.WriteLine(al[i]);  
}
```

출력(방식 2)

```
al.Remove(al[0]);
```

```
var enumerator = al.GetEnumerator();  
while (enumerator.MoveNext()) {  
    Console.WriteLine(enumerator.Current);  
}
```

출력(방식 3)

➤ 출력 결과

```
Michael Knight  
Mac Guyver  
Clark Kent  
Bruce Wayne  
Tony Stark
```

```
Michael Knight  
Mac Guyver  
Bruce Wayne  
Tony Stark
```

```
Mac Guyver  
Bruce Wayne  
Tony Stark
```


정렬

```
List<string> al = new List<string>();
```

```
al.Add("Michael Knight"); al.Add("Mac Guyver");  
al.Add("Clark Kent"); al.Add("Bruce Wayne"); al.Add("Tony  
Stark");
```

```
al.Sort();
```

정렬 (오름차순)

```
foreach (string name in al) {  
    Console.WriteLine(name);  
}  
Console.WriteLine();
```

```
al.Sort(delegate (string x, string y)  
{  
    return y.CompareTo(x);  
});
```

정렬 (내림차순)
→ y, x 자리변경 시 오름차순

```
foreach (string name in al) {  
    Console.WriteLine(name);  
}  
Console.WriteLine();
```

정렬 (람다식 사용)

```
al.Sort((string x, string y) => x.CompareTo(y));  
foreach (string name in al) {  
    Console.WriteLine(name);  
}
```

➤ 출력 결과

```
Bruce Wayne  
Clark Kent  
Mac Guyver  
Michael Knight  
Tony Stark
```

```
Tony Stark  
Michael Knight  
Mac Guyver  
Clark Kent  
Bruce Wayne
```

```
Bruce Wayne  
Clark Kent  
Mac Guyver  
Michael Knight  
Tony Stark
```

기본 입출력

```
list<string> al;  
list<string>::iterator itr;
```

ArrayList 인스턴스 생성

```
al.push_back("Michael Knight");  
al.push_back("Mac Guyver");  
al.push_back("Clark Kent");  
al.push_back("Bruce Wayne");  
al.push_back("Tony Stark");
```

데이터 입력

```
for (itr = al.begin(); itr != al.end(); itr++) {  
    cout << *itr << endl;  
}  
cout << endl;
```

출력

```
al.remove("Clark Kent");
```

데이터 삭제

```
for (itr = al.begin(); itr != al.end(); itr++) {  
    cout << *itr << endl;  
}  
cout << endl;
```

```
al.remove(al.front());
```

처음 데이터 삭제

```
for (itr = al.begin(); itr != al.end(); itr++) {  
    cout << *itr << endl;  
}
```

➤ 출력 결과

```
Michael Knight  
Mac Guyver  
Clark Kent  
Bruce Wayne  
Tony Stark
```

```
Michael Knight  
Mac Guyver  
Bruce Wayne  
Tony Stark
```

```
Mac Guyver  
Bruce Wayne  
Tony Stark
```

정렬

```
list<string> al;  
list<string>::iterator itr;
```

```
al.push_back("Michael Knight");  
al.push_back("Mac Guyver");  
al.push_back("Clark Kent");  
al.push_back("Bruce Wayne");  
al.push_back("Tony Stark");
```

```
bool Comp(string x, string y)  
{  
    if (y.compare(x) < 0)  
        return true;  
    else  
        return false;  
}
```

정렬 Comparator (내림차순)
→ x, y 자리변경 시 오름차순

```
al.sort();
```

정렬 (오름차순)

```
for (itr = al.begin(); itr != al.end(); itr++) {  
    cout << *itr << endl;  
}  
cout << endl;
```

```
al.sort(Comp);
```

정렬 (Comparator 사용)

```
for (itr = al.begin(); itr != al.end(); itr++) {  
    cout << *itr << endl;  
}  
cout << endl;
```

정렬 (람다식 사용)

```
al.sort([](string x, string y) {return (x.compare(y) < 0); });
```

```
for (itr = al.begin(); itr != al.end(); itr++) {  
    cout << *itr << endl;  
}
```

➤ 출력 결과

```
Bruce Wayne  
Clark Kent  
Mac Guyver  
Michael Knight  
Tony Stark
```

```
Tony Stark  
Michael Knight  
Mac Guyver  
Clark Kent  
Bruce Wayne
```

```
Bruce Wayne  
Clark Kent  
Mac Guyver  
Michael Knight  
Tony Stark
```

기본 입출력

❖ glib 사용

```
static void itr_print(gpointer value, gpointer user_data) {  
    printf("%s\n", (char *)value);  
}
```

Iterator 함수

```
void main( ) {  
    int i;  
    char* pTemp;
```

```
    GPtrArray* al = g_ptr_array_new();  
    g_ptr_array_add(al, g_strdup("Michael Knight"));  
    g_ptr_array_add(al, g_strdup("Mac Guyver"));  
    g_ptr_array_add(al, g_strdup("Clark Kent"));  
    g_ptr_array_add(al, g_strdup("Bruce Wayne"));  
    g_ptr_array_add(al, g_strdup("Tony Stark"));
```

GPtrArray 포인터 할당

데이터 입력

```
    g_ptr_array_foreach(al, (GFunc)itr_print, NULL);  
    printf("Wn");
```

출력(방식 1)

```
    g_ptr_array_remove_index(al, 2);
```

3번째(Index 2) 데이터 삭제

```
    for (i = 0; i < al->len; i++)  
    {  
        pTemp = g_ptr_array_index(al, i);  
        printf("%s\n", pTemp);  
    }
```

출력(방식 2)

```
    g_ptr_array_free(al, TRUE);
```

메모리 해제

➤ 출력 결과

```
Michael Knight  
Mac Guyver  
Clark Kent  
Bruce Wayne  
Tony Stark
```

```
Michael Knight  
Mac Guyver  
Bruce Wayne  
Tony Stark
```

정렬

❖ glib 사용

정렬 Comparator
(오름차순)

정렬 Comparator
(내림차순)

```
static int compare_name(gpointer a, gpointer b) {
    int * x = (int *)a; int * y = (int *)b;
    return strcmp((char *)*x, (char *)*y);
}

static int compare_name_reverse(gpointer a, gpointer b) {
    int * x = (int *)a; int * y = (int *)b;
    return strcmp((char *)*y, (char *)*x);
}

static void itr_print(gpointer value, gpointer user_data) {
    printf("%s\n", (char *)value);
}
```

```
void main( ) {
    GPtrArray* al = g_ptr_array_new();
    g_ptr_array_add(al, g_strdup("Michael Knight"));
    g_ptr_array_add(al, g_strdup("Mac Guyver"));
    g_ptr_array_add(al, g_strdup("Clark Kent"));
    g_ptr_array_add(al, g_strdup("Bruce Wayne"));
    g_ptr_array_add(al, g_strdup("Tony Stark"));

    g_ptr_array_sort(al, (gpointer)compare_name);
    g_ptr_array_foreach(al, (GFunc)itr_print, NULL);
    printf("\n");

    g_ptr_array_sort(al, (gpointer)compare_name_reverse);
    g_ptr_array_foreach(al, (GFunc)itr_print, NULL);

    g_ptr_array_free(al, TRUE);
}
```

정렬 (Comparator 사용)

➤ 출력 결과

```
Bruce Wayne
Clark Kent
Mac Guyver
Michael Knight
Tony Stark
```

```
Tony Stark
Michael Knight
Mac Guyver
Clark Kent
Bruce Wayne
```

- 저장되는 순서가 유지되지 않는 구조
- 키(key)와 값(value)의 쌍으로 저장 (키와 값 모두 객체)
- 키(key)의 중복을 허용하지 않음
- 키(key)를 이용하여 각 값(value)을 구별할 수 있음

JAVA

```
HashMap<K, V> m = new HashMap<K, V> ();  
  
m.put(key, value);  
  
for(Integer key : m.keySet() ) {  
    key...  
    m.get(key)...  
};
```

C#

```
Dictionary<K, V> m = new Dictionary<K, V>();  
  
m.Add(key, value);  
  
foreach (KeyValuePair<K, V> items in m) {  
    items.key...  
    items.value...  
};
```

C++

```
map<K, V> m;  
  
m.insert(pair<K, V>(key, value));  
  
map<string, Effort>::iterator mit;  
for (mit = m.begin(); mit != m.end(); mit++) {  
    mit->first...  
    mit->second...  
}
```

C언어 (Glib)

```
GHashTable* hash =  
    g_hash_table_new(g_str_hash, g_str_equal);  
  
g_hash_table_insert(hash, key, value);  
  
g_hash_table_foreach(hash, (GHFunc)iterator_func, NULL);
```

➤ 기타 Map

Java	C#	설명
LinkedHashMap		저장되는 순서 유지
SortedHashMap	SortedDictionary	Key로 정렬하여 저장
ConcurrentHashMap	ConcurrentDictionary	Thread-safe하게 사용

기본 입출력

```
HashMap<String, String> m = new HashMap<String, String>();
```

HashMap 인스턴스 생성

```
m.put("kit@gmail.com", "Michael Knight");  
m.put("knife@gmail.com", "Mac Guyver");  
m.put("superman@gmail.com", "Clark Kent");  
m.put("batman@gmail.com", "Bruce Wayne");  
m.put("ironman@gmail.com", "Tony Stark");
```

데이터 입력

➤ 출력 결과

```
for ( String key : m.keySet() ) {  
    System.out.println( key + " : " +m.get( key ) );  
}  
System.out.println();
```

출력

```
superman@gmail.com : Clark Kent  
ironman@gmail.com : Tony Stark  
batman@gmail.com : Bruce Wayne  
knife@gmail.com : Mac Guyver  
kit@gmail.com : Michael Knight
```

```
m.remove("superman@gmail.com");
```

데이터 삭제

```
for ( String key : m.keySet() ) {  
    System.out.println( key + " : " +m.get( key ) );  
}  
System.out.println();
```

```
ironman@gmail.com : Tony Stark  
batman@gmail.com : Bruce Wayne  
knife@gmail.com : Mac Guyver  
kit@gmail.com : Michael Knight
```

```
m.replace("batman@gmail.com", "Robin");
```

데이터(Value) 변경

```
for ( String key : m.keySet() ) {  
    System.out.println( key + " : " +m.get( key ) );  
}
```

```
ironman@gmail.com : Tony Stark  
batman@gmail.com : Robin  
knife@gmail.com : Mac Guyver  
kit@gmail.com : Michael Knight
```


기본 입출력

```
Dictionary<string, string> m = new Dictionary<string, string>();  
m.Add("kit@gmail.com", "Michael Knight");  
m.Add("knife@gmail.com", "Mac Guyver");  
m.Add("superman@gmail.com", "Clark Kent");  
m.Add("batman@gmail.com", "Bruce Wayne");  
m.Add("ironman@gmail.com", "Tony Stark");
```

Dictionary 인스턴스 생성

데이터 입력

```
foreach (KeyValuePair<string, string> items in m)  
{  
    Console.WriteLine(items.Key + " : " + items.Value);  
}  
Console.WriteLine();
```

출력

➤ 출력 결과

```
m.Remove("superman@gmail.com");  
  
foreach (KeyValuePair<string, string> items in m)  
{  
    Console.WriteLine(items.Key + " : " + items.Value);  
}  
Console.WriteLine();
```

데이터 삭제

```
m["batman@gmail.com"] = "Robin";  
  
foreach (KeyValuePair<string, string> items in m)  
{  
    Console.WriteLine(items.Key + " : " + items.Value);  
}
```

데이터(Value) 변경

```
kit@gmail.com : Michael Knight  
knife@gmail.com : Mac Guyver  
superman@gmail.com : Clark Kent  
batman@gmail.com : Bruce Wayne  
ironman@gmail.com : Tony Stark  
  
kit@gmail.com : Michael Knight  
knife@gmail.com : Mac Guyver  
batman@gmail.com : Bruce Wayne  
ironman@gmail.com : Tony Stark  
  
kit@gmail.com : Michael Knight  
knife@gmail.com : Mac Guyver  
batman@gmail.com : Robin  
ironman@gmail.com : Tony Stark
```

기본 입출력

```
map<string, string> m;  
map<string, string>::iterator mit;
```

map 인스턴스 생성

```
m.insert(pair<string, string>("kit@gmail.com", "Michael Knight"));  
m.insert(pair<string, string>("knife@gmail.com", "Mac Guyver"));  
m.insert(pair<string, string>("superman@gmail.com", "Clark Kent"));  
m.insert(pair<string, string>("batman@gmail.com", "Bruce Wayne"));  
m.insert(pair<string, string>("ironman@gmail.com", "Tony Stark"));
```

데이터 입력

```
for (mit = m.begin(); mit != m.end(); mit++) {  
    cout << mit->first << " : " << mit->second << endl;  
}  
cout << endl;
```

출력

```
m.erase("superman@gmail.com");
```

데이터 삭제

```
for (mit = m.begin(); mit != m.end(); mit++) {  
    cout << mit->first << " : " << mit->second << endl;  
}  
cout << endl;
```

데이터(Value) 변경

```
m.find("batman@gmail.com")->second = "Robin";  
  
for (mit = m.begin(); mit != m.end(); mit++) {  
    cout << mit->first << " : " << mit->second << endl;  
}
```

➤ 출력 결과

```
batman@gmail.com : Bruce Wayne  
ironman@gmail.com : Tony Stark  
kit@gmail.com : Michael Knight  
knife@gmail.com : Mac Guyver  
superman@gmail.com : Clark Kent  
  
batman@gmail.com : Bruce Wayne  
ironman@gmail.com : Tony Stark  
kit@gmail.com : Michael Knight  
knife@gmail.com : Mac Guyver  
  
batman@gmail.com : Robin  
ironman@gmail.com : Tony Stark  
kit@gmail.com : Michael Knight  
knife@gmail.com : Mac Guyver
```

기본 입출력

❖ glib 사용

```
static void iterator_print(gpointer key, gpointer value, gpointer user_data) {  
    printf("%s : %s\n", (char *)key, (char *)value);  
}
```

Iterator 함수

```
void main( ) {  
    gpointer gp;  
    GHashTable* hash = g_hash_table_new(g_str_hash, g_str_equal);
```

GHashTable 포인터 할당

데이터 입력

```
    g_hash_table_insert(hash, "kit@gmail.com", "Michael Knight");  
    g_hash_table_insert(hash, "knife@gmail.com", "Mac Guyver");  
    g_hash_table_insert(hash, "superman@gmail.com", "Clark Kent");  
    g_hash_table_insert(hash, "batman@gmail.com", "Bruce Wayne");  
    g_hash_table_insert(hash, "ironman@gmail.com", "Tony Stark");
```

➤ 출력 결과

```
    g_hash_table_foreach(hash, (GHFunc)iterator_print, NULL);  
    printf("Wn");  
    g_hash_table_remove(hash, "superman@gmail.com");  
    g_hash_table_foreach(hash, (GHFunc)iterator_print, NULL);  
    printf("Wn");
```

출력

데이터 삭제

```
    g_hash_table_replace(hash, "batman@gmail.com", "Robin");  
    g_hash_table_foreach(hash, (GHFunc)iterator_print, NULL);    printf("Wn");
```

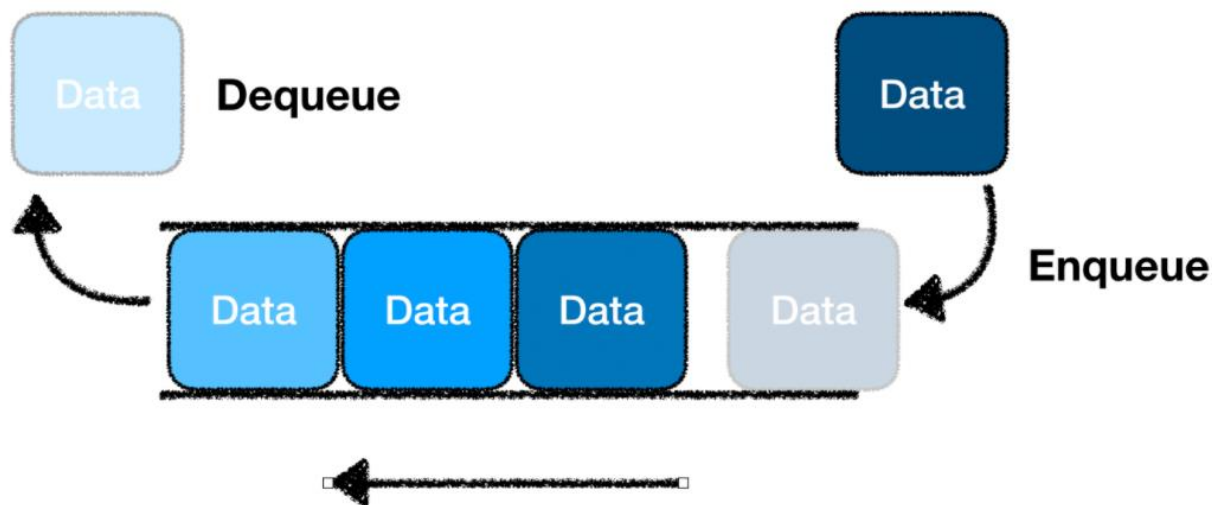
데이터(Value) 변경

```
    g_hash_table_destroy(hash);  
}
```

메모리 해제

```
ironman@gmail.com : Tony Stark  
superman@gmail.com : Clark Kent  
kit@gmail.com : Michael Knight  
knife@gmail.com : Mac Guyver  
batman@gmail.com : Bruce Wayne  
  
ironman@gmail.com : Tony Stark  
kit@gmail.com : Michael Knight  
knife@gmail.com : Mac Guyver  
batman@gmail.com : Bruce Wayne  
  
ironman@gmail.com : Tony Stark  
kit@gmail.com : Michael Knight  
knife@gmail.com : Mac Guyver  
batman@gmail.com : Robin
```

큐(queue)는 컴퓨터의 기본적인 자료 구조의 한가지로, 먼저 집어 넣은 데이터가 먼저 나오는 FIFO(First In First Out)구조로 저장하는 형식을 말한다.



Queue, List, Map, LinkedList 등의 자료구조를 성능과 용도를 고려하여 적절히 활용

Queue 사용

```
Queue<String> numberQ = new LinkedList<>();
// Deque<String> numberQ = new ArrayDeque<>();

numberQ.add("one");
numberQ.add("two");
numberQ.add("three");

System.out.println("Queue Count = " + numberQ.size());
for (String number : numberQ) {
    System.out.println(number);
}
System.out.println("Deque '" + numberQ.poll() + "'");
System.out.println("Peek : " + numberQ.peek());
System.out.println("Contains(W\"threeW\") = " +
    numberQ.contains("three"));

numberQ.clear();
System.out.println("Queue Count = " + numberQ.size());
```

ArrayList 사용

```
ArrayList<String> numberQ = new ArrayList<>();

numberQ.add("one");
numberQ.add("two");
numberQ.add("three");

System.out.println("Queue Count = " + numberQ.size());
for (String number : numberQ) {
    System.out.println(number);
}
System.out.println("Deque : " + numberQ.get(0));
numberQ.remove(0);
System.out.println("Peek : " + numberQ.get(0));
System.out.println("Contains(W\"threeW\") = " +
    numberQ.contains("three"));

numberQ.clear();
System.out.println("Queue Count = " + numberQ.size());
```

실행 결과)

```
Queue Count = 3
one
two
three
Deque : one
Peek : two
Contains("three") = true
Queue Count = 0
```

Queue 사용

```
Queue<string> numberQ = new Queue<string>();
numberQ.Enqueue("one");
numberQ.Enqueue("two");
numberQ.Enqueue("three");

Console.WriteLine("Queue Count = {0}", numberQ.Count);
foreach (string number in numberQ)
{
    Console.WriteLine(number);
}

Console.WriteLine("Deque '{0}'", numberQ.Dequeue());

Console.WriteLine("Peek : {0}", numberQ.Peek());
Console.WriteLine("Contains(W\"threeW\") = {0}",
    numberQ.Contains("three"));

numberQ.Clear();
Console.WriteLine("Queue Count = {0}", numberQ.Count);
```

실행 결과)

```
Queue Count = 3
one
two
three
Deque 'one'
Peek : two
Contains("three") = True
Queue Count = 0
```

List 사용

```
List<string> numberList = new List<string>();
numberList.Add("one");
numberList.Add("two");
numberList.Add("three");

Console.WriteLine("Queue Count = {0}", numberList.Count);
foreach (string number in numberList)
{
    Console.WriteLine(number);
}

Console.WriteLine("Deque '{0}'", numberList[0]);
numberList.RemoveAt(0);
Console.WriteLine("Peek : {0}", numberList[0]);
Console.WriteLine("Contains(W\"threeW\") = {0}",
    numberList.Contains("three"));

numberList.Clear();
Console.WriteLine("Queue Count = {0}", numberList.Count);
```

List를 이용하여 다음 기능을 구현해 보시오.

이름	국어	영어	수학
Kim	80	70	90
Lee	95	89	92
Park	88	70	94
Choi	70	100	92
Moon	100	80	93
Rho	62	99	88
Byun	71	98	77
Kang	62	73	81
Cho	99	85	66
Hong	62	70	92



List_Sample.txt (명령)

List_Sample.txt에는 학생들의 성적 데이터가 저장되어 있다.

1. Console화면에서 'PRINT'를 입력하면 이름 순(오름차순)으로 출력하시오.
2. Console화면에서 'KOREAN', 'ENGLISH', 'MATH'를 입력하면 해당 과목 성적 순(내림차순)으로 출력해 보시오.
(성적이 동일할 경우에는 이름을 오름차순으로 정렬)

* 'QUIT'을 입력하면 프로그램을 종료시킨다.

