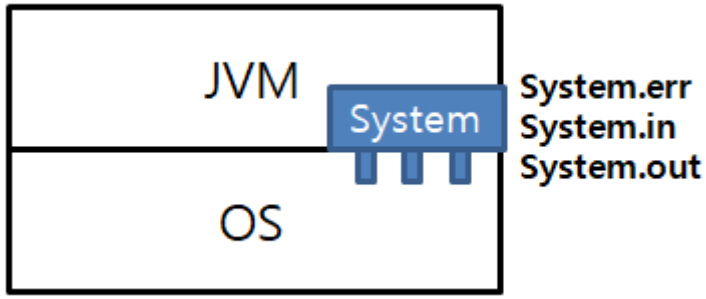
Text File 입출력 Program

경복대학교
소프트웨어융합과
배희호 교수
010-2369-4112
031-570-9600
hhbae@kbu.ac.kr

KeyBoard 입력

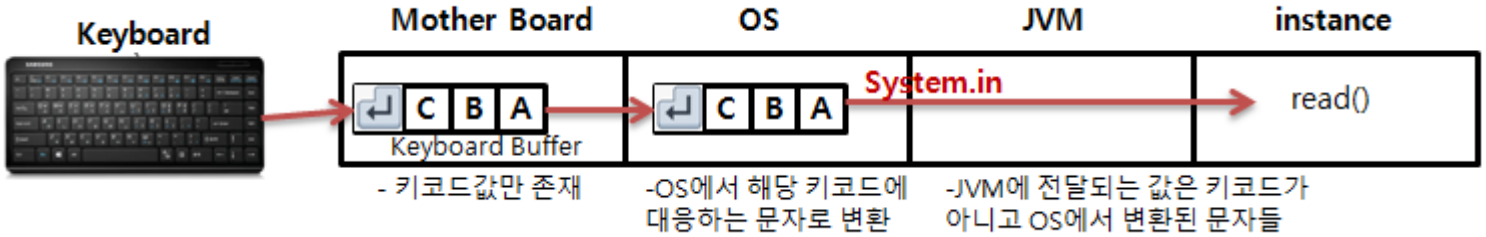
- KeyBoard 입력 내용을 어떻게 하면 화면에 표시할 수 있을까요?

Keyboard Input Stream



- JVM이 시작될 때 표준 error OutputStream, 표준 in InputStream, out OutputStream이 자동으로 생성됨
- 이 표준 Stream을 close하면 JVM을 다시 실행할 때까지 다시 Stream 연결을 못함
- Keyboard로 입력하면 MainBoard의 Keyboard Buffer에 값이 들어감
- Buffer의 크기는 256(0~255) Byte로 크기 만큼 입력 가능
- Enter가 입력되면 Buffer 내용이 OS로 전달됨

Keyboard Input Stream



- CPU에 따라 입력 순서를 그대로 처리하는 Little Endian(Intel 계열 CPU), 입력 순서를 거꾸로 처리하는 Big Endian 처리방식(RISC 계열 CPU(MAC계열))이 존재
- OS에 전달된 키 입력 내용은 JVM의 System.in(InputStream)을 통해 JVM으로 들어오게 되고 JVM에 존재하는 instance는 InputStream의 read()를 통해 해당 값을 읽어올 수 있음

Keyboard Input Stream 예제 0

```
public static void main(String[] args) {
    int data;
    int cnt = 0;
    try {
        while ((data = System.in.read()) != -1) {
            System.out.write((char) data);
            cnt++;
        }
    } catch (IOException e) {
        System.err.println(e.getMessage());
    }
    System.out.println(" 총 bytes : " + cnt);
}
```

끝내려면 <Ctl> + <D> 입력

Keyboard Input Stream 예제 1

- Keyboard에서 영문 입력 값 중 최초 입력 문자를 받아 출력하는 Program을 만들어보자

```
InputStream inputStream = System.in;
char ch;
ch = (char) inputStream.read(); // Keyboard로부터
                                1 문자 읽기
```

형 변환(캐스팅)

- Keyboard 입력 처리
  - System.in
  - System.in은 InputStream
  - read() 메소드로 Keyboard로 입력된 내용을 읽어 들일 수 있음

Keyboard Input Stream 예제 1

```
public static void main(String[] args) {
    InputStream inputStream = System.in;

    System.out.print("영문 키를 누르고 [Enter] > ");
    try {
        char ch = (char) inputStream.read();
        System.out.println("입력 문자 : " + ch);
    } catch (IOException e) {
        System.out.println(e.getMessage());
    }
}
```

```
영문 키를 누르고 [Enter] > A
입력 문자 : A

종료 코드 0(으)로 완료된 프로세스
```

Keyboard Input Stream 예제 2

- Keyboard에서 여러 영문자를 입력 받아 출력하는 Program을 만들어보자

•Windows는 [Enter] 키 입력 시 ₩r₩n이 함께 입력됨
  - ₩r (Carriage Return) : 그 줄 맨 앞으로 이동
  - ₩n (Line Feed) : 다음 줄로 이동

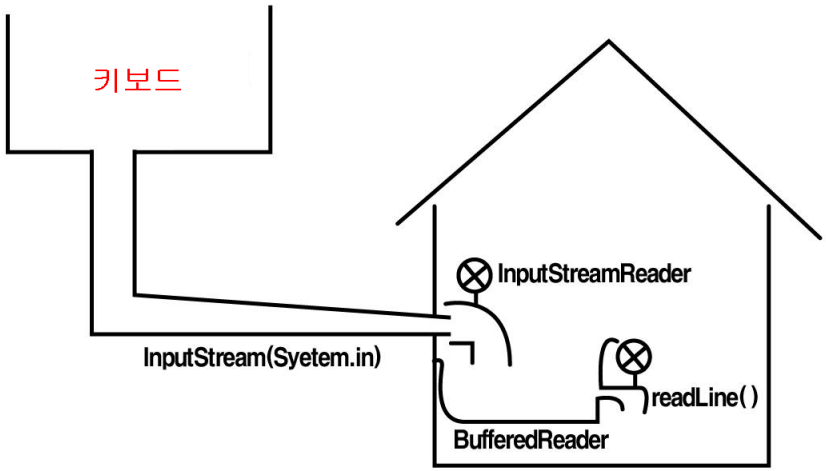
Keyboard Input Stream 예제 2

```
public static void main(String[] args) {
    InputStream inputStream = System.in;

    System.out.print("영문자 문장을 입력하고 [Enter] ");
    try {
        char ch;
        while(true) {
            ch = (char) inputStream.read();
            System.out.print(ch);
            if (ch == '₩n') { // enter는 13, '₩n'
                break;
            }
        }
    } catch (IOException e) {
        System.out.println(e.getMessage());
    }
}
```