```

PRINT
Byun    71      98      77
Cho     99      85      66
Choi    70     100      92
Kang    62      73      81
Kim     80      70      90
Lee     95      89      92
Moon    100      80      93
Park    88      97      94
Rho     65      99      88
KOREAN
Moon    100      80      93
Cho     99      85      66
Lee     95      89      92
Park    88      97      94
Kim     80      70      90
Byun    71      98      77
Choi    70     100      92
Rho     65      99      88
Kang    62      73      81
ENGLISH
Choi    70     100      92
Rho     65      99      88
Byun    71      98      77
Park    88      97      94
Lee     95      89      92
Cho     99      85      66
Moon    100      80      93
Kang    62      73      81
Kim     80      70      90
MATH
Park    88      97      94
Moon    100      80      93
Choi    70     100      92
Lee     95      89      92
Kim     80      70      90
Rho     65      99      88
Kang    62      73      81
Byun    71      98      77
Cho     99      85      66
QUIT
    
```

<출력 결과>

DS_Sample2.csv에는 2017년 직원들의 프로젝트 별 투입 MM가 월별로 저장되어 있다.
Map을 이용하여 사원들의 프로젝트 별 MM의 합계와 총 합계를 출력하시오.

월	사번	이름	A프로젝트	B프로젝트	C프로젝트
1	201001	Kim	1	0	0
1	201005	Lee	1	0	0
1	201004	Park	0	1	0
1	201003	Choi	0	1	0
1	201002	Kang	0	0	1
2	201001	Kim	1	0	0
2	201003	Choi	0	1	0
2	201004	Park	0	1	0
2	201003	Choi	0	1	0
...



DS_Sample2

201001	Kim	6	1	0	=>	7
201002	Kang	0.6	0.9	5.5	=>	7
201003	Choi	0.6	6.6	4.8	=>	12
201004	Park	0	10	0	=>	10
201005	Lee	7	1.5	0.5	=>	9

<출력 결과>

입력 메시지(문자열)를 저장하는 Queue들을 작성하시오.

Console로 입력되는 다음 명령어들을 처리해야 함.

- CREATE <Queue Name> <Queue Size>
 - : Queue Name으로 Queue생성, 정상 생성 시 "Queue Created" 출력
 - : Queue Name의 Queue가 이미 존재하는 경우 "Queue Exist" 출력
- ENQUEUE <Queue Name> <Message>
 - : Queue Name의 Queue에 Message저장, 저장 시 고유 Id값을 생성하여 함께 저장
 - : Queue Size개의 데이터가 이미 저장된 경우 "Queue Full"출력, 정상인 경우 "Enqueued" 출력
- DEQUEUE <Queue Name>
 - : Queue Name의 Queue에 가장 먼저 저장된 Message와 Message Id를 출력하고, 해당 메시지 삭제
 - : Queue가 비어 있다면 "Queue Empty" 출력
- GET <Queue Name>
 - : Queue Name의 Queue에 가장 먼저 저장된 Message와 Message Id를 출력
 - : 해당 Message는 Queue에서 삭제되지 않지만, 다시 GET할 수는 없음
- SET <Queue Name> <Message Id>
 - : Queue Name과 Message Id에 해당하는 Message를 다시 GET할 수 있게 세팅
 - : 세팅 성공 시 "Msg Set", 실패 시 "Set Fail"출력
- DEL <Queue Name> <Message Id>
 - : Queue에서 Message Id에 해당하는 Message 삭제, 삭제 성공 시 "Deleted" 출력
 - : 삭제 실패 시 "Not Deleted" 출력

<입력>

```
CREATE Q1 2
ENQUEUE Q1 MSG1_1
ENQUEUE Q1 MSG1_2
CREATE Q2 5
ENQUEUE Q2 MSG2_1
ENQUEUE Q2 MSG2_2
ENQUEUE Q2 MSG2_3
ENQUEUE Q2 MSG2_4
DEQUEUE Q2
DEQUEUE Q2
GET Q1
GET Q1
GET Q1
ENQUEUE Q1 MSG1_3
SET Q1 1
GET Q1
DEL Q1 1
```

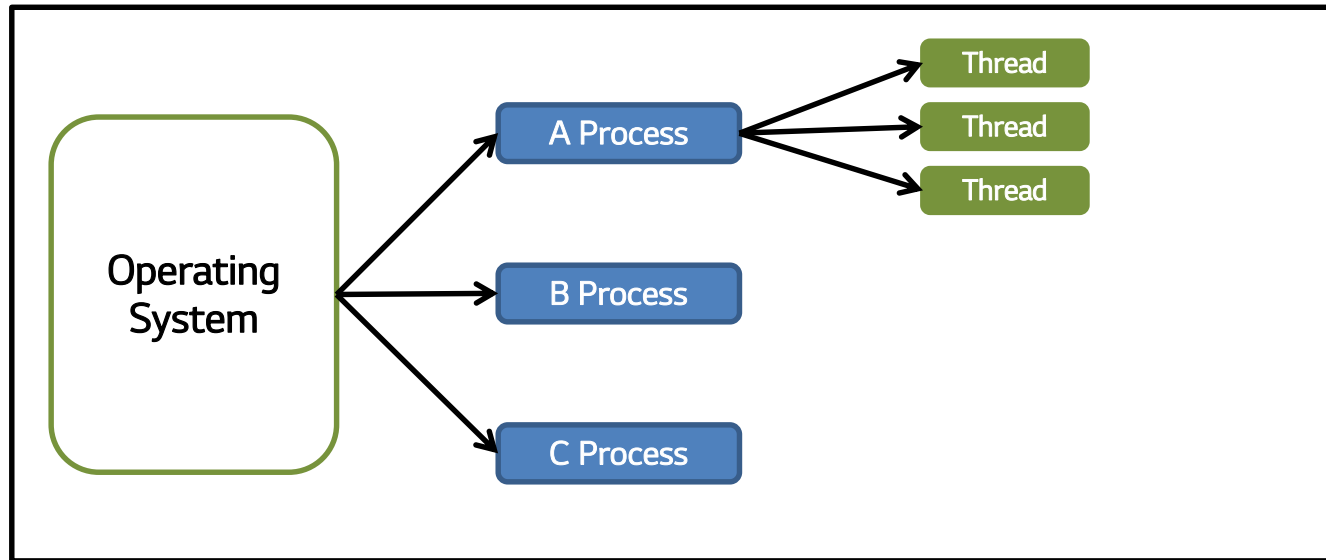
<출력>

```
CREATE Q1 2
Queue Created
ENQUEUE Q1 MSG1_1
Enqueued
ENQUEUE Q1 MSG1_2
Enqueued
CREATE Q2 5
Queue Created
ENQUEUE Q2 MSG2_1
Enqueued
ENQUEUE Q2 MSG2_2
Enqueued
ENQUEUE Q2 MSG2_3
Enqueued
ENQUEUE Q2 MSG2_4
Enqueued
DEQUEUE Q2
MSG2_1(0)
DEQUEUE Q2
MSG2_2(1)
GET Q1
MSG1_1(0)
GET Q1
MSG1_2(1)
GET Q1
No Msg
ENQUEUE Q1 MSG1_3
Queue Full
SET Q1 1
Msg Set
GET Q1
MSG1_2(1)
DEL Q1 1
Deleted
```

3. Process & Thread

- 3.1 Overview
- 3.2 Context Switch (문맥교환)
- 3.3 메모리 공간에서의 Thread
- 3.4 Thread의 장단점
- 3.5 Sample Code
- 3.6 실습

Process & Thread



1) Process

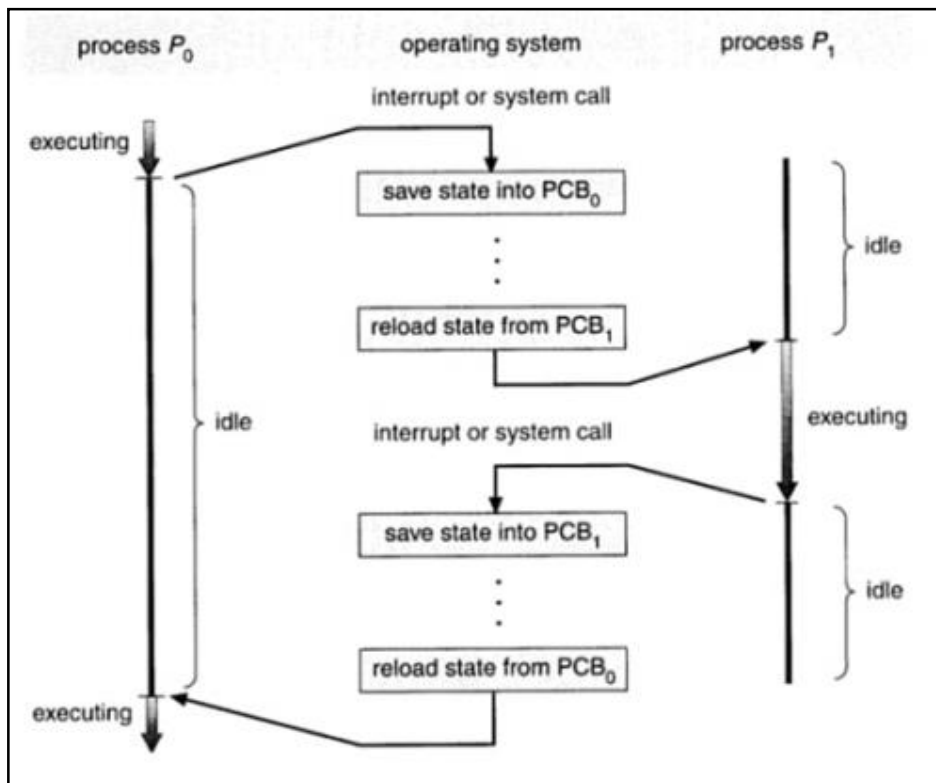
- 운영 체제가 바라보는 일의 단위

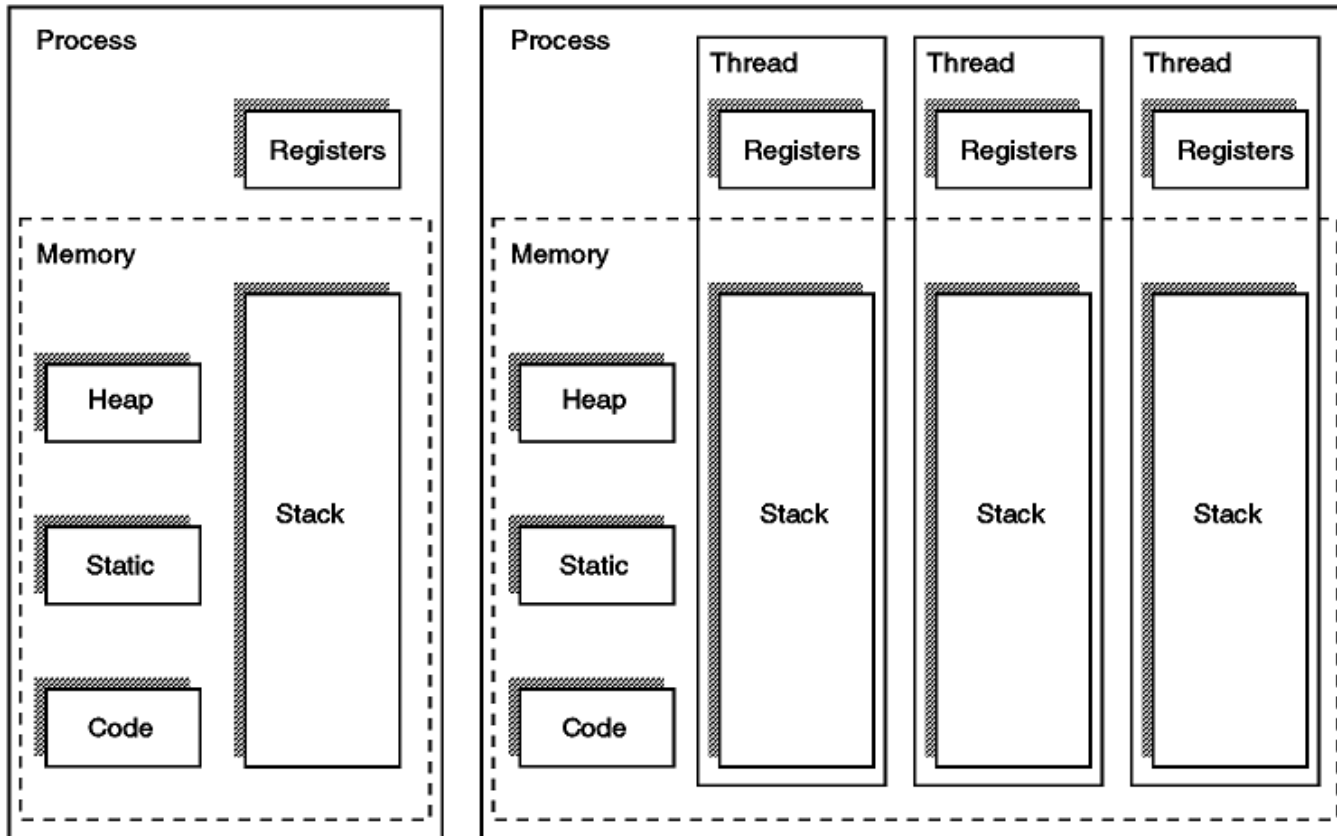
2) Thread

- Process 내에서 다시 나누어지는 일의 단위

➤ 실행하고 있는 프로그램 혹은 Process를 교환하는 것.

실행에 이용되는 프로그램 카운터, 스택 포인터, 레지스터 등의 내용을 넣어두고,
거기까지 실행해 간 Process 등의 실행에 필요한 정보를 보존하여 다음 실행을 시작하는
Process 등의 정보를 이용할 수 있게 하는 조작





각각의 Process가 독립적인 stack, heap, code, data 영역을 가진 반면에, 한 Process에 속한 Thread는 stack 영역을 제외한 메모리 영역은 공유한다.

Thread는 data, heap 영역을 공유하기 때문에 IPC 없이도 Thread 간의 통신이 가능하다.

Thread는 공유하고 있는 메모리 영역 덕분에 컨텍스트 스위칭 때문에 발생하는 오버헤드(overhead)가 Process에 비해 작다.

장점

- 응답성 향상

응용 프로그램의 일부분이 봉쇄 되거나 긴 작업을 수행하는 경우에도 프로그램의 수행이 계속 되는 것을 허용 함으로써 사용자에게 대한 응답성을 증가 시킨다.

Ex) 이미지 로딩 시 사용자 입력 처리

- 손쉬운 자원 공유

Thread는 자동적으로 그들이 속한 Process의 자원들과 메모리를 공유 한다.

- 경제성

Process 생성을 위해 메모리와 자원을 할당하는 것은 비용이 많이 든다. Thread는 자신이 속한 Process의 자원들을 공유하기 때문에 Thread를 생성하고, Thread간 문맥 교환하는 것이 더 경제적이다.

단점

- 안정성

멀티 Process는 하나의 Process에서 문제가 발생해도 다른 Process에 영향을 거의 미치지 않지만, 멀티 Thread는 하나의 Process에서 실행되기 때문에, 특정 Thread에서 발생한 문제가 다른 Thread까지 영향을 끼칠 수 있다.

- 동기화 이슈

공유 자원에 Thread가 동시에 접근하여 연산을 일으키는 경우에는 잘못된 연산 가능. Thread 간에 동기화 필요.

- 난해한 프로그래밍 기술

멀티 Process 방식에 비해서 코드의 흐름을 이해하고 결과를 예측하기 까다롭다.

문제 발생시 어떤 Thread에서 문제가 발생했는지 확인하기 어렵다.

프로세스 실행 커맨드

```
public static String getProcessOutput(List<String> cmdList)
    throws IOException, InterruptedException {
    ProcessBuilder builder = new ProcessBuilder(cmdList);
    Process process = builder.start();
    InputStream psout = process.getInputStream();
    byte[] buffer = new byte[1024];
    psout.read(buffer);
    return (new String(buffer));
}
```

프로세스 실행

출력 가져오기

```
public static void main(String[] args) throws IOException, InterruptedException {
    String output = getProcessOutput(Arrays.asList("add_2sec.exe", "2", "3"));
    System.out.println(output);
}
```

프로세스 이름

아규먼트


```
static string getProcessOutput(string fileName, string args)
{
```

```
    ProcessStartInfo start = new ProcessStartInfo();
```

```
    start.FileName = fileName;
```

```
    start.UseShellExecute = false;
```

프로세스 이름

```
    start.RedirectStandardOutput = true;
```

```
    start.CreateNoWindow = true;
```

```
    start.Arguments = args;
```

아규먼트

```
    Process process = Process.Start(start);
```

프로세스 실행

```
    StreamReader reader = process.StandardOutput;
```

```
    return reader.ReadLine();
```

출력 가져오기

```
}
```

```
static void Main(string[] args)
```

```
{
```

```
    string output = getProcessOutput("add_2sec.exe", "2 3");
```

```
    Console.WriteLine(output);
```

```
}
```

```
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>

void * pThreadFunc(void *arg)
{
    printf("Thread is running...");
}

int main( )
{
    pthread_t thread1;
    int nRet;

    if ((nRet = pthread_create(&thread1, NULL, pThreadFunc, NULL)) != 0)
    {
        perror("pthread create error!\n");
        return -1;
    }

    pthread_join(thread1, NULL);

    return 0;
}
```

❖ 참고 사이트

<https://linuxprograms.wordpress.com/2007/12/29/threads-programming-in-linux-examples/>

◆ Thread class 상속

```
class ThreadClass extends Thread {
    public void run() {
        System.out.print("Thread is
running...");
    }
}

public class ThreadSample {
    public static void main(String[] args) {
        ThreadClass tc1 = new ThreadClass();

        tc1.start();

        try {
            tc1.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

◆ Runnable Interface 이용

```
class ThreadClass implements Runnable {
    public void run(){
        System.out.println("thread is
running...");
    }
}

public class ThreadRunnable {
    public static void main(String[] args) {
        ThreadClass m1=new ThreadClass();
        Thread t1 =new Thread(m1);
        t1.start();

        try {
            t1.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

❖ 참고 사이트

<https://www.javatpoint.com/creating-thread>

◆ 여러 방식으로 활용

```
public class ThreadTest {
    public static void main(String[] args) {

        Thread t1 = new Thread( new Runnable () {
            @Override
            public void run(){
                System.out.println(Thread.currentThread().getName() + " is running...");
            }
        });
        t1.start();

        // Lambda Runnable 1
        Runnable taskR = () -> {
            System.out.println(Thread.currentThread().getName() + " is running");
        };
        new Thread(taskR).start();

        // Lambda Runnable 2
        new Thread(() -> {
            System.out.println(Thread.currentThread().getName() + " is running");
        }).start();
    }
}
```

```
class ThreadSample
{
    public class Worker
    {
        // This method will be called when the thread is started.
        public void DoWork()
        {
            Console.WriteLine("Thread is running...");
        }
    }

    static void Main(string[] args)
    {
        // Create the thread object. This does not start the thread.
        Worker workerObject1 = new Worker();
        Thread workerThread1 = new Thread(workerObject1.DoWork);

        // Start the worker thread.
        workerThread1.Start();

        // Use the Join method to block the current thread
        // until the object's thread terminates.
        workerThread1.Join();
    }
}
```

❖ 참고 사이트

[https://msdn.microsoft.com/ko-kr/library/7a2f3ay4\(v=vs.90\).aspx](https://msdn.microsoft.com/ko-kr/library/7a2f3ay4(v=vs.90).aspx)

```
#include <windows.h>
#include <stdio.h>
#include <process.h>

unsigned __stdcall ThreadFunc(void* pArguments)
{
    printf("Thread is running...");

    _endthreadex(0);
    return 0;
}

int main()
{
    HANDLE hThread1;
    unsigned int threadID1;

    hThread1 = (HANDLE)_beginthreadex(NULL, 0, &ThreadFunc, NULL, 0, &threadID1);

    WaitForSingleObject(hThread1, INFINITE);

    CloseHandle(hThread1); // 반드시 핸들을 닫아줘야 함.

    return 0;
}
```

❖ 참고 사이트

<https://msdn.microsoft.com/ko-kr/library/kdzttdcb.aspx>

Thread 2개를 만든 후, Main함수와 Thread 2개에서 동시에 0부터 9까지 출력하시오.
어디서 출력하였는지 구분할 수 있게 숫자 앞에 [Main], [Thread1], [Thread2] 표시하시오.

```
[Main] 0
[Thread1] 0
[Thread2] 0
[Thread1] 1
[Thread2] 1
[Main] 1
[Thread1] 2
[Thread2] 2
[Main] 2
[Main] 3
[Thread1] 3
[Thread2] 3
[Thread2] 4
[Main] 4
[Thread1] 4
[Main] 5
[Thread2] 5
[Thread1] 5
[Thread1] 6
[Main] 6
[Thread2] 6
[Thread1] 7
[Thread2] 7
[Main] 7
[Thread1] 8
[Main] 8
[Thread2] 8
[Thread1] 9
[Thread2] 9
[Main] 9
```

NUM.TXT에 저장되어 있는 5쌍의 숫자들을 add_2sec.exe를 통해 덧셈을 실행시킨 후, 각각의 결과들을 모두 출력하시오.

[조건]

- 전체 실행 시간은 5초 이내
- 결과의 출력 순서는 상관없음
- 실행 시작과 끝에 현재시각 출력

* add_2sec.exe [num1] [num2] 실행하면, 2초 후 num1 + num2의 값을 출력함.

[입력] NUM.TXT

1 2
3 4
5 6
7 8
9 10

[출력]

Start - Tue Mar 15 18:30:23 KST 2022
3 + 4 = 7
5 + 6 = 11
1 + 2 = 3
9 + 10 = 19
7 + 8 = 15
End - Tue Mar 15 18:30:25 KST 2022

4. 동기화 (Synchronization)

4.1 Overview

4.2 동기화 개념

4.3 동기화 용어

4.4 Mutex?

4.5 Mutex & Semaphore

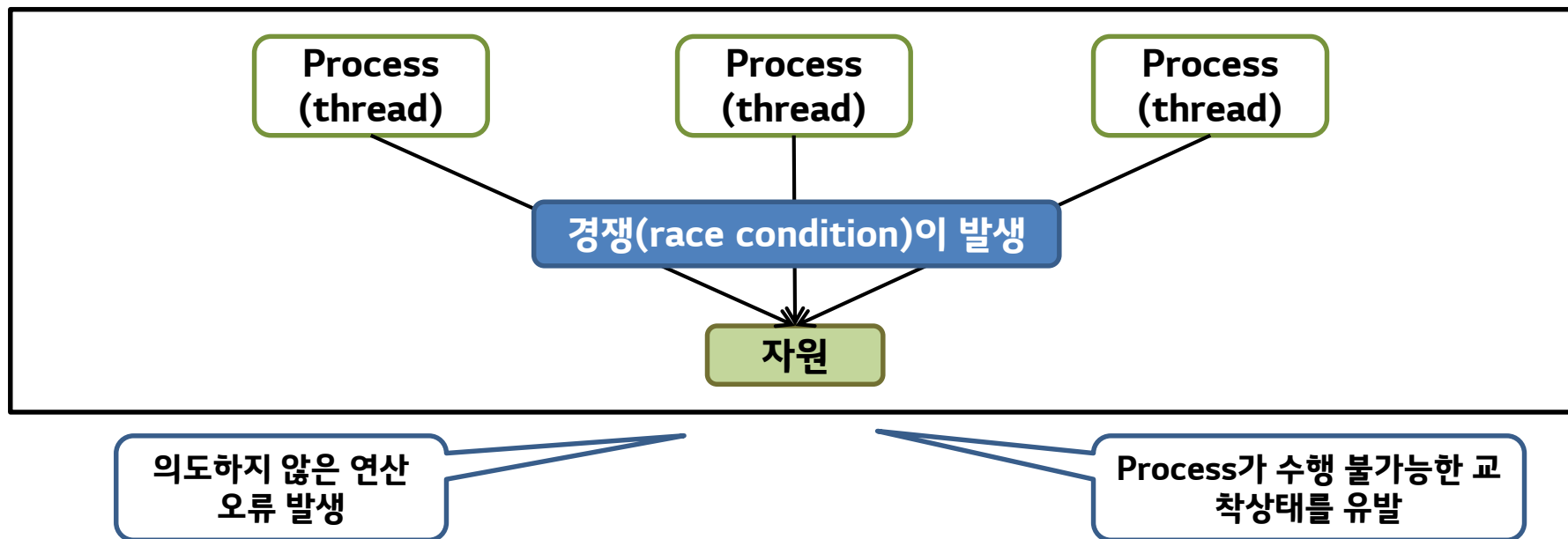
4.6 Mutex 동작 방식

4.7 실습

동기화?



- 다중 CPU 환경, 다중 process 또는 다중 thread 환경에서 동작하는 process, thread들이 공통된 자원에 접근하고자 할 경우에 발생하는 **순서와 동시성을 배제하기 위한 목적으로 설계된 개념**
- 동기화는 2개 이상의 프로그램이 서로 상호간에 실행하기 위한 규칙이며, 위반 시 중대한 오류가 발생할 수 있다.



구분	설 명
경쟁 조건 (Race Condition)	2개 이상의 Process(또는 thread)들이 공유 자원에 읽기/쓰기를 하고, 그 결과가 실행 순서에 따라 달라질 수 있는 현상
임계 영역 (Critical Section)	여러 Process(또는 thread)가 동시에 접근하는 공유 자원
상호 배제 (Mutual Exclusion)	경쟁조건을 피하기 위해 한 Process(또는 thread)가 공유 자원을 사용하고 있을 때, 다른 Process(또는 thread)는 사용하지 못하게 배제하는 것



◆ Mutex 정의

- Mutual Exclusion 으로 우리말로 해석하면 “상호 배제”라고 한다.
- Critical Section을 가진 Thread들의 running time이 겹치지 않게, 각각 단독으로 실행 되게 하는 기술이다.

◆ 상호배제 구간

- Thread가 공유데이터를 사용하는 경우, 다른 thread가 이를 사용하는 것을 막는 구간이다.
- 구간을 정의하지 않으면 높은 우선순위의 thread가 참조하여 잘못된 결과를 만들어낼 수 있다.

◆ 행동

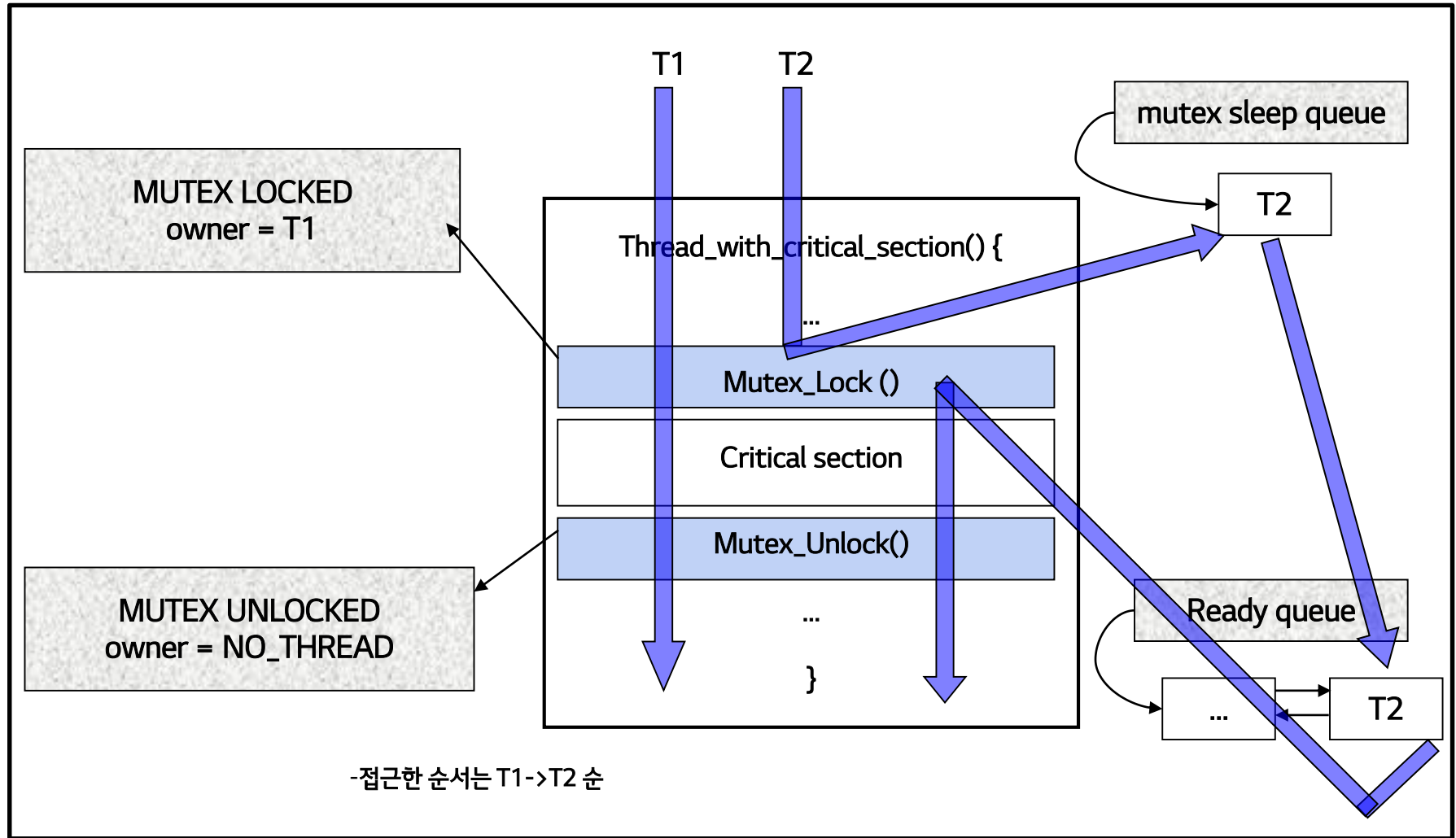
- 상호배제 구간(임계영역)에 들어가기 전에 mutex 변수를 잠그고(lock), 구간이 끝나면 풀어(unlock)준다.

◆ Mutex

- 실행프로그램(task) 간의 상호배제(mutual exclusion) 구간을 정의하여 동작하는 동기화 기법이다. (즉 임계영역에 접근하여 작업하는 thread가 동일한 자원을 다른 thread가 사용하지 못하도록 하는 기법)
- Mutex라는 공용 변수를 사용해야 하고, lock에 소유자를 따져야 하는 방식이다.

◆ Semaphore

- Mutex와 비슷한 “binary semaphore 방식” 과 “수를 계수하는 counting semaphore 방식” 이 있다.
둘 다 value라는 공용 변수를 사용하고 value의 값을 '+' 하거나 '-' 하는 방법으로 count하여 자원에 접근여부를 결정한다.
- Mutex 처럼 lock 소유자라는 개념이 없으며 단순한 연산 (+,-) 으로 자원사용여부를 판단하는 방식이다.



-접근한 순서는 T1->T2 순

```
class ThreadClass implements Runnable{
    static ReentrantLock lock = new ReentrantLock();
    public void run(){
        lock.lock();
        try {
            SaveFile(data);
        }
        finally {
            lock.unlock();
        }
    }
}
```

```
public class ThreadRunnable {
    public static void main(String[] args) {
        Thread t1 = new Thread(new ThreadClass());
        t1.start();

        try {
            t1.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

```
public synchronized void SaveFile(String data)
{
    ...
}
```

❖ 참고 사이트

<http://winterbe.com/posts/2015/04/30/java8-concurrency-tutorial-synchronized-locks-examples/>


```
class ThreadSample
{
    public class Worker
    {
        private static Mutex mut = new Mutex();
        // This method will be called when the thread is started.
        public void DoWork()
        {
            mut.WaitOne();
            SaveFile(data);
            mut.ReleaseMutex();
        }
    }

    static void Main(string[] args)
    {
        Worker workerObject1 = new Worker();
        Thread workerThread1 = new Thread(workerObject1.DoWork);

        workerThread1.Start();

        workerThread1.Join();
    }
}
```

❖ 참고 사이트

[https://msdn.microsoft.com/ko-kr/library/system.threading.mutex\(v=vs.110\).aspx?cs-save-lang=1&cs-lang=csharp#code-snippet-1](https://msdn.microsoft.com/ko-kr/library/system.threading.mutex(v=vs.110).aspx?cs-save-lang=1&cs-lang=csharp#code-snippet-1)

```
pthread_mutex_t mutex_var;

void * pThreadFunc(void *arg)
{
    .....

    pthread_mutex_lock(&mutex_var);
    SaveFile(data);
    pthread_mutex_unlock(&mutex_var);

    .....
}

int main( )
{
    pthread_mutex_init(&mutex_var, NULL);

    .....
    pthread_mutex_lock(&mutex_var);
    SaveFile(data);
    pthread_mutex_unlock(&mutex_var);

    return 0;
}
```

❖ 참고 사이트

<https://github.com/angrave/SystemProgramming/wiki/Synchronization,-Part-1:-Mutex-Locks>

```
CRITICAL_SECTION CriticalSection;
```

```
unsigned __stdcall ThreadFunc(void* pArguments)
{
    .....
    EnterCriticalSection(&CriticalSection);
    SaveFile(data);
    LeaveCriticalSection(&CriticalSection);
    .....
}
```

```
int main()
{
    .....

    InitializeCriticalSectionAndSpinCount(&CriticalSection, 0);

    .....

    DeleteCriticalSection(&CriticalSection);

    return 0;
}
```

❖ 참고 사이트

[https://msdn.microsoft.com/ko-kr/library/windows/desktop/ms686908\(v=vs.85\).aspx](https://msdn.microsoft.com/ko-kr/library/windows/desktop/ms686908(v=vs.85).aspx)

Mutex를 사용하여 Main과 2개의 Thread함수에서 다음과 같이 1~30까지 숫자를 연속으로 출력하게 하시오.

```
Thread1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
Thread2
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
Main
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
```

<Mutex 사용>

```
[Thread1]
1 [Thread2]
[Main]
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 2 3 4 5 6 7 8 9 10 11 12 1 2 3 4 5 6 7 8 24 25 26 27 28 29 30
13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
```

<Mutex 사용하지 않음>

Ch 5. Encryption/Decryption

5.1 Overview

5.2 Base64

5.3 AES

5.4 Cryptographic hash function

5.5 Code Obfuscation

5.6 실습

Encoding

사용성을 위해 데이터의 형태나 형식을 변환하는 처리나 처리 방식. 동일한 알고리즘을 사용하여 되돌릴 수 있음. 키가 사용되지 않음.

예) ASCII, UNICODE, URL encoding, Base64

Encryption

데이터의 기밀성을 유지하기 위해 사용되며 원래의 값으로 되돌리기 위해서는 비밀 유지가 필요한 키를 사용해야 함.

예) AES, Blowfish, RSA

Hashing

임의의 길이의 데이터를 고정된 길이의 데이터로 매핑.

암호화 해시 함수(cryptographic hash function)는 해시 값으로부터 원래의 입력값과의 관계를 찾기 어려운 성질을 갖게 함.

예) SHA-3, MD5, etc.

Obfuscation (난독화)

리버스 엔지니어링에 의해 소스 코드가 유출되는 것을 막기 위해서 사람들이 쉽게 이해할 수 없게 코드를 변환하는 것.

예) Javascript obfuscator, Proguard

8비트 이진 데이터(예를 들어 실행 파일이나, ZIP 파일 등)를 문자 코드에 영향을 받지 않는 공통 ASCII 영역의 문자들만으로 이루어진 일련의 문자열로 바꾸는 인코딩 방식.

원래 Base 64를 글자 그대로 번역하여 보면 64진법이란 뜻.

64가 2의 제곱수($64 = 2^6$)이며, 2의 제곱수들에 기반한 진법들 중에서 화면에 표시되는 ASCII 문자들을 써서 표현할 수 있는 가장 큰 진법.

ASCII 문자들이 128개가 되지 않는 까닭에 이 인코딩은 전자 메일을 통한 이진 데이터 전송 등에 많이 쓰이고 있음.

인코딩된 문자열은 알파벳 대소문자와 숫자, 그리고 "+", "/" 기호 64개로 이루어지며, "="는 끝을 알리는 코드로 쓰임.

Ex) This is a Base64 test.

→ VGhpcyBpcyBhIEJhc2U2NCB0ZXN0Lg==

Encoding 테스트 : <https://www.base64encode.org/>

Decoding 테스트 : <https://www.base64decode.org/>

◆ Advanced Encryption Standard (고급 암호화 표준)

- 미국 정부 표준으로 지정된 블록 암호 형식. 이전의 DES를 대체하며, 미국 표준 기술 연구소(NIST)가 5년의 표준화 과정을 거쳐 2001년 11월 26일에 연방 정보 처리 표준(FIPS 197)으로 발표함. 2002년 5월 26일부터 표준으로 효력을 발휘하기 시작.
- 벨기에 암호학자인 존 대먼과 빈센트 라이먼에 의해서 만들어졌으며, 처음에는 두 사람의 이름을 합해서 레인달(Rijndael, [rɛindaː l])이라는 이름을 사용.
- 128비트의 블록 크기를 가지고 있고 128, 160, 192, 224, 256비트 등 128비트 이상의 모든 32의 배수 비트 길이의 키를 사용할 수 있으며, 미국 표준으로 인정받은 것은 128비트임.