Keyboard Input Stream 예제 3

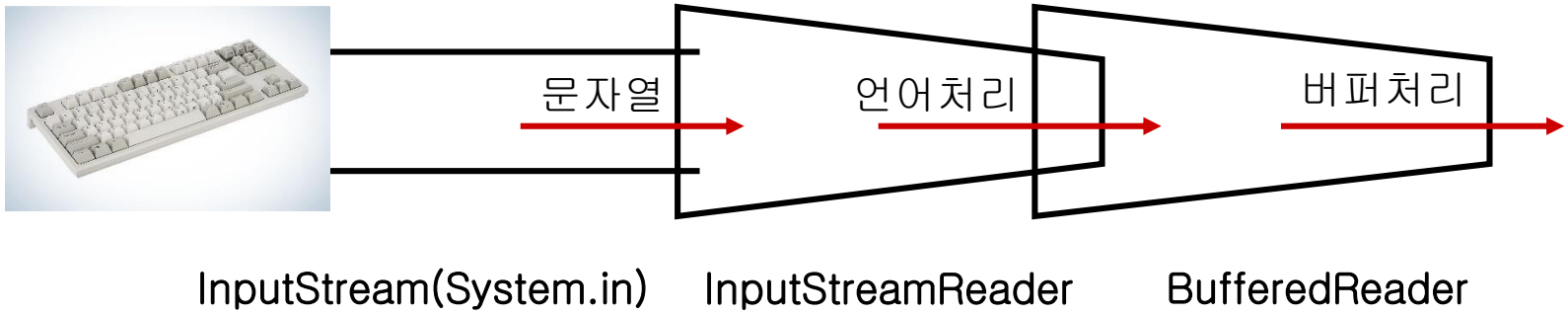
- 한글은 입력이 될까요?

- 한글은 8bit Stream을 사용하면 깨짐
  - 한글을 입력 받기 위해서는 8 Bit와 16 Bit Stream을 연결해야 함
- 2 Stream의 크기가 달라 바로 연결 못함
  - 2 Stream을 연결하는 Stream이 필요



Keyboard Input Stream 예제 3

- Filter : Stream에 연결해서 사용하는 Reader나 Buffer 같은 것



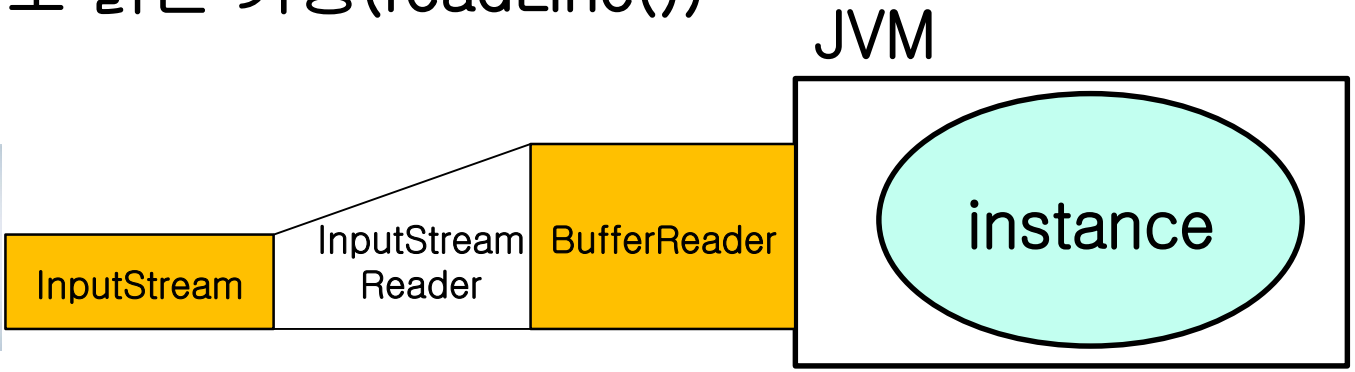

```
InputStream inputStream = System.in;
InputStreamReader streamReader = new InputStreamReader(inputStream);
BufferedReader reader = new BufferedReader(streamReader) ;
```

위 셋을 합하면

```
BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
```

Keyboard Input Stream 예제 3

- InputStream
  - System.in
- InputStreamReader
  - 8 Bit – 16 Bit Stream 결합
  - 읽어 들인 charset을 변환(Parameter 2개 받는 생성자 사용 시)
- BufferedReader
  - 속도 개선
  - 줄 단위로 읽는 기능(readLine())



Keyboard Input Stream 예제 3

- System.in을 통해 문자열을 받으려면, Buffer를 사용해야 함

byte

InputStream 클래스

InputStream is = System.in;

InputStreamReader 클래스

char

BufferedReader 클래스

행

- InputStreamReader 클래스 : Byte Stream과 문자 Stream 사이의 중개
- BufferedReader 클래스 : Buffer에 문자 Stream을 읽어 들임

Keyboard Input Stream 예제 3

```
public static void main(String[] args) {
    InputStream stream = System.in;
    InputStreamReader streamReader = new InputStreamReader(stream); // has-a
    BufferedReader reader = new BufferedReader(streamReader);       // is-a

    System.out.print("문장을 입력하고 [Enter] ");
    try {
        String statement = reader.readLine();
        System.out.println(statement);
        reader.close();
    } catch (IOException ie) {
        System.out.println(ie.getMessage());
    }
}
```

Keyboard Input Stream 예제 4

```
public static void main(String[] args) {
    int readByte;
    System.out.println(" 문자를 입력하세요.(종료 <Ctrl>+<D> ");
    InputStreamReader streamReader = new InputStreamReader(System.in);
    OutputStreamWriter streamWriter = new OutputStreamWriter(System.out);
    while (true) {
        try {
            if ((readByte = streamReader.read()) == -1)
                break;
            streamWriter.write(readByte);
            streamWriter.flush();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
    System.out.println(" 프로그램을 종료 합니다.");
}
```

처리 내용을 화면에 출력하기

- System.out
  - Console로 Data를 출력하기 위해 사용하는 System 클래스의 out 정적 Field
  - out은 PrintStream 타입의 필드
  - PrintStream은 OutputStream의 하위 클래스
  - out Field를 OutputStream 타입으로 변환해서 사용할 수 있음

```
OutputStream outputStream = System.out;
```