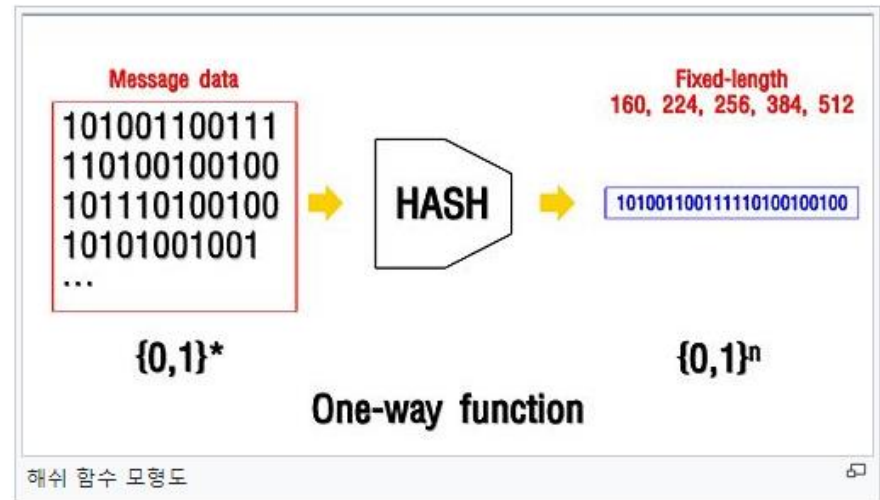
◆ DES (Data Encryption Standard)

- 블록 암호의 일종으로, 미국 NBS (National Bureau of Standards, 현재 NIST)에서 국가 표준으로 정한 암호. DES는 대칭키 암호이며, 56비트의 키를 사용. DES는 현재 취약한 것으로 알려져 있음

Encryption 테스트 : <https://www.browserling.com/tools/aes-encrypt>

Decryption 테스트 : <https://www.browserling.com/tools/aes-decrypt>

암호학적 해시함수(cryptographic hash function)는 해시 함수의 일종으로, 임의의 길이를 갖는 메시지를 입력하여 고정된 길이의 해시값을 출력하는 함수이다. 각 메시지마다 해시값이 다르기 때문에 해시함수는 메시지의 무결성을 확인하는 방법으로 메시지의 내용이 변경되지 않았다는 것을 보장해준다. 또한, 일방향 함수를 포함하고 있기 때문에 해시값에서 원문을 재현할 수는 없고 같은 해시값을 가진 다른 데이터를 작성하는 것도 극히 어렵다. 이런 특성을 이용해서 통신의 암호화 보조수단이나 사용자 인증, 디지털 서명 등에 응용되고 있다.



해시함수의 종류

- MD(Message Digest) : 미국 MIT의 로널드 리베스트 교수가 공개키 기반 구조를 만들기 위해 RSA와 함께 개발 → MD2, MD4, MD5
- SHA(Secure Hash Algorithm) : 미국 NSA에 의해 만들어졌다. MD4가 발전한 형태이다. MD5보다 조금 느리지만 좀더 안전한 것으로 알려져 있다. → SHA-1, SHA-2 (SHA-256, SHA-384, SHA-512)

SHA256 Hash Generator <http://passwordsgenerator.net/sha256-hash-generator/>

```
//detect which browser is used
var detect = navigator.userAgent.toLowerCase();
var OS,browser,version,total,thestring;
if (checkIt('konqueror'))
{
    browser = "Konqueror";
    OS = "Linux";
}
else if (checkIt('opera')) browser = "Opera"
else if (checkIt('msie')) browser = "Internet Explorer"
else if (!checkIt('compatible'))
{
    browser = "Netscape Navigator"
    version = detect.charAt(8);
}
else browser = "An unknown browser";
//version of browser
if (!version) version = detect.charAt(place + thestring.length);
//client OS
if (!OS)
{
    if (checkIt('linux')) OS = "Linux";
    else if (checkIt('x11')) OS = "Unix";
    else if (checkIt('mac')) OS = "Mac"
    else if (checkIt('win')) OS = "Windows"
    else OS = "an unknown operating system";
}
//check the string
function checkIt(string)
{
    place = detect.indexOf(string) + 1;
    thestring = string;
    return place;
}
```

Original Code



```
var e=navigator.userAgent.toLowerCase();var f,b,c,total,d;if(a('konqueror')){b="Konqueror";f="Linux";};else if(a('opera'))b="Opera";else if(a('msie'))b="Internet Explorer";else if(!a('compatible')){b="Netscape Navigator";c=e.charAt(8);};else b="An unknown browser";if(!c)c=e.charAt(g+d.length);if(!f){if(a('linux'))f="Linux";else if(a('x11'))f="Unix";else if(a('mac'))f="Mac";else if(a('win'))f="Windows";else f="an unknown operating system";};function a(string){g=e.indexOf(string)+1;d=string;return g;};
```

Obfuscated Code

코드 난독화의 목적

→ 역공학(Reverse-Engineering)에 의해 소스 코드가 노출되는 것을 방지

- 코드 난독화는 바이러스나 웹 제작자들에 의해 처음 연구됨
- 가장 기본적인 코드 난독화는 바이너리에서 심볼 정보를 제거하거나 변경하는 것
 - C/C++인 경우는 심볼 정보를 지우고, 자바의 경우는 심볼 정보가 프로그램 수행에 필요하므로 이름을 변경
- 기타 난독화 방법
 - 필요 이상으로 복잡한 코드를 만들거나, 아무 것도 하지 않는 코드 삽입
 - 코드를 여기저기로 복사하고 옮김
 - 데이터를 알아보기 힘들게 인코딩함

◆ Base64

```
void Base64Sample(String TestString) throws UnsupportedEncodingException  
{
```

```
    Encoder encoder = Base64.getEncoder();
```

Base64 인코더

```
    String encodedString = encoder.encodeToString(TestString.getBytes("UTF-8"));
```

```
    System.out.println(encodedString);
```

Base64 인코딩 후 문자열로 변환

```
    Decoder decoder = Base64.getDecoder();
```

Base64 디코더

```
    byte[] decodedBytes = decoder.decode(encodedString);
```

Base64 디코딩

```
    String decodedString = new String(decodedBytes, "UTF-8");
```

디코딩 결과를 문자열로 변환

```
    System.out.println(decodedString);
```

```
}
```

➤ 실행 예

입력 : This is a Base64 test.

출력 : VGhpcyBpcyBhIEJhc2U2NCB0ZXN0Lg==

◆ Base64

```
void Base64Sample(string str)
{
```

```
    byte[] byteStr = System.Text.Encoding.UTF8.GetBytes(str);
    string encodedStr;
    byte[] decodedBytes;
```

문자열을 Base64로 인코딩

```
    encodedStr = Convert.ToBase64String(byteStr);
```

인코딩 결과를 문자열로 변환

```
    Console.WriteLine(encodedStr);
```

```
    decodedBytes = Convert.FromBase64String(encodedStr);
```

문자열을 Base64로 디코딩

```
    Console.WriteLine(Encoding.Default.GetString(decodedBytes));
```

디코딩 결과를 문자열로 변환 후
출력

➤ 실행 예

입력 : This is a Base64 test.

출력 : VGhpcyBpcyBhIEJhc2U2NCB0ZXN0Lg==

◆ Base64

```
void Base64Sample(char * str)
{
```

```
    char encodedStr[100];
```

```
    char decodedStr[100];
```

```
    int nDestLen;
```

```
    Base64Encode((const BYTE *)str, strlen(str), encodedStr,
                  &nDestLen, ATL_BASE64_FLAG_NOCLRF);
```

문자열을 Base64로 인코딩

```
    encodedStr[nDestLen] = 0;
```

```
    printf("%s\n", encodedStr);
```

```
    Base64Decode((LPCSTR)encodedStr, strlen(encodedStr),
                  (BYTE*)decodedStr, &nDestLen);
```

문자열을 Base64로 디코딩

```
    decodedStr[nDestLen] = 0;
```

```
    printf("%s\n", decodedStr);
```

```
}
```

➤ 실행 예

입력 : This is a Base64 test.

출력 : VGhpcyBpcyBhIEJhc2U2NCB0ZXN0Lg==

◆ Base64

❖ glib 사용

```
void Base64Test(char * szStr)
{
```

```
    gchar * gEncodedStr;
    gchar * gDecodedStr;
    gsize decSize = 0;
```

문자열을 Base64로 인코딩

```
    gEncodedStr = g_base64_encode((const gchar *)szStr, strlen(szStr));
```

```
    printf("%s\n", gEncodedStr);
```

```
    gDecodedStr = (gchar *)g_base64_decode((const gchar *)gEncodedStr, &decSize);
```

```
    printf("%s\n", gDecodedStr);
```

문자열을 Base64로 디코딩

```
    g_free(gEncodedStr);
    g_free(gDecodedStr);
```

인코딩, 디코딩에 사용한 문자열 메모리 해제

```
}
```

➤ 실행 예

입력 : This is a Base64 test.

출력 : VGhpicyBpcyBhIEJhc2U2NCB0ZXN0Lg==

◆ SHA-256

void SHA256(String input) throws NoSuchAlgorithmException

{

 MessageDigest mDigest = MessageDigest.getInstance("SHA-256");

SHA-256 MessageDigest
인스턴스 생성

 byte[] result = mDigest.digest(input.getBytes());

SHA-256 암호화

 StringBuffer sb = new StringBuffer();

 for (int i = 0; i < result.length; i++) {

 sb.append(Integer.toString((result[i] & 0xFF) + 0x100, 16).substring(1));

 }

암호화 결과 문자열 변환

 System.out.println(sb.toString());

}

➤ 실행 예

입력 : 1234

출력 : 03AC674216F3E15C761EE1A5E255F067953623C8B388B4459E13F978D7C846F4

◆ SHA-256

```
void SHA256Sample(string strInput)
{
```

```
    byte[] hashValue;
```

```
    byte[] byteInput = System.Text.Encoding.UTF8.GetBytes(strInput);
```

입력 문자열 바이트 배열로 변환

```
    SHA256 mySHA256 = SHA256Managed.Create();
```

SHA-256 클래스 인스턴스 생성

```
    hashValue = mySHA256.ComputeHash(byteInput);
```

SHA-256 암호화

```
    for (int i=0; i<hashValue.Length; i++)
```

```
        Console.WriteLine(String.Format("{0:X2}", hashValue[i]));
```

암호화 결과 출력

➤ 실행 예

입력 : 1234

출력 : 03AC674216F3E15C761EE1A5E255F067953623C8B388B4459E13F978D7C846F4

◆ SHA-256

❖ clr 사용

```
void SHA256Sample(std::string str)
{
```

```
    SHA256 ^ mySHA256 = SHA256Managed::Create();
    array<Byte>^ hashValue;
```

SHA-256 클래스 인스턴스 생성

```
    array< Byte >^ input = gcnew array< Byte >(str.length());
    Marshal::Copy((IntPtr)(char *)str.c_str(), input, 0, str.length());
```

입력 문자열 array<Byte>로 변환

```
    hashValue = mySHA256->ComputeHash(input);
```

SHA-256 암호화

```
    for (int i = 0; i < hashValue->Length; i++)
        Console::Write(String::Format("{0:X2}", hashValue[i]));
```

암호화 결과 출력

```
}
```

➤ 실행 예

입력 : 1234

출력 : 03AC674216F3E15C761EE1A5E255F067953623C8B388B4459E13F978D7C846F4

◆ SHA-256

❖ Open Source (mbed) 사용

```
void SHA256Test(char * szStr)
{
```

```
    unsigned char outBuf[32] = {0,};
    int i = 0;
```

Output Buffer 설정

```
    mbedtls_sha256( szStr, strlen(szStr), outBuf, 0 );
```

SHA-256 암호화

```
    for (i=0; i<32; i++)
```

```
    {
```

```
        printf("%02X",outBuf[i]);
```

암호화 결과 출력

```
    }
```

```
}
```

➤ 실행 예

입력 : 1234

출력 : 03AC674216F3E15C761EE1A5E255F067953623C8B388B4459E13F978D7C846F4

1. 문자열을 입력 받아 Base64로 Encoding한 값을 출력하고, 그 값을 다시 Decoding하여 입력한 값과 동일한지 확인해 보시오.

Ex) 'This is a Base64 test.' → 'VGhpcyBpcyBhIEJhc2U2NCB0ZXN0Lg==' → 'This is a Base64 test.'

2. 1번에서 입력 받은 값을 SHA-256으로 Encryption 해서 결과를 출력해 보시오.

Ex) '1234' -> '03AC674216F3E15C761EE1A5E255F067953623C8B388B4459E13F978D7C846F4'

SHA-256 참고 사이트)

C#

[https://msdn.microsoft.com/ko-kr/library/system.security.cryptography.sha256\(v=vs.110\).aspx?cs-save-lang=1&cs-lang=csharp#code-snippet-1](https://msdn.microsoft.com/ko-kr/library/system.security.cryptography.sha256(v=vs.110).aspx?cs-save-lang=1&cs-lang=csharp#code-snippet-1)

JAVA

<https://gist.github.com/avilches/750151>

C언어

<https://tls.mbed.org/sha-256-source-code> (소스)

C++

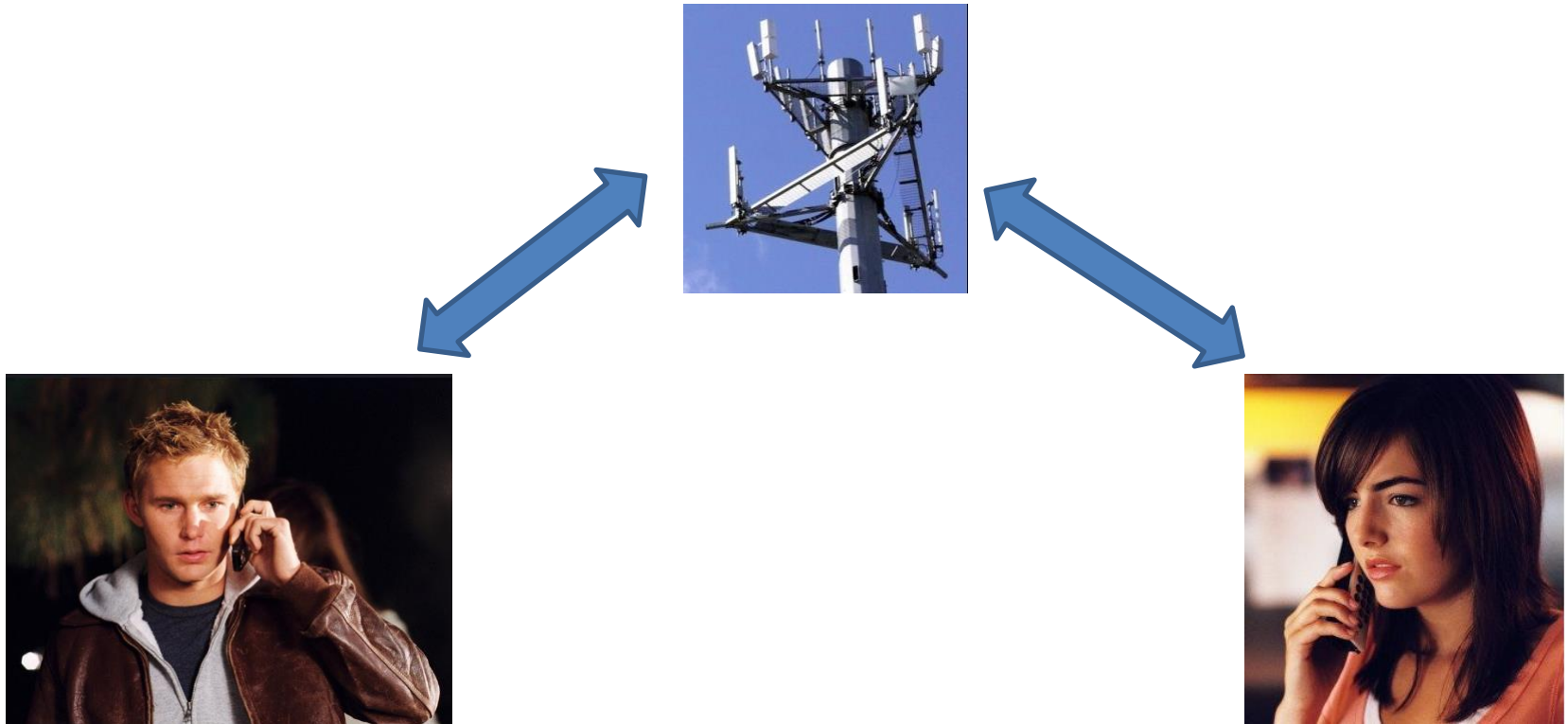
[https://msdn.microsoft.com/ko-kr/library/system.security.cryptography.sha256\(v=vs.110\).aspx?cs-save-lang=1&cs-lang=cpp#code-snippet-2](https://msdn.microsoft.com/ko-kr/library/system.security.cryptography.sha256(v=vs.110).aspx?cs-save-lang=1&cs-lang=cpp#code-snippet-2) (clr 옵션 필요)

https://sourceforge.net/projects/hashlib2plus/?source=typ_redirect (소스)

Ch 6. 네트워크 프로그래밍

- 6.1 Intro
- 6.2 Network Model
- 6.3 Internet Address
- 6.4 TCP
- 6.5 UDP
- 6.6 TCP vs UDP
- 6.7 소켓이란?
- 6.8 TCP/IP 소켓통신
- 6.9 Sample Code
- 6.10 실습

전화를 걸고 싶은데, 무엇이 필요할까요?



➤ 일련의 통신 과정을 여러 계층으로 분리한 개념

- 각 계층은 네트워크 통신을 위한 전체 기능 중에서 특정 기능만을 책임지고, 인접한 계층끼리만 상호 동작하도록 설계됨

➤ 계층 구조의 장점

- 네트워크 통신을 단순화
- 네트워크 구성요소를 표준화하여 여러 업체의 장비 개발과 지원 가능
- 서로 다른 유형의 네트워크 하드웨어/소프트웨어 사이의 통신 가능
- 한 계층의 변경이 다른 계층에 영향을 미치지 않으므로 계층 기능의 개발 속도 향상

➤ 대표적인 Network Model

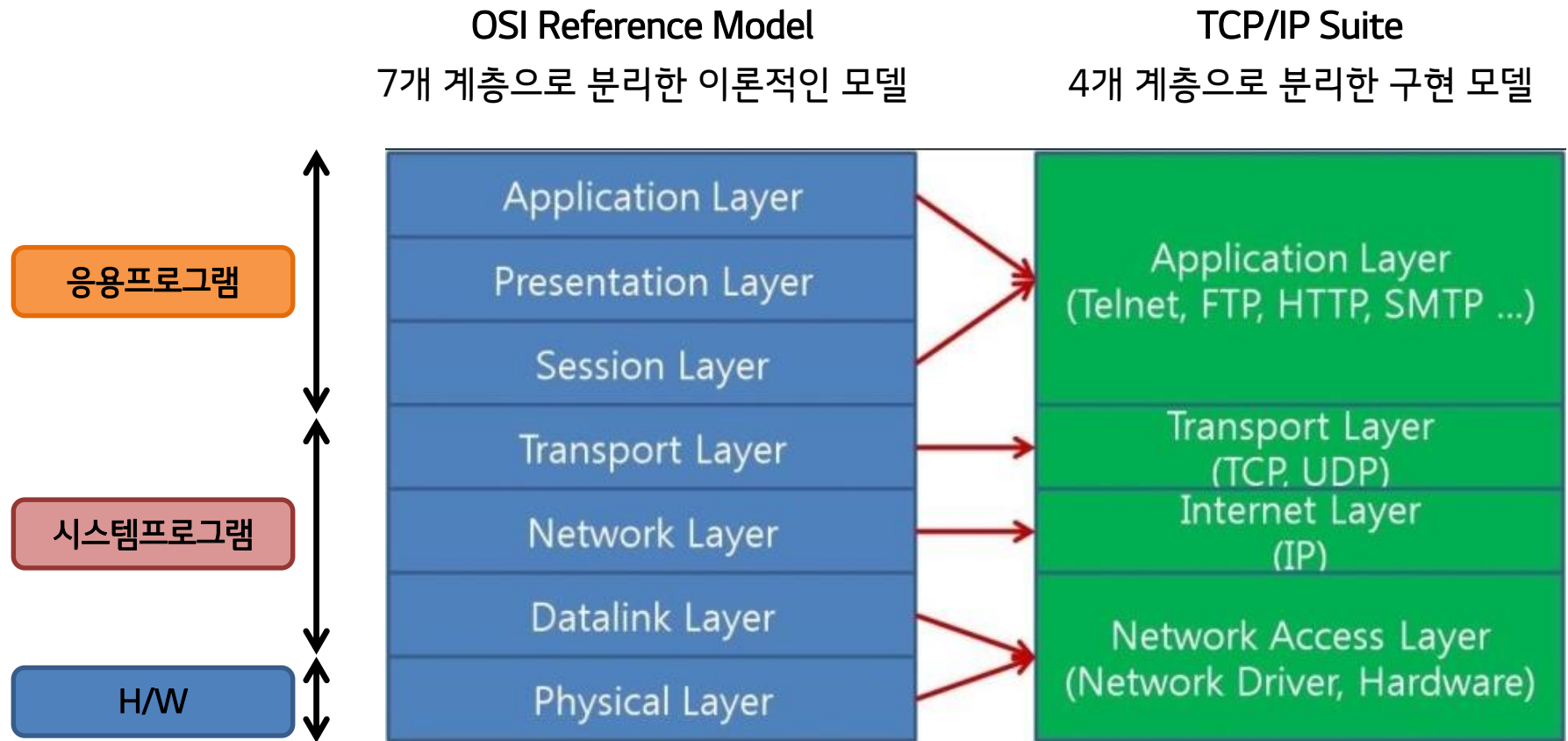
▪ TCP/IP Suite

미 국방성(DoD, Department of Defense)은 핵전쟁을 포함한 어떤 상황에서도 정상적으로 동작할 수 있는 네트워크를 필요로 했고, 이를 위해 TCP/IP 모델 개발을 위한 재정을 지원했음

▪ OSI Reference Model

ISO (International Organization for Standardization)는 네트워크 사이의 비호환성 및 통신 불능 문제를 해결하기 위해 DECnet, SNA, TCP/IP 등의 서로 다른 네트워크 기술들을 연구하여 서로 다른 네트워크 사이에 호환성을 제공하고 상호운용 가능한 네트워크를 생산할 수 있도록 도와주는 네트워크 모델을 개발함

➤ OSI Reference Model과 TCP/IP Suite 비교

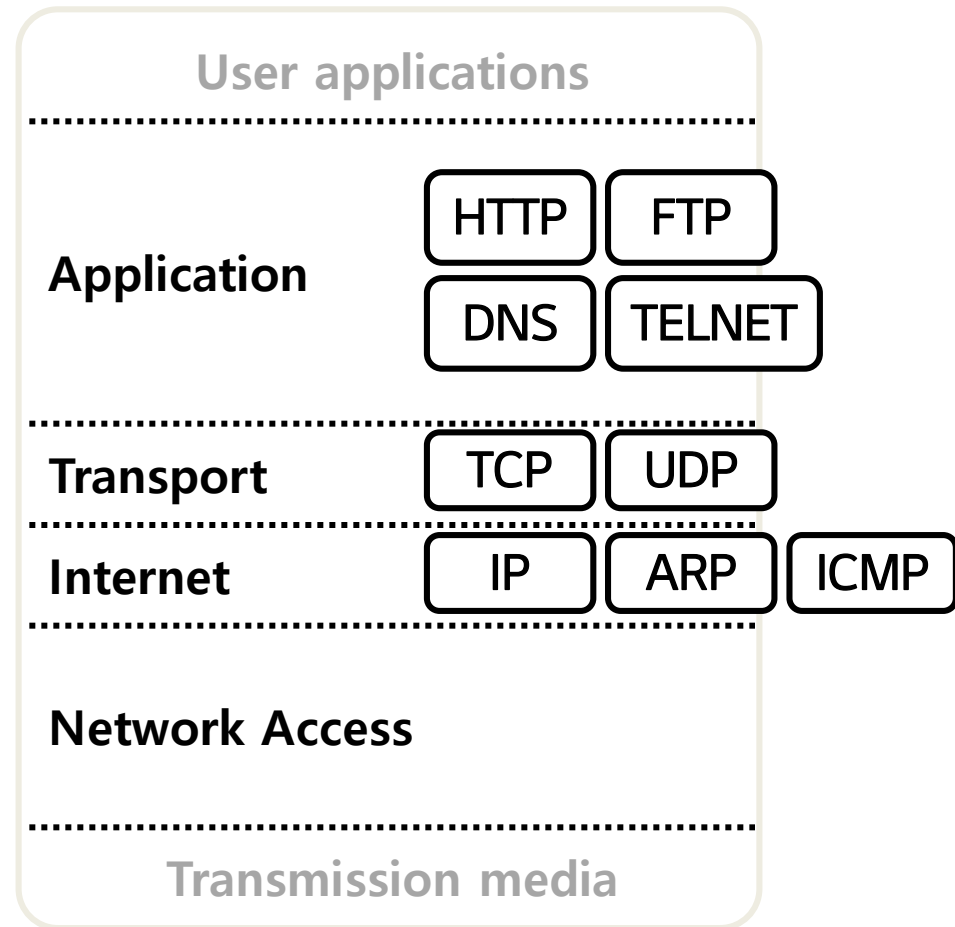


➤ Protocol 정의

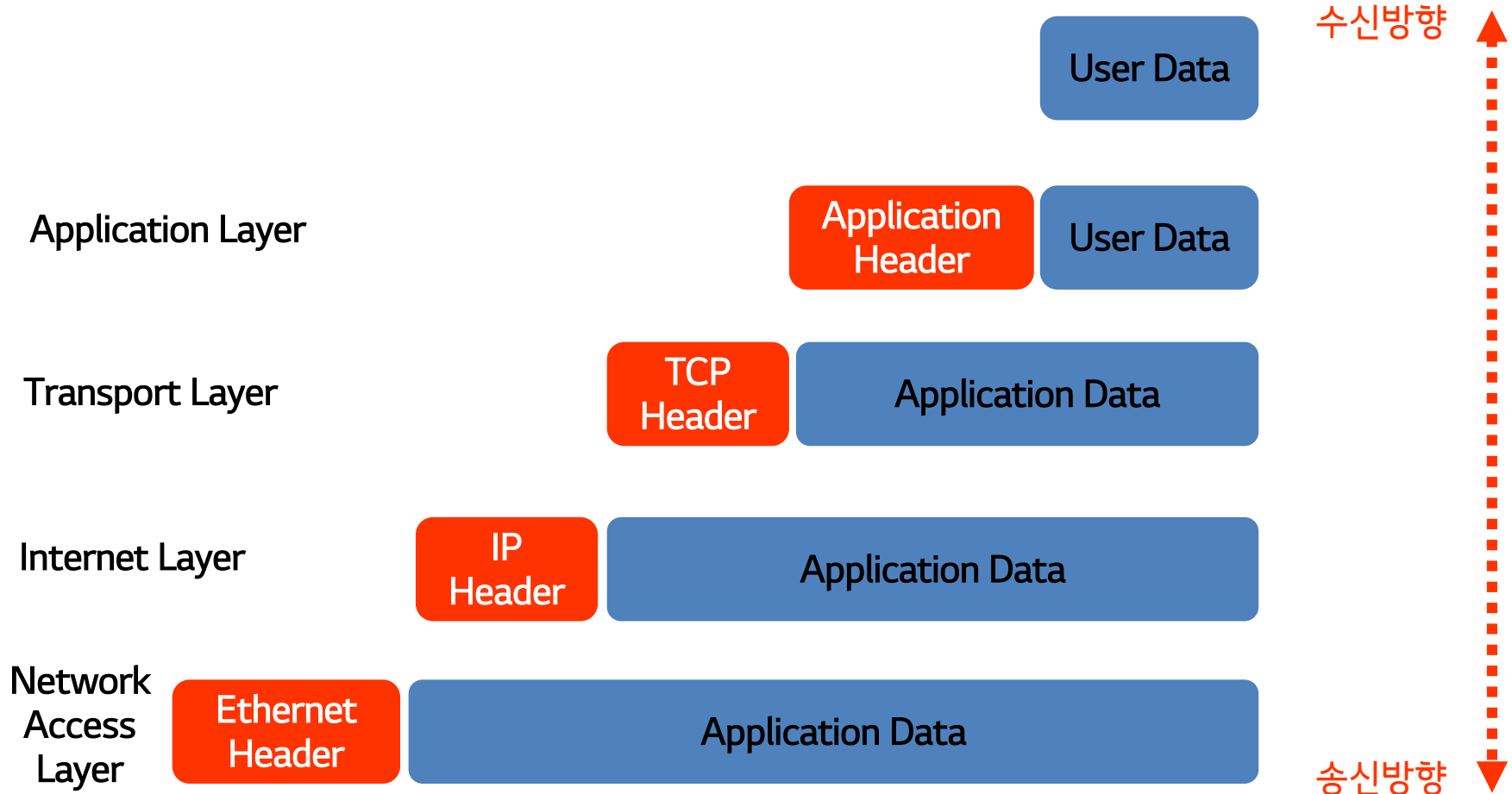
컴퓨터 사이의 통신은 data message를 교환함으로써 이루어지는데 컴퓨터가 이런 message를 받아서 처리하기 위해서는 message가 어떻게 정의되었고 무엇을 의미하는지 알 수 있어야 함

Data message 형식과 교환 방법을 기술하여 컴퓨터들이 어떻게 통신해야 하는지를 결정해주는 표준 규칙 집합

TCP/IP Suite에 포함된 주요 프로토콜

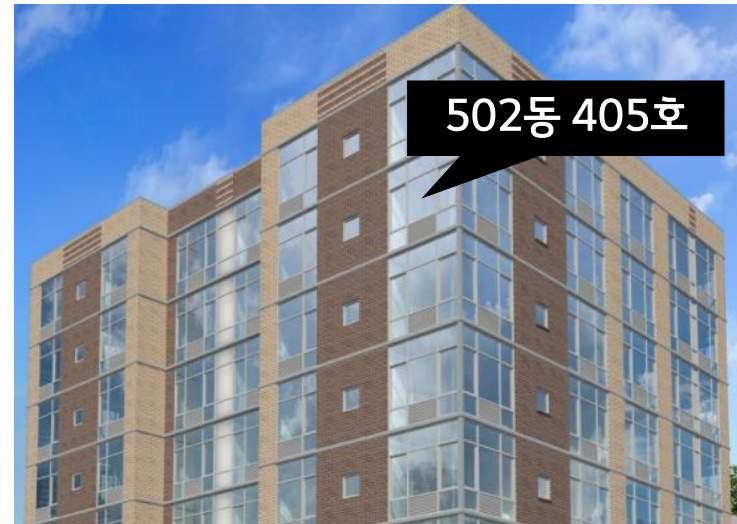


➤ Packet Encapsulation



➤ 통신하려는 양쪽 컴퓨터를 식별할 수 있는 방법

- 사용할 수 있는 주소가 한정되어 있으므로 중복되지 않도록 주소를 부여하기 위해 네트워크 주소(= network ID)와 해당 네트워크에 속한 호스트 주소(= host ID)의 두 부분으로 구성

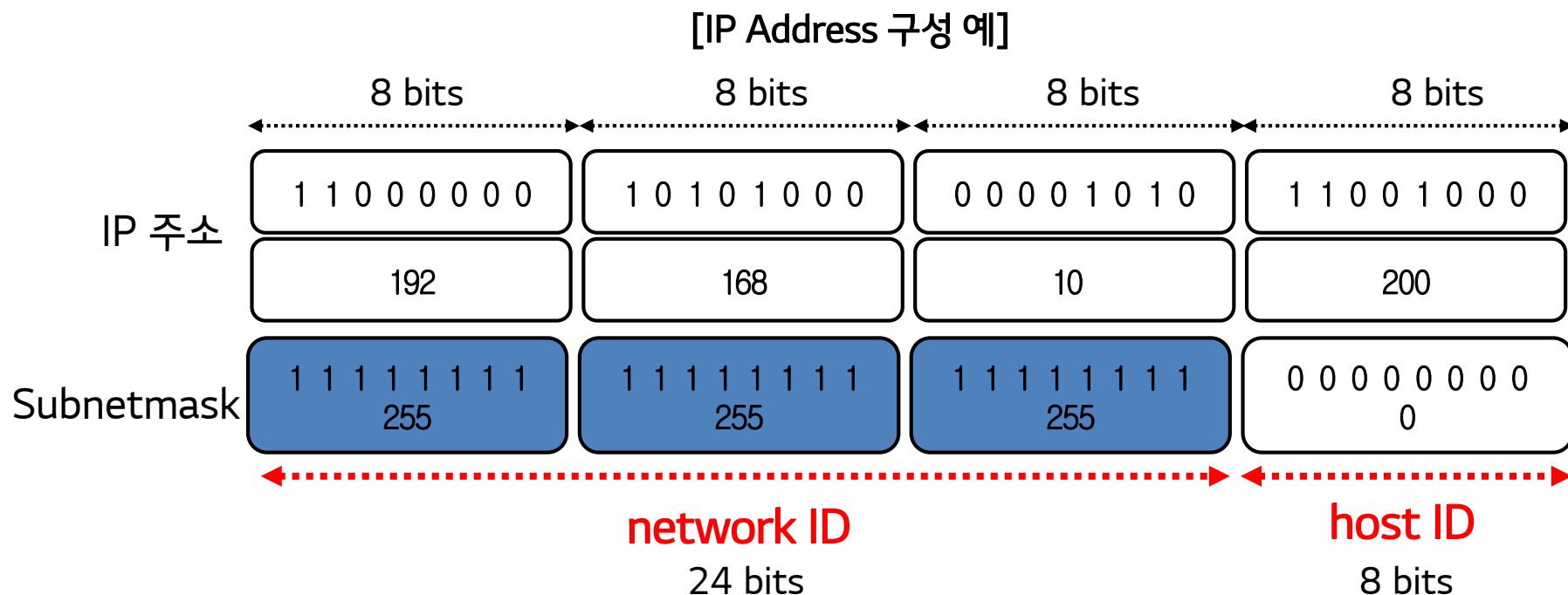


'동'으로 구분하지 않는다면 '405'호 주소는 중복될 것임 → '동'에 해당하는 개념이 '네트워크 주소'이며 '번지'에 해당하는 개념이 '호스트 주소'가 됨



➤ IP Address 이해 – Network ID와 Host ID 구분

- Subnetmask: IP Address 중 어느 bit까지가 network ID인지 식별하기 위해 사용하는 일종의 필터로, IP Address와 동일한 형식(32bit)으로 되어있으며 network ID에 해당하는 부분이 1로 표시되어 있음
- IP Address와 Subnetmask를 AND 연산하여 network ID와 host ID를 구분



➤ IP Address 이해 – Network 구분 예

IP 주소 (192.168.10.200)	11000000	10101000	00001010	11001000
	AND 연산			
subnet 마스크 (255.255.255.0)	11111111	11111111	11111111	00000000
	결과			
네트워크 주소	11000000 192	10101000 168	00001010 10	00000000 0

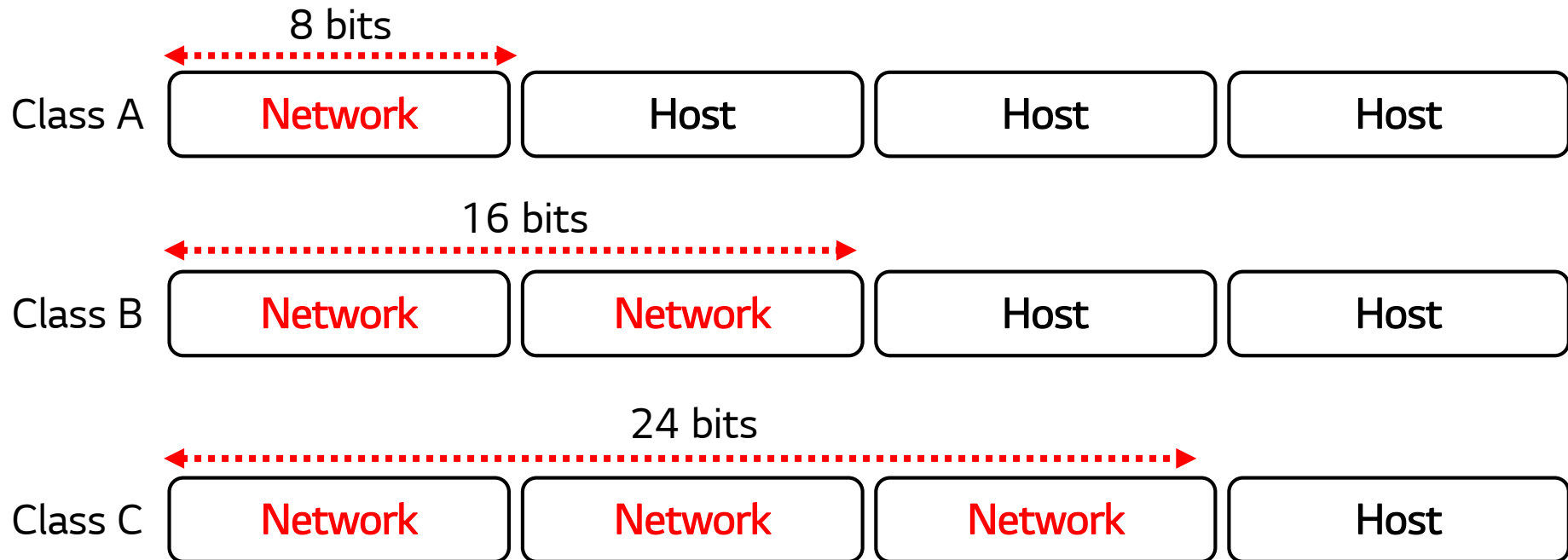
'192.168.10.0 네트워크'에 속해 있는 '200번 호스트'라는 의미

IP 주소 (192.168.20.150)	11000000	10101000	00010100	10010110
	AND 연산			
subnet 마스크 (255.255.255.0)	11111111	11111111	11111111	00000000
	결과			
네트워크 주소	11000000 192	10101000 168	00010100 20	00000000 0

'192.168.20.0 네트워크'에 속해 있는 '150번 호스트'라는 의미

➤ IP Address 이해 – Classful Addressing

- 네트워크 크기는 “네트워크에 포함된 호스트 수에 따라 서로 다르게 구현되어야 한다”는 가정에서 출발한 개념으로 네트워크 규모에 따라 구분



➤ IP Address 이해 – Classful Address Range

Class	Address Range (Decimal value of 1 st Octet)	Host Range
Class A	<ul style="list-style-type: none"> Class A 주소의 첫 번째 bit는 항상 0으로 시작 첫 번째 octet은 00000000 ~ 01111111 사이의 값이어야 함 10진수 0~127 범위 중에서 0과 127은 예약되어 사용할 수 없음 첫 번째 octet이 10진수 1~126 사이의 값으로 시작하는 주소 	<ul style="list-style-type: none"> 호스트를 표시하는 octet 이 모두 0인 경우는 네트워크 자체 주소임 호스트를 표시하는 octet 이 모두 1인 경우는 broadcast를 위한 주소임
Class B	<ul style="list-style-type: none"> Class B 주소의 처음 2bit는 항상 10으로 시작 첫 번째 octet은 10000000 ~ 10111111(10진수 128~191) 사이의 값으로 시작하는 주소 	
Class C	<ul style="list-style-type: none"> Class C 주소의 처음 3bit는 항상 110으로 시작 첫 번째 octet은 11000000~11011111(10진수 192~223) 사이의 값으로 시작하는 주소 	

➤ TCP 기능 정의 – RFC 793

- TCP는 송신지에서 수신지까지 정확하고 신뢰성 있게 정보 흐름을 제어하여 데이터를 전송하는 역할을 수행한다

[RFC 793 내용 일부]

The Transmission Control Protocol (TCP) is intended for use as a **highly reliable host-to-host protocol** between hosts in packet-switched computer communication networks, and in interconnected systems of such networks.