처리 내용을 화면에 출력하기

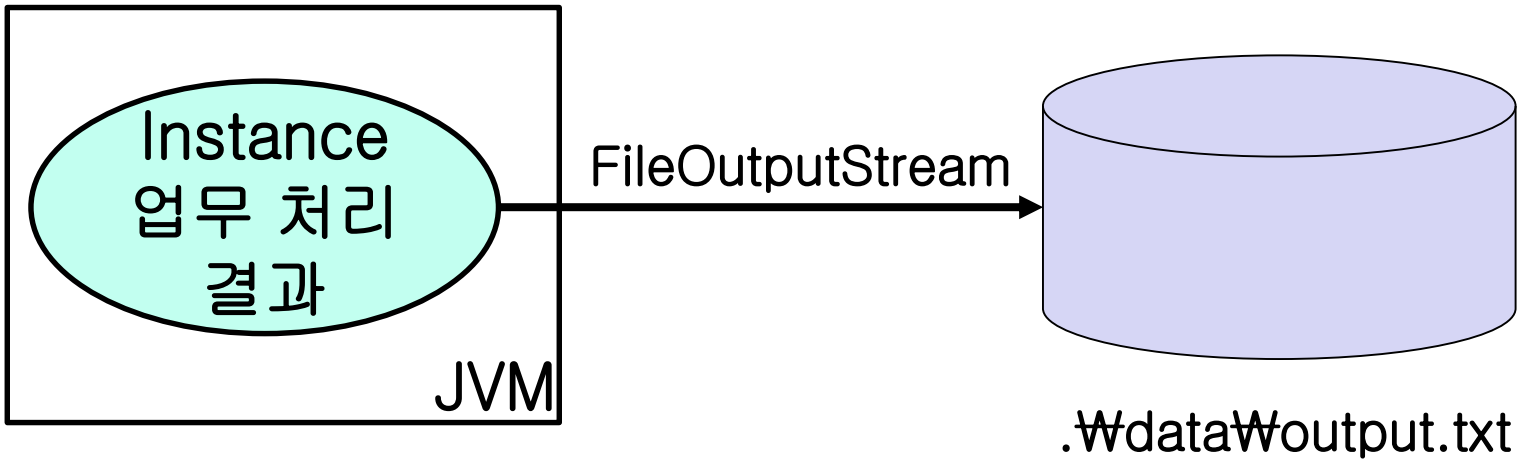
```
public static void main(String[] args) {
    try {
        OutputStream outputStream = System.out;
        for (byte b = 48; b < 58; b++) {
            outputStream.write(b);  // 아스키 코드 48~ 57까지의 문자 출력
        }
        outputStream.write('₩n');
        for (byte b = 97; b < 123; b++) {
            outputStream.write(b);
        }
        outputStream.write(10);
        String hangul = "우리는 행복합니다";
        byte[] hangulBytes = hangul.getBytes();
        outputStream.write(hangulBytes);
        outputStream.flush();
    } catch (IOException e) {
        System.out.println(e.getMessage());
    }
}
```

처리 내용을 화면에 출력하기

```
public static void main(String[] args) {
    try {
        PrintStream output = new PrintStream(System.out);
        int age = 25;

        output.printf("I am %d years old.", age);
        output.close();
    } catch(Exception e) {
        System.out.println(e.getMessage());
    }
}
```

처리 내용을 Text File로 출력하기

- Program의 처리 내용을 저장소에 Text File로 저장하는 방법을 알아보자



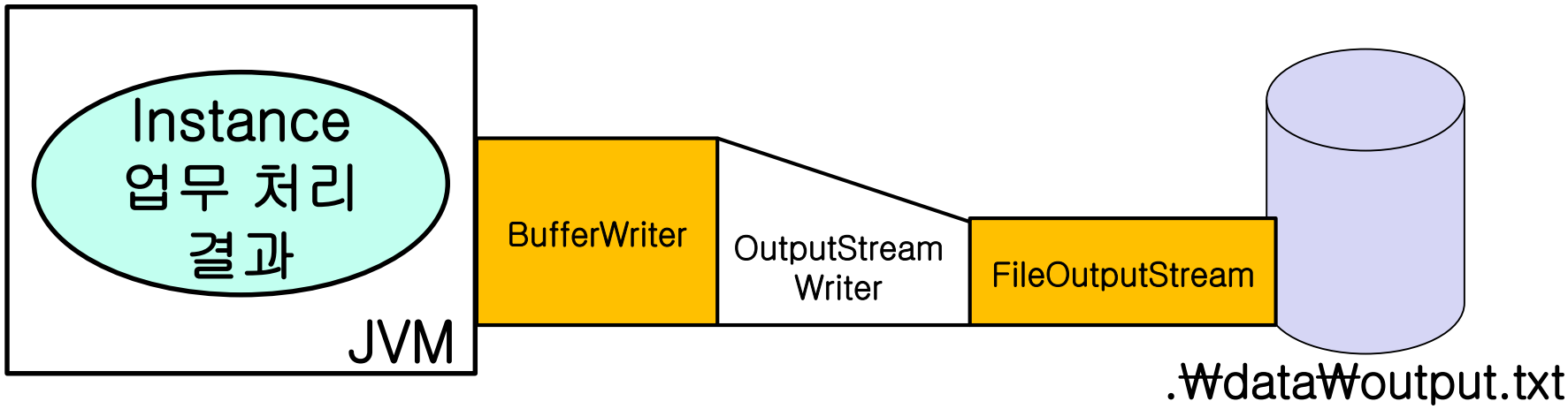
- Text File 출력
  - 8 Bit Stream
    - FileOutputStream
      - 입력된 경로의 File을 생성
      - 없으면 생성하고 File이 존재하면 덮어 씀

Text File 출력하기(I)

```
public static void main(String[] args) {
    String outfile = ".//data//test.txt";
    int data = 65;  // 'A'

    try {
        FileOutputStream outputStream = new FileOutputStream(outfile);
        outputStream.write(data); // 값을 스트림에 기록(수에 대한 문자가 출력됨)
        outputStream.flush();
        System.out.println("파일 기록 완료");
        outputStream.close();
    } catch (IOException e) {
        System.out.println(e.getMessage());
    }
}
```

Text File 출력하기(II)

- 한글이 포함된 문자열 쓰기
  - 8 Bit Stream과 16 Bit Stream을 사용하면 문자열을 File에 기록 가능
  - 16 Bit OutputStream들은 Writer를 상속받는 클래스들
  - OutputStreamWriter를 사용하면 문자열을 File로 쓸 수 있음
  - 속도 개선을 위해 BufferedWriter를 사용

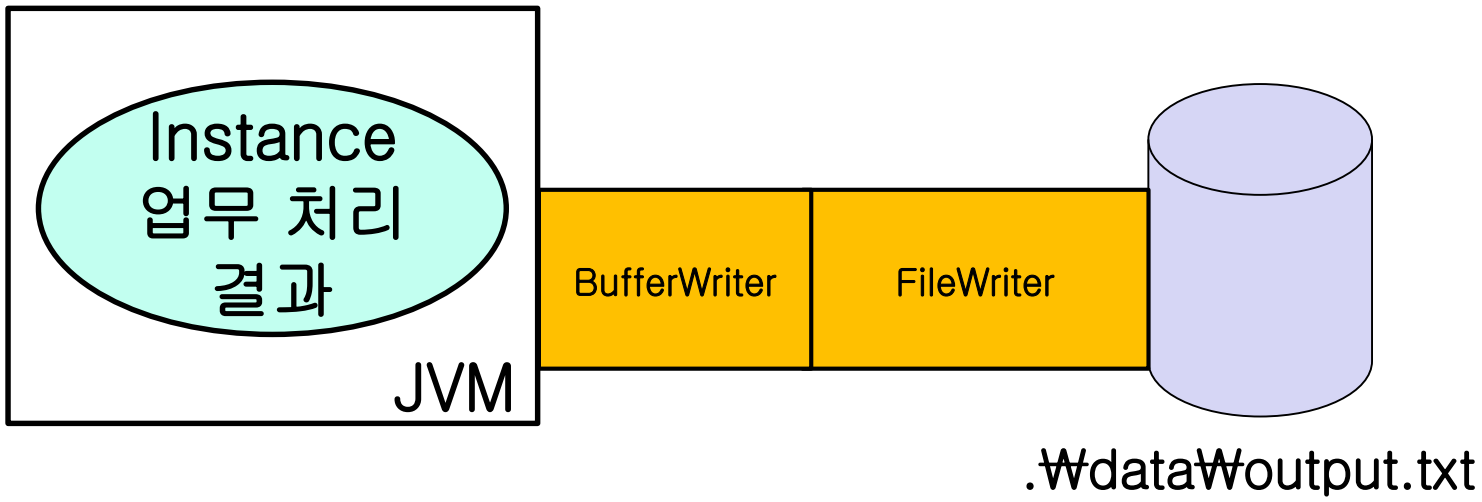


Text File 출력하기(II)

```
public static void main(String[] args)  {
    String outfile = ".₩₩data₩₩test2.txt";
    String message = "I love you.₩n나는 당신을 사랑합니다";

    try {
        FileOutputStream outputStream = new FileOutputStream(outfile);
        OutputStreamWriter writer = new OutputStreamWriter(outputStream);
        BufferedWriter bufferedWriter = new BufferedWriter(writer);
        bufferedWriter.write(message);
        bufferedWriter.flush();
        bufferedWriter.close();
        writer.close();
        outputStream.close();
        System.out.println("파일 기록 완료");
    } catch (IOException e) {
        System.out.println(e.getMessage());
    }
}
```

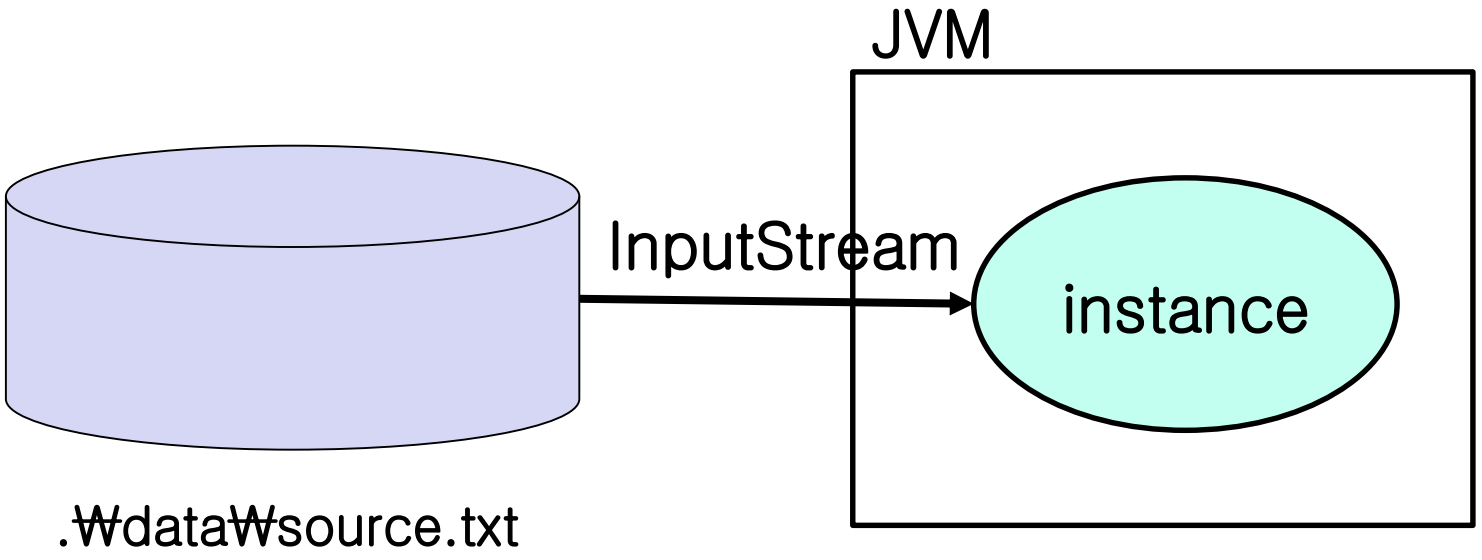
Text File 출력하기(III)