... 중략 ...

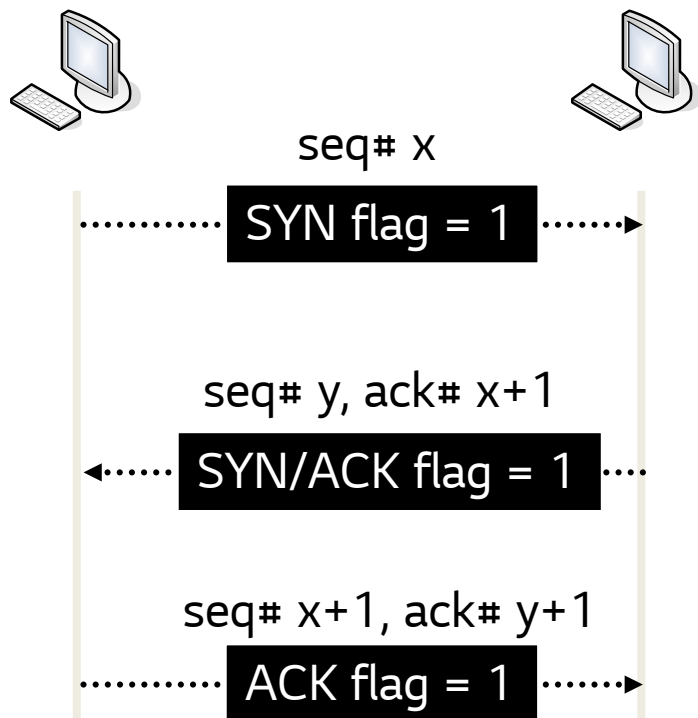
TCP is a **connection-oriented, end-to-end reliable protocol** designed to fit into a layered hierarchy of protocols which support multi-network applications. The TCP provides for **reliable inter-process communication between pairs of processes** in host computers attached to distinct but interconnected computer communication networks.

... 중략...

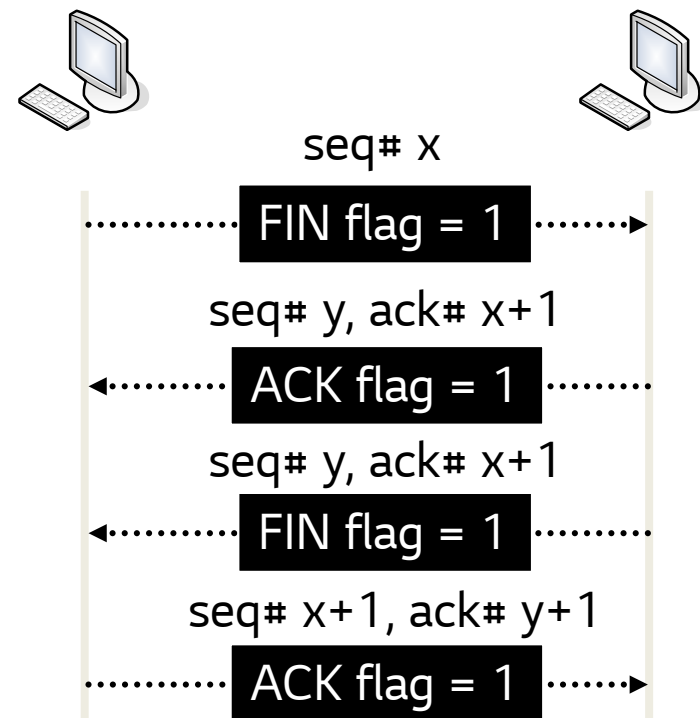
➤ TCP Connection Establishment & Termination

- 일련의 데이터 전송을 위해 데이터 전송을 관리할 논리적인 커넥션을 생성함
- 커넥션 생성/종료 의사를 Flag로 전달하며, 상호합의 하에 커넥션을 생성/종료함

[connection 개설 – 3way handshaking]

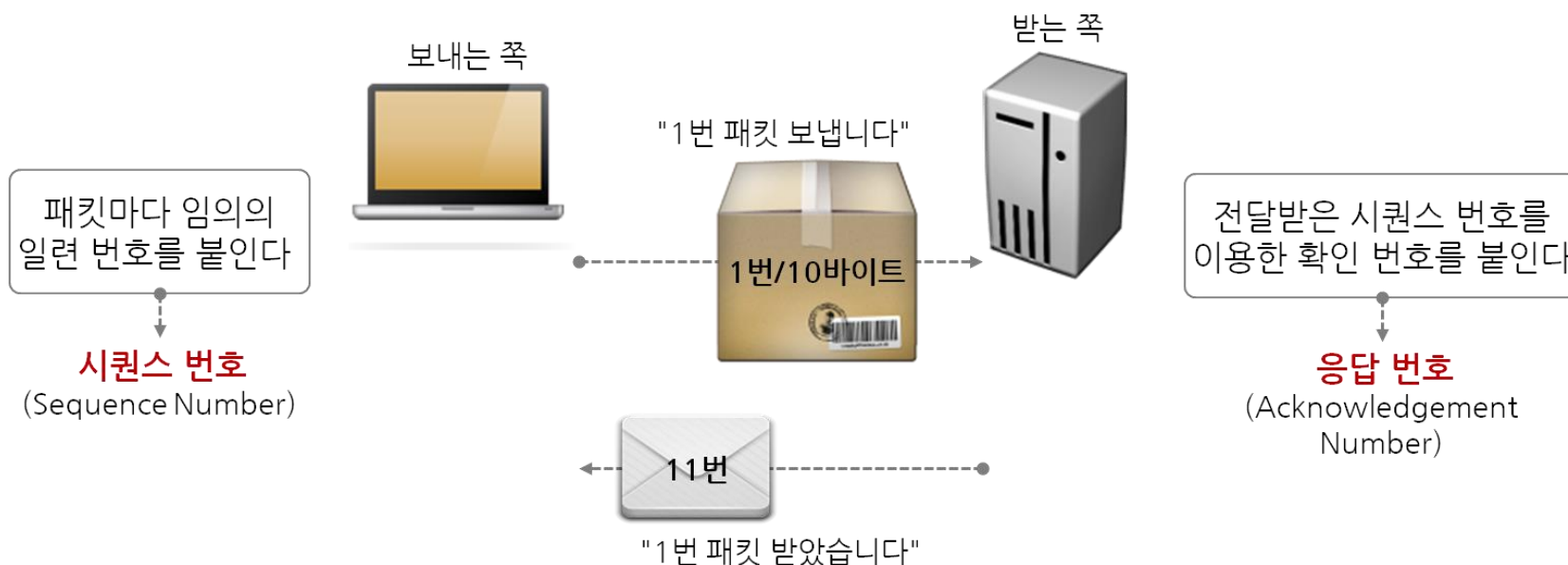


[connection 종료]



➤ TCP 신뢰성 (Sequence & Acknowledgement Number)

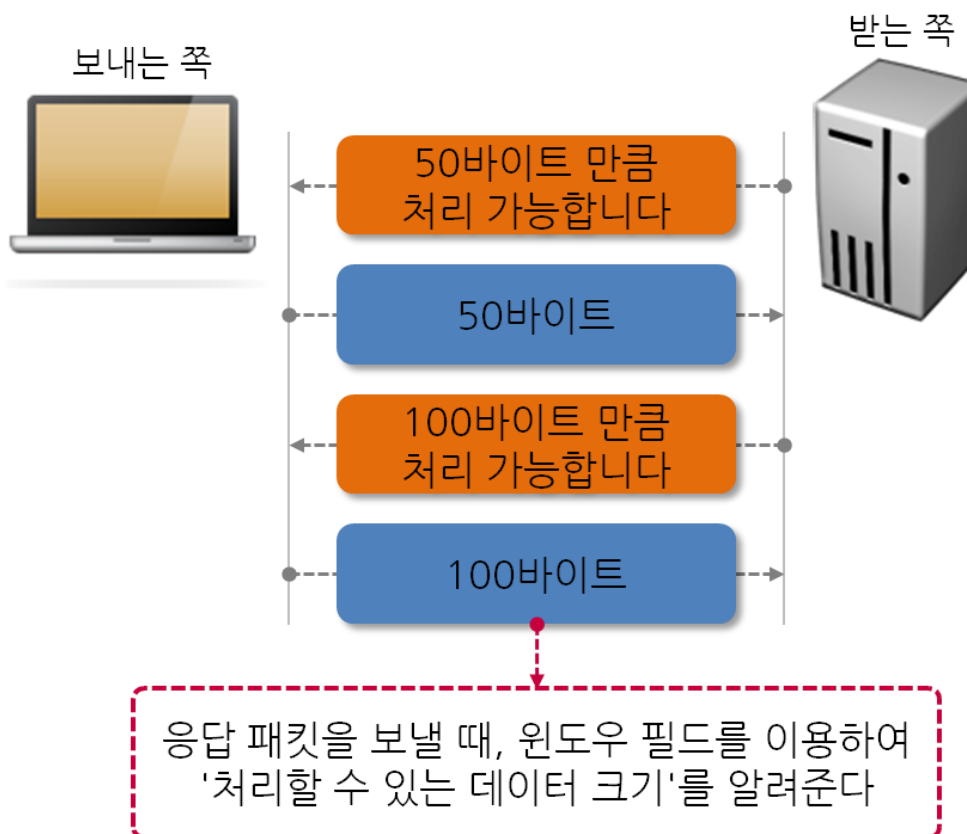
- 송신패킷에 포함된 전송 중인 Byte 정보를 이용하여 패킷의 수신 여부를 확인함
- 패킷이 정상적으로 전달되지 않았다고 판단될 경우, 해당 패킷을 재전송함



- 시퀀스 번호를 붙여서 보내면, 받은 쪽에서는 시퀀스 번호에 받은 데이터 크기만큼 더한 값을 확인 번호로 생성해서 응답해주는 구조로 동작
- 시퀀스 번호와 데이터 크기를 알아야만 적절한 응답 번호를 만들 수 있으므로, 보낸 패킷을 정상적으로 수신했음을 확인하는 의미

➤ TCP 흐름제어 (Sliding Window)

- 송신패킷에 대한 응답패킷에 자신이 처리할 수 있는 수신버퍼 여유 정보를 포함
- 송신 호스트는 해당 여유분만큼만 패킷을 송신하고 다음 응답패킷을 기다림



➤ UDP(User Datagram Protocol) 정의 : RFC 768

- TCP를 사용하기 위해서는 일정 부분만큼의 오버헤드를 감수해야 하지만, 패킷이 정상적으로 전달되었다는 보장을 모든 어플리케이션이 요구하는 것은 아니기 때문에 전달을 보장하기 위한 오버헤드를 감수하는 대신 좀 더 빠르게 세그먼트를 교환할 수 있는 protocol이 필요함

[RFC 768 내용 일부]

... 중략 ...

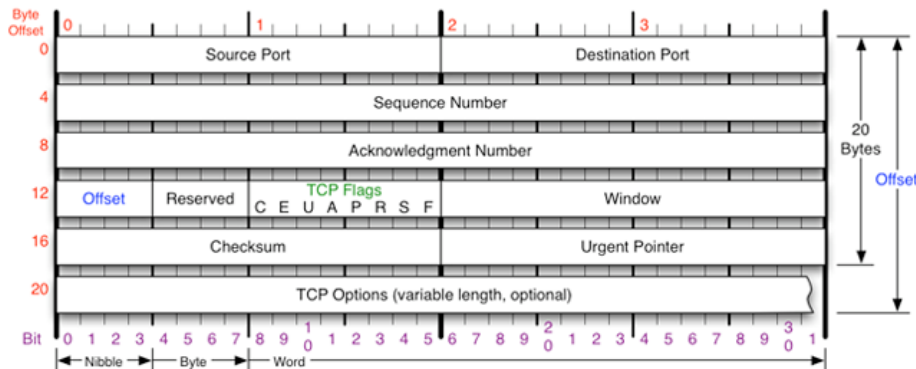
This protocol assumes that the Internet Protocol (IP) is used as the underlying protocol.

This protocol provides a procedure for application programs to send messages to other programs with a minimum of protocol mechanism. The protocol is transaction oriented, and delivery and duplicate protection are not guaranteed. Applications requiring ordered reliable delivery of streams of data should use the Transmission Control Protocol (TCP).

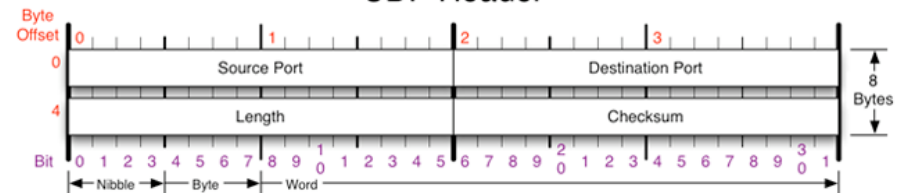
... 중략 ...

Properties	TCP	UDP
Reliability	Reliable	Unreliable
Connection Type	Connection-Oriented	Connectionless
Transmission	Byte-Oriented	Message-Oriented
Flow Control	Yes	No
Congestion Control	Yes	No
Delivery	Strictly Ordered	Unordered
Cast	Unicast	Unicast, Multicast, Broadcast
Used for	E-Mail, File sharing, Downloading	Voice/Video Streaming, Light Message

TCP Header

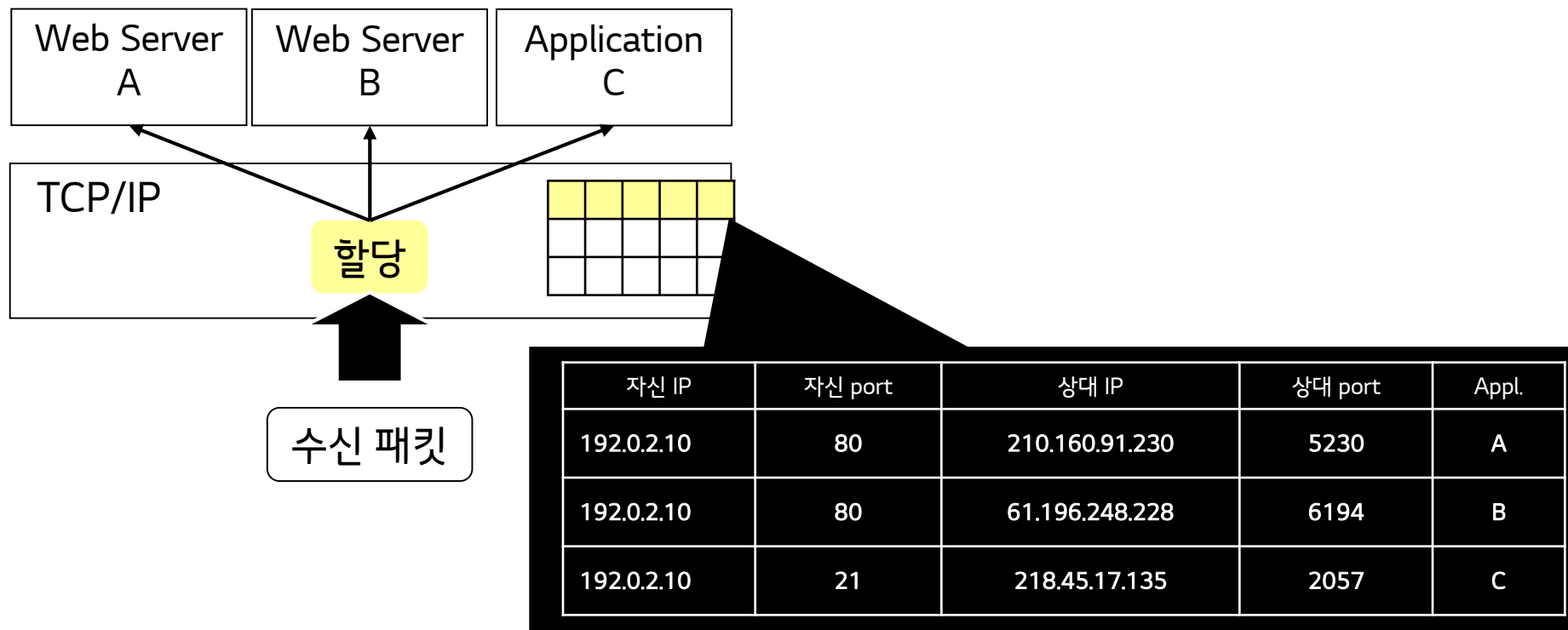


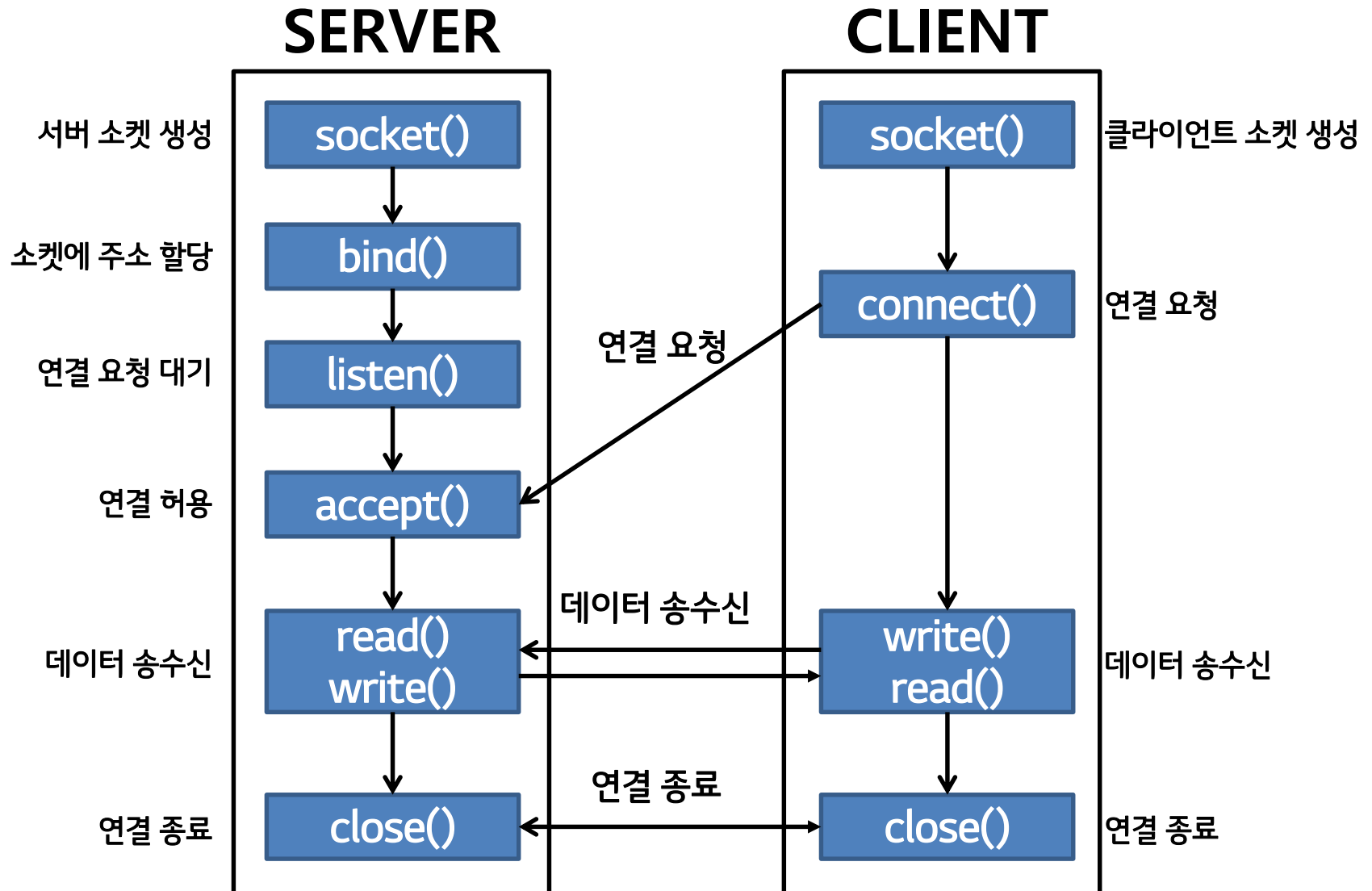
UDP Header



➤ TCP/IP 내부에서 관리하는 메모리 영역을 의미

- TCP를 사용하는지, UDP를 사용하는지, 현재 통신 상태는 어떤지 등에 대한 제어 정보들을 저장/관리 → TCP/IP 내부 프로토콜은 해당 내용을 참조하면서 동작
- 송신 측과 수신 측의 조합으로 식별 가능