- FileWriter
  - 16 bit Stream을 바로 사용하기
  - FileWriter 사용



Text File 내용 읽기

- 저장소에 저장되어 있는 Text File을 읽어보자



File을 FileInputStream을 사용해서 read()로
1 Byte씩 읽어온다면 한글은 깨짐

Text File 내용 읽기

- Text File 준비
  - 영문과 한글이 혼합되어 있는 File 준비

```
I am a boy.
나는 한국인 입니다
```

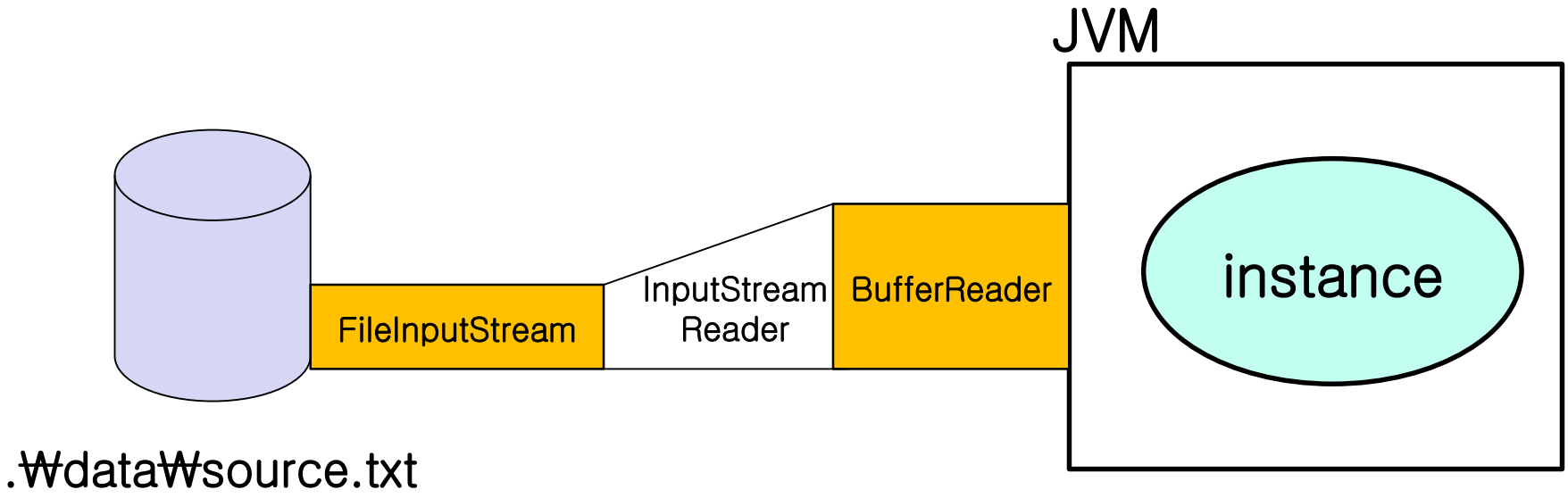
  - Project File에서 data 폴더를 만들고 그 아래 저장

Text File 내용 읽기(I)

```
public static void main(String[] args) {
    String datafile = ".₩₩data₩₩source.txt";

    File file = new File(datafile);
    if (file.exists()) {
        try {
            FileInputStream inputStream = new FileInputStream(file);
            int data;
            while ((data = inputStream.read()) != -1) { // 읽어들인 내용이 있을 때까지
                System.out.print((char) data);
            }
            inputStream.close();
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }
    } else {
        System.out.println("파일이 없음");
    }
}
```

Text File 내용 읽기(II)

- 한글 포함한 파일 입력 받기
  - 한글 처리를 하기 위해 8 Bit Stream과 16 Bit Stream을 연결하여 File Data를 읽어 들여야 함
  - InputStreamReader 사용



Text File 내용 읽기(II)

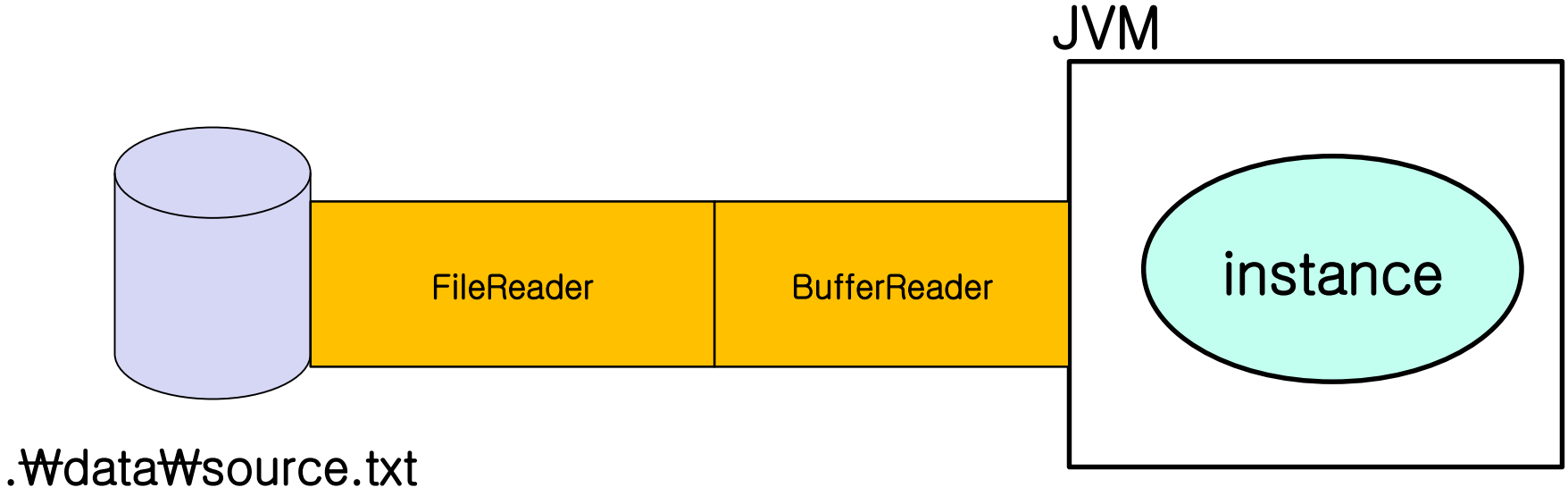
```
public static void main(String[] args) {
    String datafile = ".₩₩data₩₩source.txt";

    File file = new File(datafile);
    if (file.exists()) {
        try {
            FileInputStream inputStream = new FileInputStream(file);
            InputStreamReader streamReader = new InputStreamReader(inputStream);
            BufferedReader reader = new BufferedReader(streamReader);
            String line;
            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }
            reader.close();
            streamReader.close();
            inputStream.close();
```

Text File 내용 읽기(II)

```
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }
    } else {
        System.out.println("파일이 없음");
    }
}
```

Text File 내용 읽기(III)

- FileReader
  - 원천 Data가 한글을 포함한 Data라면 16 Bit Stream을 바로 사용




```
File file = new File("경로");
FileReader fileRader = new FileReader(file); // File 객체로 바로 16bit Stream 생성
BufferedReader reader = new BufferedReader(fileReader);
```

Text File 내용 읽기(III)

```
public static void main(String[] args) {
    String datafile = ".₩₩data₩₩source.txt";

    File file = new File(datafile);
    if (file.exists()) {
        try {
            FileReader fileReader = new FileReader(file);
            BufferedReader reader = new BufferedReader(fileReader);
            String line;
            while ((line = reader.readLine()) != null)
                System.out.println(line);
            reader.close();
            fileReader.close();
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }
    } else {
        System.out.println("파일이 없음");
    }
}
```

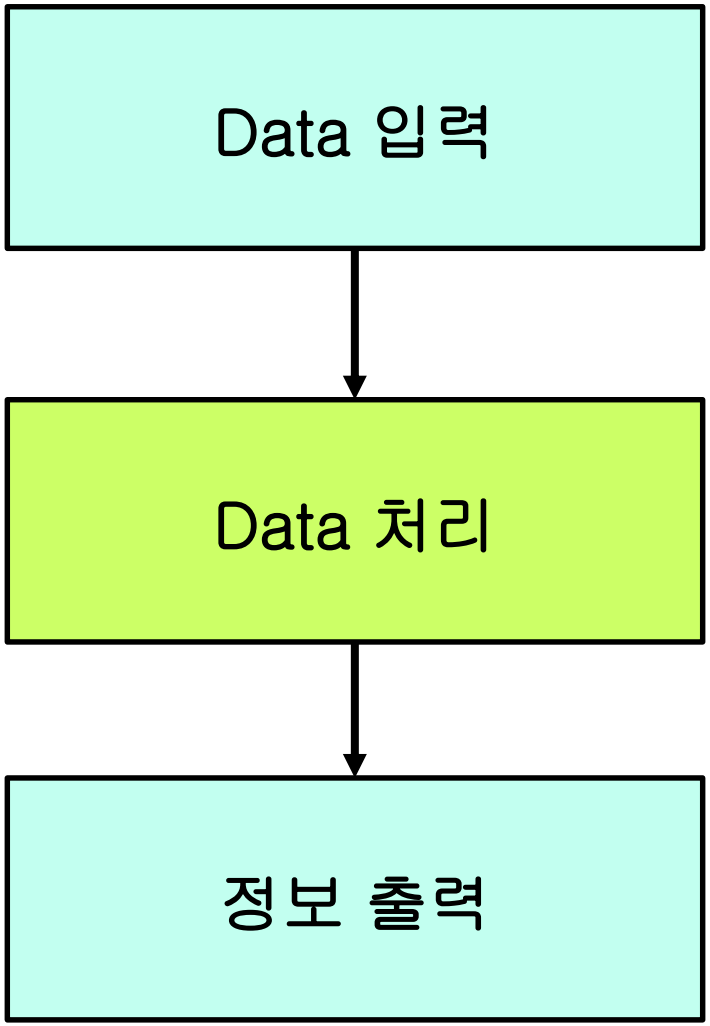
Character Set

- EUC-KR
  - 한글 Encoding 방식 중 하나, 현재 산업 표준
  - 글자 Code Set에 쓸 문자를 현대 한국어에서 빈도가 높은 한글을 추려내어 '가'부터 '힝'까지 배당한 후 문자를 기록하는 방식
  - 과거에 사용빈도가 많은 단어를 토대로 만듦
  - '아햏햏'같은 단어는 사용을 못함, '햏'이란 문자가 없기 때문에 사용을 못함
  - 초기 MS Office가 EUC-KR 기반 이었음
  - 문서가 저장 안되는 문제가 발생
  - MS에서 해결하고자 만들어 줌 MS949(CP949)
- CP949 (code page 949)
  - MS Windows의 기본 Code Page
  - 한글 Encoding의 한 종류, EUC-KR의 확장형

Character Set

- UTF-8
  - UniCode를 위한 가변 길이 문자열 Encoding 방식 중 하나
  - Universal Coded Character Set + Transformation Format – 8 bit의 약자
  - 한글은 3 Byte로 표기

File 입출력



- ✓ 대입 방법(연산자)
- ✓ KeyBoard에서 입력
- ✓ File 입력
- ✓ DataBase에서 검색

- ✓ Monitor로 출력
- ✓ File 출력
- ✓ DataBase에 삽입

File 입출력

- File 입출력의 순서
  - 입력 혹은 출력에 적절한 File 객체를 생성

```
File file = new File(".//data//temp.txt");
```

  - File을 연다 (혹은 개방한다고 표현)
    - 입출력 Stream 연결 (보조 Stream 연결)
    - 열기에 실패했을 경우의 예외 처리를 해줌
    - try ~ catch 문을 작성하거나 클래스 자체에 throws 사용
  - 원하는 작업을 처리
  - File을 닫음

File 입출력

- Program에서 사용된 Data를 영구적으로 File 형태로 저장하여 다음 에 다른 Program에서 사용하게 하기 위해 필요한 기능
- Program에서 생성된 Data를 File 형태로 Computer의 HDD나 보조 기억장치에 저장한 다음 다른 Program에서 이 Data를 읽어 사용하는 기능

File Read

- 모든 File은 Character로 이루어져 있다고 생각하면 됨
- 심지어 Image File도 Character로 이루어져 있기 때문에 File을 읽을 때 Character를 읽는다고 생각하면 됨

- JAVA에서 File을 읽는 방법은 한 글자씩 읽는 방법과, File을 한 Line씩 읽는 방법이 있음
  - FileReader (한 글자씩 읽는 방법)
  - BufferedReader (한 Line씩 읽는 방법)
  - Scanner (한번에 한 Line씩 읽음)
  - FileUtils (한번에 모두 읽음)
  - Files (한번에 모두 읽음)

File Read

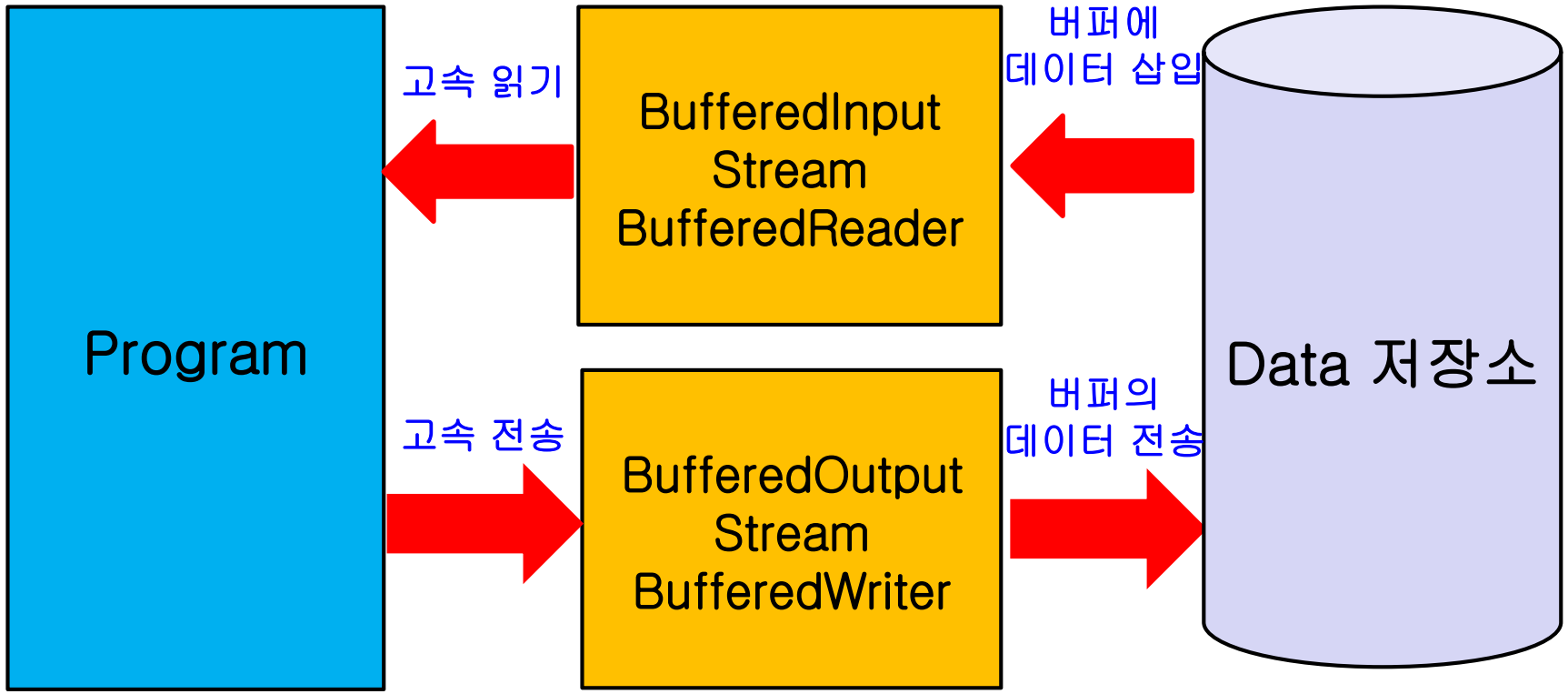
- FileReader
  - File을 읽을 때는 기본적으로 java.io.FileReader가 사용
  - 생성된 File을 객체로 선언하고, 선언된 File 객체를 인자로 FileReader를 선언하면 File에 직접 접근하여 File 읽을 수 있음
  - FileReader는 File을 한 글자씩 읽어올 수 있음