▪ SERVER

```
public static void main(String[] args) throws IOException {
    ServerSocket listener = new ServerSocket(9090);
    try {
        Socket socket = listener.accept();
        try {
            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
            out.println( "test" );
        } finally {
            socket.close();
        }
    }
    finally {
        listener.close();
    }
}
```

▪ CLIENT

```
public static void main(String[] args) throws IOException {
    Socket s = new Socket("127.0.0.1", 9090);
    BufferedReader input = new BufferedReader(new InputStreamReader(s.getInputStream()));
    String answer = input.readLine();
    System.out.println(answer);
}
```

▪ SERVER

```
static void Main(string[] args)
{
    IPAddress ipAddress = IPAddress.Parse("127.0.0.1");
    IPEndPoint localEndPoint =
        new IPEndPoint(ipAddress, 9090);

    Socket listener = new Socket(
        AddressFamily.InterNetwork, SocketType.Stream,
        ProtocolType.Tcp);
    try
    {
        listener.Bind(localEndPoint);
        listener.Listen(10);

        Socket handler = listener.Accept();
        byte[] msg = Encoding.ASCII.GetBytes("test");
        handler.Send(msg);
        handler.Shutdown(SocketShutdown.Both);
        handler.Close();
    }
    catch (Exception e)
    {
        Console.WriteLine(e.ToString());
    }
}
```

▪ CLIENT

```
static void Main(string[] args)
{
    byte[] bytes = new byte[1024];
    try {
        IPAddress ipAddress = IPAddress.Parse("127.0.0.1");
        IPEndPoint remoteEP = new
            IPEndPoint(ipAddress, 9090);

        Socket sender = new Socket(
            AddressFamily.InterNetwork,
            SocketType.Stream, ProtocolType.Tcp);

        try
        {
            sender.Connect(remoteEP);
            int bytesRec = sender.Receive(bytes);
            Console.WriteLine(
                Encoding.ASCII.GetString(bytes, 0, bytesRec));
            sender.Shutdown(SocketShutdown.Both);
            sender.Close();
        }
        catch (Exception e)
        {
            Console.WriteLine(e.ToString());
        }
    }
}
```


▪ SERVER

```
#define PORT 9000
int main ( ) {
    char buffer[100];
    int c_socket, s_socket;
    struct sockaddr_in s_addr, c_addr;
    int len, n;

    s_socket = socket(PF_INET, SOCK_STREAM, 0);
    memset(&s_addr, 0, sizeof(s_addr));
    s_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    s_addr.sin_family = AF_INET;
    s_addr.sin_port = htons(PORT);

    bind(s_socket,
        (struct sockaddr *) &s_addr, sizeof(s_addr));
    listen(s_socket, 5);
    len = sizeof(c_addr);
    c_socket =
        accept(s_socket, (struct sockaddr *)&c_addr, &len);

    sprintf(buffer, "test"); n = strlen(buffer);
    write(c_socket, buffer, n);
    close(c_socket);
    close(s_socket);
}
```

▪ CLIENT

```
#define PORT 9000
#define BUF_SIZE 256
#define IPADDR "127.0.0.1"
int main ( ) {
    int c_socket;
    struct sockaddr_in c_addr;
    int len, n;
    char rcvBuffer[BUF_SIZE];

    c_socket = socket(PF_INET, SOCK_STREAM, 0);
    memset(&c_addr, 0, sizeof(c_addr));
    c_addr.sin_addr.s_addr = inet_addr(IPADDR);
    c_addr.sin_family = AF_INET;
    c_addr.sin_port = htons(PORT);

    connect(c_socket, (struct sockaddr *) &c_addr,
        sizeof(c_addr));

    n = read(c_socket, rcvBuffer, sizeof(rcvBuffer));

    rcvBuffer[n] = '\0';
    printf("received data : %s\n", rcvBuffer);
    close(c_socket);
}
```

▪ SERVER

```
#pragma comment(lib, "Ws2_32.lib")
int main(void) {
    WSADATA wsaData; int iResult;
    SOCKET ListenSocket, ClientSocket;
    struct addrinfo *result = NULL; struct addrinfo hints;
    int iSendResult; char sendbuf[512];

    iResult = WSASStartup(MAKEWORD(2, 2), &wsaData);
    ZeroMemory(&hints, sizeof(hints));
    hints.ai_family = AF_INET;
    hints.ai_socktype = SOCK_STREAM;
    hints.ai_protocol = IPPROTO_TCP;
    hints.ai_flags = AI_PASSIVE;
    iResult = getaddrinfo(NULL, 9090, &hints, &result);
    ListenSocket = socket(result->ai_family,
                          result->ai_socktype, result->ai_protocol);
    iResult = bind(ListenSocket, result->ai_addr,
                  (int)result->ai_addrlen);

    freeaddrinfo(result);
    iResult = listen(ListenSocket, SOMAXCONN);
    ClientSocket = accept(ListenSocket, NULL, NULL);
    sprintf_s(sendbuf, 4, "test");
    iSendResult = send(ClientSocket, sendbuf, 4, 0);
    closesocket(ClientSocket); closesocket(ListenSocket);
    iResult = shutdown(ClientSocket, SD_SEND);
    WSACleanup();
    return 0;
}
```

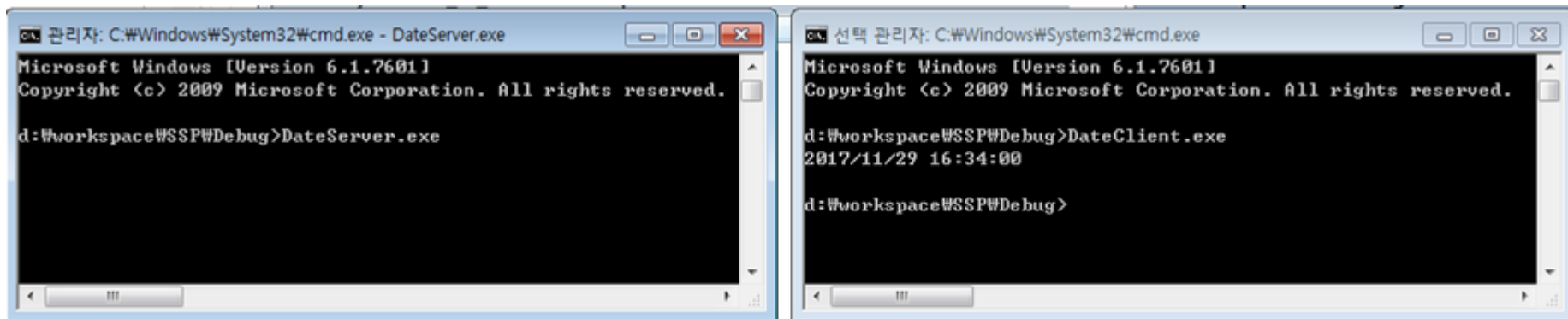
▪ CLIENT

```
#pragma comment(lib, "Ws2_32.lib")
void main()
{
    int c_socket;
    struct sockaddr_in c_addr;
    char rcvBuffer[512];
    int len;
    WSADATA wsaData;
    IN_ADDR inaddr;

    WSASStartup(MAKEWORD(2, 2), &wsaData);
    c_socket = socket(AF_INET, SOCK_STREAM, 0);
    memset(&c_addr, 0, sizeof(c_addr));
    inet_pton(AF_INET, "127.0.0.1", &c_addr.sin_addr);
    c_addr.sin_family = AF_INET;
    c_addr.sin_port = htons(9090);

    connect(c_socket,
            (struct sockaddr *) &c_addr, sizeof(c_addr));
    len = recv(c_socket, rcvBuffer, sizeof(rcvBuffer), 0);
    rcvBuffer[len] = '\0';
    printf("%s\n", rcvBuffer);
    closesocket(c_socket);
}
```

1. Client에서 Server에 접속하면 Server는 현재 날짜와 시각을 Client로 전송하고, Client는 전송 받은 값을 출력하시오.



The image shows two side-by-side Windows command prompt windows. The left window is titled '관리자: C:\Windows\System32\cmd.exe - DateServer.exe' and shows the command 'DateServer.exe' being executed at the prompt 'd:\workspace\SSP\Debug>'. The right window is titled '선택 관리자: C:\Windows\System32\cmd.exe' and shows the command 'DateClient.exe' being executed at the prompt 'd:\workspace\SSP\Debug>'. The output of DateClient.exe shows the date and time '2017/11/29 16:34:00'.

```
관리자: C:\Windows\System32\cmd.exe - DateServer.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

d:\workspace\SSP\Debug>DateServer.exe

선택 관리자: C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

d:\workspace\SSP\Debug>DateClient.exe
2017/11/29 16:34:00

d:\workspace\SSP\Debug>
```

2. Client에서 Server에 접속하여 파일을 전송하는 프로그램을 작성하시오.

- ClientFiles 폴더의 모든 파일을 전송하여 ServerFiles폴더에 저장
 - Client는 파일 전송 완료 후 종료
 - Server는 파일을 수신 완료하고 다시 Client 접속 대기
 - Server는 'QUIT'입력을 받으면 종료

▪ SERVER

```
Socket s = listener.accept();
DataInputStream is = new DataInputStream(s.getInputStream());
try {
    String fileName = null;
    // 파일이름 수신
    while ((fileName = is.readUTF()) != null) {
        // 파일크기 수신
        int fileSize = is.readInt();
        FileOutputStream fw
            = new FileOutputStream("./ServerFiles/" + fileName);
        int length;
        while (fileSize > 0) {
            // 파일내용 수신
            length = is.read(buffer, 0, Math.min(fileSize, buffer.length));
            fileSize -= length;
            fw.write(buffer, 0, length);
        }
        fw.close();
        System.out.println(fileName+" is received.");
    }
}
```

▪ CLIENT

```
Socket s = new Socket("127.0.0.1", 27015);
DataOutputStream os = new DataOutputStream(s.getOutputStream());

byte[] buffer = new byte[4096];
int length;

// get all the files from a directory
File directory = new File("./ClientFiles");
File[] fList = directory.listFiles();
for (File file : fList) {
    if (file.isFile()) {
        // 파일이름 전송
        os.writeUTF(file.getName());
        // 파일크기 전송
        os.writeInt((int) file.length());

        FileInputStream is = null;
        try {
            is = new FileInputStream(file.getPath());
            while ((length = is.read(buffer)) != -1) {
                // 파일내용 전송
                os.write(buffer, 0, length);
            }
        } finally {
            if (is != null) { is.close(); }
        }
    }
}
}
```

▪ SERVER

```
Socket handler = listener.Accept();

NetworkStream ns = new NetworkStream(handler);
BinaryReader br = new BinaryReader(ns);
FileStream fs = null;

try
{
    string filename;
    // 파일이름 수신
    while ((filename = br.ReadString()) != null)
    {
        // 파일크기 수신
        int length = (int)br.ReadInt64();

        fs = new FileStream (ReceiveFolder + "/" +
                               filename, FileMode.Create);

        while (length > 0)
        {
            // 파일내용 수신
            int nReadLen = br.Read(bytes, 0,
                                   Math.Min(BUF_SIZE, length));
            fs.Write(bytes, 0, nReadLen);
            length -= nReadLen;
        }
        fs.Close();
        Console.WriteLine(filename + " is received.");
    }

    handler.Shutdown(SocketShutdown.Both);
    handler.Close();
}
```

▪ CLIENT

```
sender.Connect(remoteEP);

NetworkStream ns = new NetworkStream(sender);
BinaryWriter bw = new BinaryWriter(ns);

DirectoryInfo di = new DirectoryInfo("./ClientFiles");
FileInfo[] fiArr = di.GetFiles();
foreach (FileInfo infoFile in fiArr)
{
    // 파일이름 전송
    bw.Write(infoFile.Name);
    long lSize = infoFile.Length;

    // 파일크기 전송
    bw.Write(lSize);

    // 파일내용 전송
    FileStream fs = new FileStream(infoFile.FullName, FileMode.Open);
    while (lSize > 0)
    {
        int nReadLen = fs.Read(bytes, 0,
                               Math.Min(BUF_SIZE, (int)lSize));
        bw.Write(bytes, 0, nReadLen);
        lSize -= nReadLen;
    }
    fs.Close();
}
```

Ch 7. JSON

7.1 Overview

7.2 JSON 예시

7.3 JSON Library

7.4 실습

JSON이란?

JSON(제이슨[1], JavaScript Object Notation)은 속성-값 쌍(attribute-value pairs and array data types (or any other serializable value)) 또는 "키-값 쌍"으로 이루어진 데이터 오브젝트를 전달하기 위해 인간이 읽을 수 있는 텍스트를 사용하는 개방형 표준 포맷이다.

비동기 브라우저/서버 통신 (AJAX)을 위해, 넓게는 XML(AJAX가 사용)을 대체하는 주요 데이터 포맷이다.

특히, 인터넷에서 자료를 주고 받을 때 그 자료를 표현하는 방법으로 알려져 있다.

자료의 종류에 큰 제한은 없으며, 특히 컴퓨터 프로그램의 변수값을 표현하는 데 적합하다.

<https://ko.wikipedia.org/wiki/JSON>

JSON의 기본 자료형

- 수(Number)
- 문자열(String): 0개 이상의 유니코드 문자들의 연속. 문자열은 큰 따옴표(")로 구분하며 역슬래시 이스케이프 문법을 지원한다.
- 참/거짓(Boolean): true 또는 false 값
- 배열(Array): 0 이상의 임의의 종류의 값으로 이루어진 순서가 있는 리스트. 대괄호로 나타내며 요소는 쉼표로 구분한다.
- 객체(Object): 순서가 없는 이름/값 쌍의 집합으로, 이름(키)이 문자열이다.
- null: 빈 값으로, null을 사용한다.

```
{  
  "name": "spiderman",  
  "age": 45,  
  "married": true,  
  "specialty": ["martial art", "gun"],  
  "vaccine": {"1st": "done", "2nd": "expected", "3rd": null},  
  "children": [{"name": "spiderboy", "age": 10}, {"name": "spidergirl", "age": 8}],  
  "adress": null  
}
```

The diagram illustrates the data types of the values in the JSON object. Callouts point to specific values:

- String**: points to the value of "name" ("spiderman").
- Number**: points to the value of "age" (45).
- Boolean**: points to the value of "married" (true).
- Array**: points to the value of "specialty" (["martial art", "gun"]).
- Object**: points to the value of "vaccine" ({"1st": "done", "2nd": "expected", "3rd": null}).
- null**: points to the value of "adress" (null).

[Java]

- Google Gson 2.8.6
(<https://github.com/google/gson>)

[C#]

- Newtonsoft.Json
(<https://www.newtonsoft.com/json>)

[C]

- json-c
(<https://github.com/json-c/json-c>)

Json Serialization/Deserialization

- Json 변환 (ex. String <-> JsonObject)

```
import com.google.gson.JsonElement;  
import com.google.gson.JsonParser;  
  
public class MyJson {  
  
    public static void main(String[] args) {  
        JsonElement jsonElement = JsonParser.parseString("{ W\"keyW\":W\"valueW\" }");  
        System.out.println(jsonElement.toString());  
    }  
}
```

Json Serialization/Deserialization

- Json 변환 (ex. String <-> JsonObject)

```
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;

JObject json = new JObject();
json["name"] = "John Doe";
json["salary"] = 300100;
string jsonstr = json.ToString();
Console.WriteLine("Json : " + jsonstr);
JObject json2 = JObject.Parse(jsonstr);
Console.WriteLine($"Name : {json2["name"]}, Salary : {json2["salary"]}");
```

Json Serialization/Deserialization

- Json 변환 (ex. String <-> JsonObject)

```
#include <json-c/json.h>
```

```
json_object *myobj = json_object_new_object();  
json_object_object_add(myobj, "name", json_object_new_string("KIM"));  
json_object_object_add(myobj, "phone", json_object_new_string("010000000000"));  
const char *result_json = json_object_to_json_string_ext(myobj, JSON_C_TO_STRING_PLAIN);  
printf("json: %s\n", result_json);
```

```
json_object *root, *name, *phone;  
root = json_tokener_parse(result_json);  
json_object_object_get_ex(root, "name", &name);  
json_object_object_get_ex(root, "phone", &phone);  
printf("name: %s, phone: %s\n", json_object_get_string(name),  
      json_object_get_string(phone));
```

1. JSON Library를 활용하여 다음 데이터를 sample.json 파일로 저장하시오.

```
{
  "name":"spiderman",
  "age":45,
  "married":true,
  "specialty":["martial art", "gun"],
  "vaccine":{"1st":"done","2nd":"expected","3rd":null},
  "children": [{"name":"spiderboy", "age":10}, {"name":"spidergirl", "age":8}],
  "adress":null
}
```

* 파일에 출력 시 개행 없어도 상관없음

```
{"name":"spiderman","age":45,"married":true,"specialty":["martial art","gun"],"vaccine":{"1st":"done","2nd":"expected"},"children":[{"name":"spiderboy","age":10},{"name":"spidergirl","age":8}]}
```

2. sample.json 파일을 읽어서 다음과 같이 출력하시오.

name(age) : spiderman(45)

name(age) : spidergirl(8)

3. sample.json 파일을 읽은 후, 첫 번째 level의 Value Type을 알아내어 다음과 같이 출력하시오.

Key : name / Value Type : String

Key : age / Value Type : Number

Key : married / Value Type : Boolean

Key : specialty / Value Type : Array

Key : vaccine / Value Type : Object

Key : children / Value Type : Array

Key : address / Value Type : null

<JAVA>

Key : name / Value Type : String

Key : age / Value Type : Integer

Key : married / Value Type : Boolean

Key : specialty / Value Type : Array

Key : vaccine / Value Type : Object

Key : children / Value Type : Array

Key : address / Value Type : Null

<C#>

Key : name / Value Type : string

Key : age / Value Type : int

Key : married / Value Type : boolean

Key : specialty / Value Type : array

Key : vaccine / Value Type : object

Key : children / Value Type : array

Key : address / Value Type : null

<C>

Ch 8. Http 통신

7.1 Overview

7.2 HTTP History

7.3 HTTP 프로토콜

7.4 실습

HTTP 통신이란?