File Read

- BufferedReader
  - FileReader의 확장판 같은 개념인 BufferedReader
  - File을 한 문장씩 읽기 때문에 FileReader 보다는 훨씬 빠름
  - File을 읽는 Code의 대부분은 BufferedReader로 만들어져 있음

File Read

- Buffer 사용



File Read

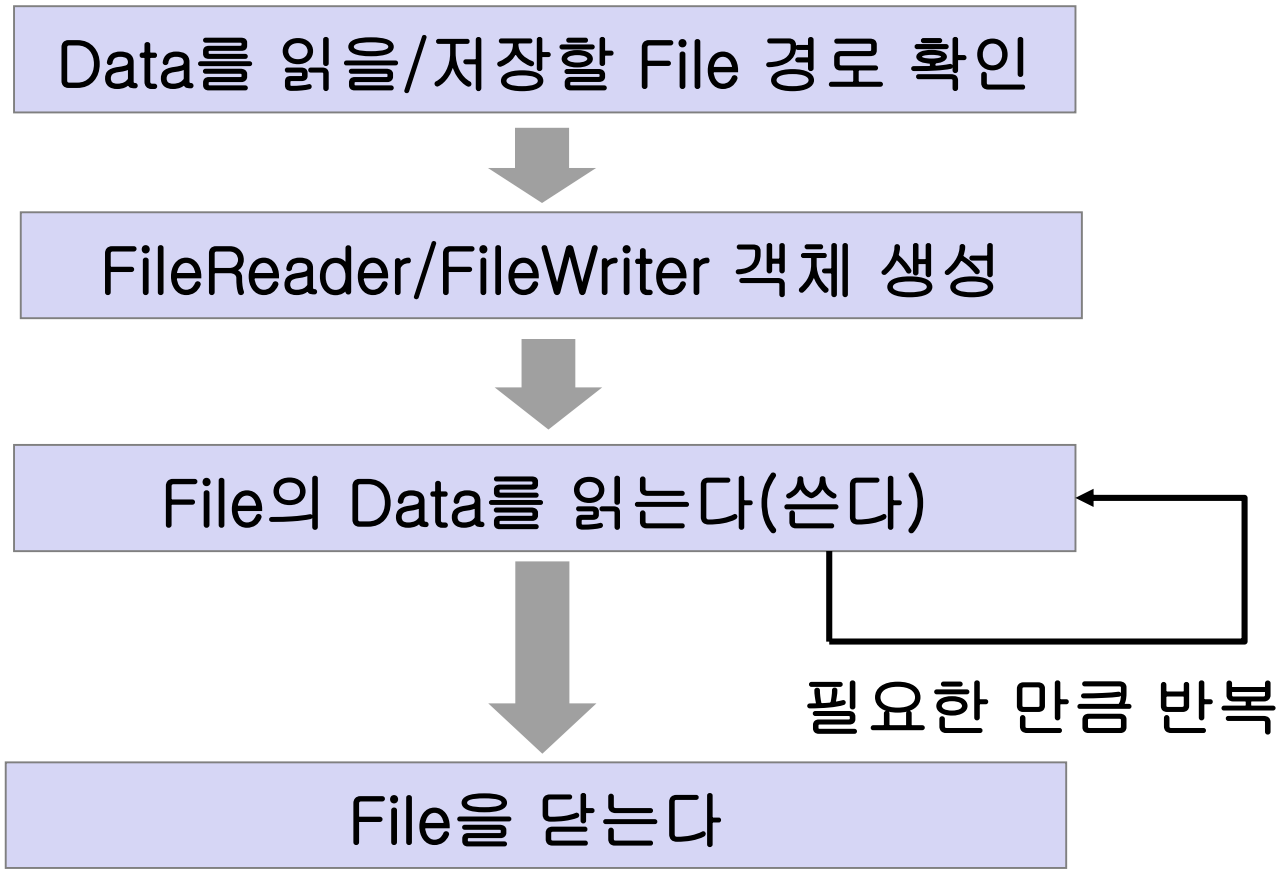
- Scanner
  - Scanner 클래스를 이용하면, File의 Text를 Delimiter를 이용하여 잘라서 읽을 수 있음
  - 기본 delimiter는 줄 바꿈을 포함한 공백('₩t', '₩f', '₩r', ' ', '₩n')
  - 다양한 Data형으로 읽을 수 있어 File 처리 Program에 많이 사용

File Read

- Files
  - java.nio.file.Files 클래스는 Java 7이후부터 사용 가능
  - Files 클래스는 모두 static 메소드로 구성이 되어있음
  - Files 클래스를 이용하면, Text File 내용 전체를 List나 배열, String에 쉽게 담을 수 있음

File Read 예제

- File로부터 Data를 읽는(쓰는) 단계



File Read 예제

- 아래 File을 준비하여 5가지 방법으로 읽어보자

- File 준비 (input.txt)

```
hello
how are you
nice to meet you
안녕하세요, 경복대학교에서 공부합니다
```

File Read 예제

- FileReader

```
public static void main(String[] args) {
        final String filename = ".₩₩data₩₩input.txt";
        File file = new File(filename);
        if (file.exists()) {
            try{
                FileReader filereader = new FileReader(file);
                int ch;
                while((ch = filereader.read()) != EOF) {
                    System.out.print((char) ch);
                }
                System.out.printf("₩n");
                filereader.close();
            }catch(IOException e){
                System.err.println(e.getMessage());
            }
        } else {
            System.out.printf("₩n %s파일이 존재하지 않습니다.", filename);
        }
    }
}
```

File Read 예제

- BufferedReader

```
public static void main(String[] args) {
    final String filename = ".₩₩data₩₩input.txt";
    File file = new File(filename);
    if (file.exists()) {
        try{
            FileReader filereader = new FileReader(file);
            BufferedReader bufReader = new BufferedReader(filereader);
            String line;
            while((line = bufReader.readLine()) != null){
                System.out.println(line);
            }
            bufReader.close();
        } catch(IOException e){
            System.err.println(e.getMessage());
        }
    } else {
        System.out.printf("₩n %s파일이 존재하지 않습니다.", filename);
    }
}
```

File Read 예제

- Scanner

```
public static void main(String[] args) throws FileNotFoundException {
    final String filename = ".₩₩data₩₩input.txt";
    File file = new File(filename);
    if (file.exists()) {
        Scanner input = new Scanner(file);
        while (input.hasNextLine()) {
            String line = input.nextLine();
            System.out.println(line);
        }
        input.close();
    }else {
        System.out.printf("₩n %s파일이 존재하지 않습니다.", filename);
    }
}
```