Client의 요청(Request)이 있을 때 서버가 응답(Response)하여 해당 정보를 전송하고 연결을 종료하는 방식

HTTP History

HTTP(Hypertext Transfer Protocol) : 웹상에서 클라이언트와 서버 간 통신을 위한 프로토콜

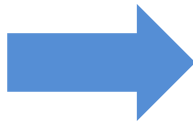
HTTP /0.9

GET /mypage.html

<HTML>

HTML Page

</HTML>



HTTP /1.0

GET /mypage.html HTTP/1.0

User-Agent: NCSA_Mosaic/2.0 (Windows 3.1)

200 OK

Date: Sun, 13 Nov 1994 10:22:53 GMT

Server: CERN/3.0 libwww/2.17

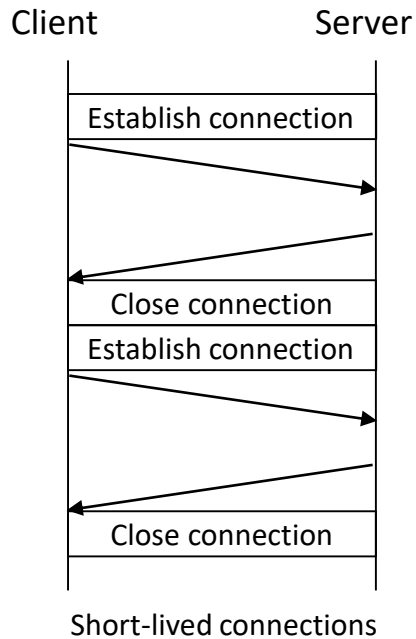
Content-Type: text/html

<HTML>

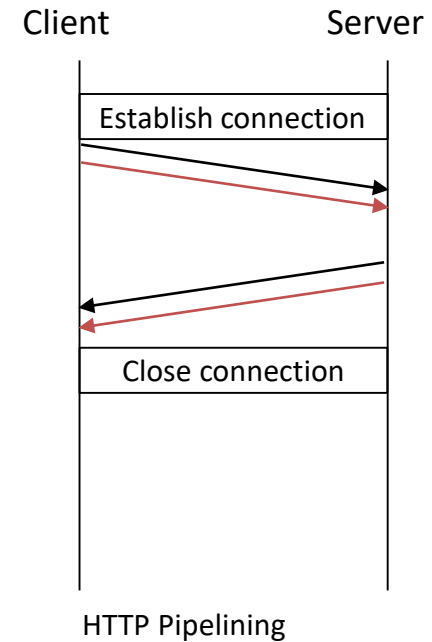
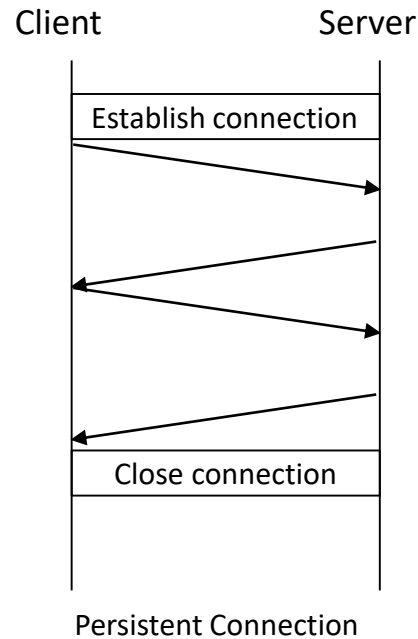
HTML Page with an image

</HTML>

HTTP /1.0



HTTP /1.1



HTTP /2 (2015년)

기존 HTTP /1.x 버전의 성능 향상에 초점을 맞춘 프로토콜 (2015년)

표준의 대체가 아닌 확장

특징

- ✓ HTTP 메시지 전송 방식 변화
 - 바이너리 프레이밍 계층 사용
- ✓ Request and Response Multiplexing
 - Head Of Line Blocking 해결
- ✓ Stream Prioritization
 - 리소스간 우선 순위 설정 가능
- ✓ Server Push
 - 클라이언트가 요청하지 않은 데이터를 서버가 전송
- ✓ Header Compression
 - 헤더의 크기를 줄여 페이지 로드 시간 감소

HTTP /3 (2020년)

- ✓ QUIC (UDP 기반)
- ✓ TCP의 Head of Line Blocking 문제 극복

URL (Uniform Resource Locators)

- 서버에 자원을 요청하기 위한 영문 주소

http://www.mydomain.com:8888/my/resource/path?qa=x&qy=y

protocol host port Resource path query

Http 요청 메서드

- GET : 존재하는 자원에 대한 요청
- POST : 새로운 자원을 생성
- PUT : 존재하는 자원에 대한 변경
- DELETE : 존재하는 자원에 대한 삭제
- HEAD : 문서의 헤더 정보만 요청하며, 응답데이터(body)를 받지 않음
- TRACE : 클라이언트가 요청한 자원에 도달하기까지의 경로를 기록하는 루프백(loop back) 검사용,
클라이언트가 요청 자원에 도달하기까지 거쳐가는 프록시나 게이트웨이의 중간 경로부터 최종 수신 서버까지의
경로를 알아낼 때 사용

Http Header 주요 항목

1. 공통헤더

- Date : HTTP 메시지가 만들어진 시각
- Connection : 일반적으로 HTTP/1.1을 사용하며 Connection은 기본적으로 keep-alive로 되어 있음
- Content-Length : 요청과 응답 메시지의 본문 크기를 바이트 단위로 표시
- Cache-Control : 요청과 응답 내의 캐싱 메커니즘을 위한 디렉티브를 정하기 위해 사용
- **Content-Type** : 콘텐츠의 타입(MIME)과 문자열 인코딩(utf-8 등등)을 명시
- Content-Language : 사용자의 언어
- Content-Encoding : 응답 콘텐츠를 압축해서 보내면, 브라우저가 해제해서 사용
- Content-Location, Pragma, Trailer, ...

2. 요청헤더

- Host : 서버의 도메인 네임
- User-Agent : 사용자가 어떤 클라이언트(운영체제, 앱, 브라우저 등)를 통해 요청을 보냈는지 알 수 있음
- Accept : 클라이언트가 허용할 수 있는 파일 형식(MIME TYPE)
- Cookie, Origin, If-Modified-Since, Authorization, ...

3. 응답헤더

- Server : 웹서버 정보
- Access-Control-Allow-Origin, Allow, Content-Disposition, Location, ...

Content-Type

1. Multipart Related MIME 타입

- Content-Type: Multipart/related
- Content-Type: Application/X-FixedRecord

2. XML Media의 타입

- Content-Type: text/xml
- Content-Type: Application/xml
- Content-Type: Application/xml-external-parsed-entity
- Content-Type: Application/xml-dtd
- Content-Type: Application/mathtml+xml
- Content-Type: Application/xslt+xml

3. Application의 타입

- Content-Type: Application/EDI-X12
- Content-Type: Application/EDIFACT
- Content-Type: Application/javascript
- Content-Type: Application/octet-stream
- Content-Type: Application/ogg
- Content-Type: Application/x-shockwave-flash
- **Content-Type: Application/json**
- Content-Type: Application/x-www-form-urlencoded

Content-Type

4) 오디오 타입

- Content-Type: audio/mpeg
- Content-Type: audio/x-ms-wma
- Content-Type: audio/vnd.rn-realaudio

5) Multipart 타입

- Content-Type: multipart/mixed: MIME E-mail;
- Content-Type: multipart/alternative: MIME E-mail;
- Content-Type: multipart/related: MIME E-mail
- **Content-Type: multipart/formed-data**

6) TEXT 타입

- Content-Type: text/css
- Content-Type: text/html
- Content-Type: text/javascript
- Content-Type: text/plain
- Content-Type: text/xml

7) file 타입

- Content-Type: application/msword
- Content-Type: application/pdf
- Content-Type: application/vnd.ms-excel
- Content-Type: application/x-javascript
- Content-Type: application/zip
- Content-Type: image/jpeg
- Content-Type: text/css
- Content-Type: text/html
- Content-Type: text/plain
- Content-Type: text/xml
- Content-Type: text/xsl

Http 상태 코드 (Status Code) - 서버에서 보내는 응답(Response) 정보

▪ 주요 상태 코드

1. 2xx : 대부분 성공을 의미
 - 200 : GET 요청에 대한 성공
 - 204 : No Content. 성공했으나 응답 본문에 데이터가 없음
 - 205 : Reset Content. 성공했으나 클라이언트의 화면을 새로 고침하도록 권고
 - 206 : Partial Content. 성공했으나 일부 범위의 데이터만 반환
2. 3xx : 대부분 클라이언트가 이전 주소로 데이터를 요청하여 서버에서 새 URL로 리다이렉트를 유도하는 경우
 - 301 : Moved Permanently, 요청한 자원이 새 URL에 존재
 - 303 : See Other, 요청한 자원이 임시 주소에 존재
 - 304 : Not Modified, 요청한 자원이 변경되지 않았으므로 클라이언트에서 캐싱된 자원을 사용하도록 권고. ETag와 같은 정보를 활용하여 변경 여부를 확인
3. 4xx : 대부분 클라이언트의 코드가 잘못된 경우. 유효하지 않은 자원을 요청했거나 요청이나 권한이 잘못된 경우 발생
 - 400 : Bad Request, 잘못된 요청
 - 401 : Unauthorized, 권한 없이 요청. Authorization 헤더가 잘못된 경우
 - 403 : Forbidden, 서버에서 해당 자원에 대해 접근 금지
 - 405 : Method Not Allowed, 허용되지 않은 요청 메서드
 - 409 : Conflict, 최신 자원이 아닌데 업데이트하는 경우. ex) 파일 업로드 시 버전 충돌
4. 5xx : 서버 에러
 - 501 : Not Implemented, 요청한 동작에 대해 서버가 수행할 수 없는 경우
 - 503 : Service Unavailable, 서버가 과부하 또는 유지 보수로 내려간 경우

1. Client에서 Server에 "http://127.0.0.1:8088/requestDate"로 요청하고, Server는 현재 날짜와 시각을 Client로 응답하게 하시오. (요청 Method는 'GET' 사용)

C:\Windows\System32\cmd.exe

```
C:\Users\GENE\source\repos\HTTP_SERVER\HTTP_SERVER\bin\Debug>HTTP_SERVER.exe  
Request : http://127.0.0.1:8088/requestDate  
  
C:\Users\GENE\source\repos\HTTP_SERVER\HTTP_SERVER\bin\Debug>
```

C:\Windows\System32\cmd.exe

```
C:\Users\GENE\source\repos\HTTP_SERVER\HTTP_CLIENT\bin\Debug>HTTP_CLIENT.exe  
Response : - 2021-07-02 오후 11:06:16  
  
C:\Users\GENE\source\repos\HTTP_SERVER\HTTP_CLIENT\bin\Debug>
```


2. Client에서 Server에 접속하여 파일 목록을 json형태로 전송하는 프로그램을 작성하시오.

- Input 폴더의 파일 목록을 전송하여 Output폴더에 저장
 - Client는 목록 전송 완료 후 종료
 - Server는 목록을 수신하여 '수신시간.json'파일로 저장하고 다시 Client 접속 대기
 - 요청 Method는 'POST'사용
 - Content Type은 "application/json" 사용

```
{  
  "Folder": "Input",  
  "Files": [  
    "close_x.png",  
    "config.js",  
    "desktop.js",  
    "japanese_over.png",  
    "main-cef-mac.css",  
    "main-cef-ui-theme.css",  
    "MyAll.txt",  
    "November Rain.txt",  
    "progress_bg_center.bmp",  
    "progress_bg_left.bmp",  
    "progress_bg_right.bmp",  
    "progress_center.bmp",  
    "progress_left.bmp",  
    "progress_right.bmp",  
    "test.exe"  
  ]  
}
```

부록 1) Redirection

표준 입력, 표준 출력의 리다이렉션 [편집]

리다이렉션은 특정한 문자열들을 명령어 사이에 두어서 추가되는 것이 보통이다. 일반적으로, 이러한 문자열들의 문법은 다음과 같다:

```
명령어1 > 파일1
```

위의 줄은 명령어1을 실행하며 이로써 나오는 출력물을 파일1로 내보낸다. 기존에 파일1의 존재하는 경우 기존의 파일 내용은 지우고 새롭게 추가된다. 파일의 끝에 출력물을 추가하려면 >> 연산자를 이용한다:

```
명령어1 >> 파일1
```

다음을 이용하면

```
명령어1 < 파일1
```

명령어1을 실행하되, 파일1이 입력의 대상이 된다. (자판을 이용하는 것과는 반대로)

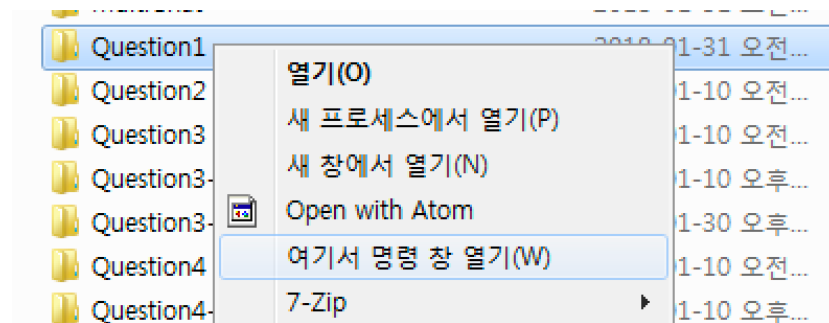
```
명령어1 < 들어오는파일 > 나가는파일
```

위의 줄은 두 개의 기능을 수행한다: 명령어1은 들어오는파일에서 내용을 불러들인 뒤 나가는파일에 기록한다.

<https://ko.wikipedia.org/wiki/리다이렉션>

부록 2) Java Command를 이용한 실행 방법

- 1. Question1 폴더에서 Shift + 오른쪽 마우스 클릭 후 여기서 명령창 열기 실행하여 Command 창 띄움



- 2. 명령창에 다음을 입력
java -cp bin 패키지.메인클래스명 < 입력파일명

```
관리자: C:\Windows\system32\cmd.exe
C:\dev\workspace_ssp\Question1>java -cp bin card.validator.client.ValidatorLauncher < ..\TestData\Question1\LOGIN_TEST.TXT
LOGIN FAIL
LOGIN FAIL
LOGIN FAIL
LOGIN FAIL
LOGIN FAIL
LOGIN FAIL
LOGIN FAIL
LOGIN SUCCESS
```

◆ 현재 날짜, 시각 문자열로 가져오기

```
1) LocalDateTime now = LocalDateTime.now();  
String strDT = now.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss.SSS"));  
  
2) long ct = System.currentTimeMillis();  
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss.SSS");  
String strCT = sdf.format(ct);
```

◆ 문자열 날짜, 시각 → Date 타입으로 변경

```
String strTime = "2022-03-31 21:40:15";  
SimpleDateFormat transFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");  
Date dt = transFormat.parse(strTime);
```

◆ Date 타입 → LocalDateTime 타입으로 변경

```
LocalDateTime dt = dt.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
```

◆ 시간 차이 계산

```
String start = "20220331142310"; String end = "20220331142420";  
SimpleDateFormat sf = new SimpleDateFormat("yyyyMMddHHmmss");  
Date dd1 = sf.parse(start); Date dd2 = sf.parse(end);  
long diff = dd2.getTime() - dd1.getTime();  
System.out.println(diff/1000); // Sec Difference
```

◆ 10진수 4자리로 출력

```
int a = 14;  
System.out.println(String.format("%04d", a));
```

출력) 0014

◆ 16진수 출력

```
int a = 14;  
System.out.println(String.format("%02X %02x", a, a));
```

출력) 0E 0e

◆ 소수점 출력

```
double b = 12.345678;  
System.out.println(String.format("%08.3f", b));
```

➔ 총 8자리 확보 후 앞자리에 0붙여주고, 뒤 3자리 소수점 반올림하여 출력

출력) 0012.346

◆ 위치로 자르기

```
String strTest = "My book | Your pen | His desk";  
System.out.println(strTest.substring(10)); ➔ 10자리부터 끝까지  
System.out.println(strTest.substring(10, 18)); ➔ 10자리에서 18자리까지
```

출력)
Your pen | His desk
Your pen

◆ Delimiter 사용하여 자르기

```
String strTest = "My book | Your pen | His desk";  
String [] words = strTest.split(" \| "); ➔ 파이프(|)문자는 이스케이프 처리(\|)가 필요함  
for (String item : words)  
    System.out.println(item);
```

출력)
My book
Your pen
His desk

◆ String → Byte Array

```
String strTest = "ABCD123";  
byte [] byteTest = new byte[80];  
byteTest = strTest.getBytes("UTF-8");  
for (byte b : byteTest)  
    System.out.print(b + " ");
```

출력) 65 66 67 68 49 50 51

◆ Byte Array → String

```
String strTest2 = new String(byteTest);  
System.out.println(strTest2);
```

출력) ABCD123

◆ Scanner 사용

```
Scanner sc = new Scanner(System.in);  
String input = sc.nextLine();  
System.out.println("Output : " + input);
```

입력) haha

출력) Output : haha

◆ BufferedReader 사용

```
InputStreamReader reader = new InputStreamReader(System.in);  
BufferedReader br = new BufferedReader(reader);  
String input = br.readLine();  
System.out.println("Output : " + input);
```

입력) haha

출력) Output : haha

◆ 현재 날짜, 시각 문자열로 가져오기

```
DateTime now = DateTime.Now;  
string strNow = now.ToString("yyyy/MM/dd HH:mm:ss.fff");
```

◆ 문자열 날짜, 시각 → DateTime 타입으로 변경

```
string strTime = "2022-03-31 21:40:15";  
DateTime dt = DateTime.ParseExact(strTime, "yyyy-MM-dd HH:mm:ss", null);
```

◆ 시간 차이 계산

```
string strTime1 = "20220331143610";  
string strTime2 = "20220331143720";  
DateTime dt1 = DateTime.ParseExact(strTime1, "yyyyMMddHHmmss", null);  
DateTime dt2 = DateTime.ParseExact(strTime2, "yyyyMMddHHmmss", null);  
TimeSpan ts = dt2 - dt1;  
Console.WriteLine(ts.TotalSeconds); // Sec Difference
```

출력) 70

◆ 10진수 4자리로 출력

```
int a = 14;  
Console.WriteLine(string.Format("{0:D4}", a));
```

출력) 0014

◆ 16진수 출력

```
int a = 14;  
Console.WriteLine(string.Format("{0:X2} {1:x2}", a, a));
```

출력) 0E 0e

◆ 소수점 출력

```
double b = 12.345678;  
Console.WriteLine(string.Format("{0:f3}", b));
```

➔ 소수점 3자리까지 반올림하여 출력

출력) 12.346

◆ 위치로 자르기

```
String strTest = "My book | Your pen | His desk";  
Console.WriteLine(strTest.Substring(10)); ➔ 10자리부터 끝까지  
Console.WriteLine(strTest.Substring(10, 8)); ➔ 10자리에서 8자리 이후까지
```

출력)
Your pen | His desk
Your pen

◆ Delimiter 사용하여 자르기

```
String strTest = "My book | Your pen | His desk";  
string[] words = strTest.Split(new[] { " | " }, StringSplitOptions.None);  
foreach (var item in words)  
    Console.WriteLine(item);
```

출력)
My book
Your pen
His desk

◆ String → Byte Array

```
string strTest = "ABCD123";  
byte[] byteTest = System.Text.Encoding.UTF8.GetBytes(strTest);  
foreach (var b in byteTest)  
    Console.Write(b + " ");
```

출력) 65 66 67 68 49 50 51

◆ Byte Array → String

```
string strTest2 = System.Text.Encoding.UTF8.GetString(byteTest);  
Console.WriteLine(strTest2);
```

출력) ABCD123