File Read 예제

- FileUtils

```
public static void main(String[] args) {
    final String filename = ".₩₩data₩₩input.txt";

    File file = new File(filename);
    if (file.exists()) {
        try {
            List<String> list = FileUtils.readLines(file, StandardCharsets.UTF_8);
            for (int i = 0; i < list.size(); i++)
                System.out.println(list.get(i));
        } catch (IOException e) {
            System.err.println(e.getMessage());
        }
    } else {
        System.out.printf("₩n %s파일이 존재하지 않습니다.", filename);
    }
}
```

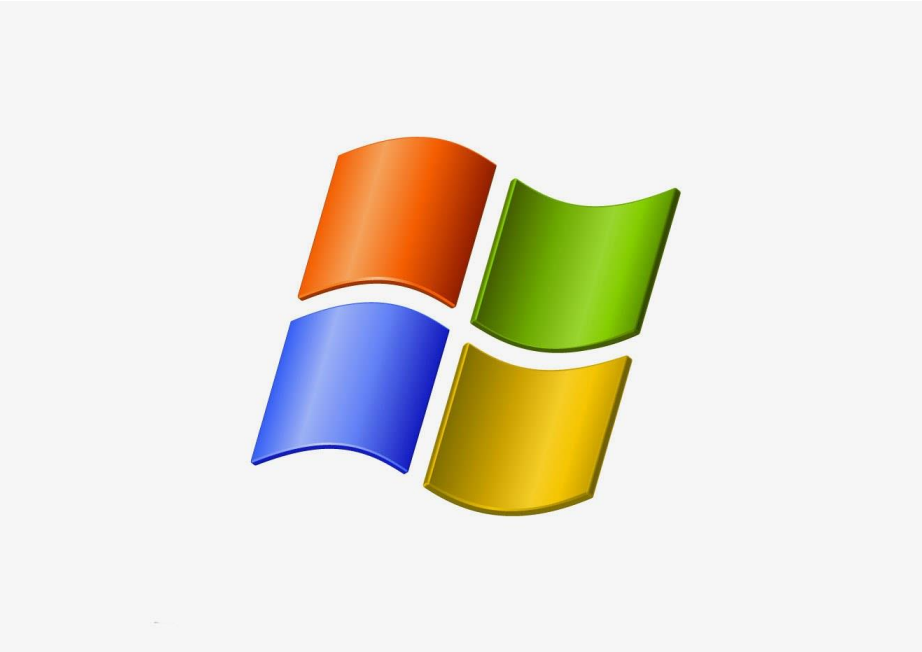
File Read 예제

- Files

```
public static void main(String[] args)  {
    final String filename = ".₩₩data₩₩input.txt";
    Path path = Paths.get(filename);
    Charset cs = StandardCharsets.UTF_8;
    if (Files.exists(path)) {
        try {
            List<String> list = Files.readAllLines(path, cs);
            for(String readLine : list){
                System.out.println(readLine);
            }
        } catch (IOException e) {
             System.err.println(e.getMessage());
        }
    } else {
        System.out.printf("₩n %s파일이 존재하지 않습니다.", filename);
    }
}
```

Windows의 system.ini File 읽기

- FileInputStream을 이용하여 사용자 컴퓨터의 windows 디렉터리에 있는 system.ini 파일을 읽고 화면에 출력하라.
- system.ini 파일은 텍스트 파일이다



Windows의 system.ini File 읽기

```
import java.io.*;

public class FileInputStreamEx {
    public static void main(String[] args) {
        FileInputStream input = null;
        try {
            input = new FileInputStream("c:₩₩windows₩₩system.ini");
            int ch;
            while ((ch = input.read()) != -1) {
                System.out.print((char) ch);
            }
            input.close();
        } catch (IOException e) {
            System.out.println("입출력 오류");
        }
    }
}
```

파일 끝을 만나면 -1 리턴

File Write

- File에 Data를 쓰는 방법
  - FileOutputStream (Byte 단위 쓰기)
  - FileWriter (문자 단위 쓰기)
  - BufferedWriter
  - PrintWriter (print() 메소드 사용)
  - Files
  - OutputStreamWriter (보조 스트림)
  - DataOutputStream (스스로 출력할 수 없음)

File Write

- FileOutputStream
  - InputStream과 마찬가지로 OutputStream 역시 Byte 단위로 Data를 처리하는 클래스
  - FileOutputStream은 OutputStream 클래스를 상속받아 만든 클래스로 Byte 단위로 Data를 처리하게끔 됨
  - FileOutputStream에 값을 쓸 때는 byte 배열로 써야 하므로 String을 byte 배열로 바꾸어 주는 getBytes() 메소드를 이용
  - 줄 바꿈을 위해서 ₩r₩n이 필요

File Write

- FileWriter
  - File에 문자를 쓰기 위해 활용되는 클래스로 OutputStreamWriter를 상속
  - File을 쓰는 기본적인 클래스는 java.io.FileWriter
  - FileReader와 쌍을 이루고 있으며 기본적으로 문자열, int 등을 File에 쓰는 것이 가능
  - FileWriter는 기존의 File의 내용을 모두 삭제하고, 그 위에 새로 쓰기 때문에 신중하게 사용해야 함
  - FileWriter 클래스만을 활용하여 File에 내용을 쓰는 경우, FileWriter 객체를 생성하고 그 인자로 File Instance를 담은 뒤에 여기에 내용을 쓰고 flush()를 호출 함
  - File 이어 쓰기 기능은 FileWriter의 선언 부분에 두 번째 인자로 boolean 타입의 true를 같이 보내면 됨

File Write

- BufferedWriter
  - BufferedWriter 역시 File 쓰기를 담당
  - FileWriter 객체를 인자로 BufferedWriter 객체를 생성하고, 그 사용법은 FileWriter와 동일
  - 이름에서 느껴지는 것처럼 BufferedWriter는 Buffer를 갖고 있음. 그리고 BufferedWriter가 다음 2가지 조건에서 FileWriter보다 효과적
    - Buffer보다 작은 크기의 쓰기일 경우
    - 한 곳이 아닌 여러 곳에서 쓰기가 이루어 지는 경우
  - 항상 BufferedWriter가 효과적인 것은 아님

File Write

- PrintWriter
  - PrintWriter는 BufferedWriter의 차이는 단 하나로 개행(‘₩n’)을 하는 문자열을 넣을 수 있음
  - 다른 쓰기 메소드들은 개행을 하는 문자를 따로 적어 주어야 했는데, PrintWriter는 .println() 메소드를 통해 File에 개행 된 문자열을 넣을 수 있음
  - .print()는 .write() 메소드와 동일

File Write

- Files
  - String을 byte로 변환하여 write()의 인자로 전달하면 됨
  - Files.write() 내부에서 OutputStream으로 문자열을 File에 씀
  - close()도 호출해 주기 때문에 사용자가 close()를 호출하지 않아도 됨

File Write 예제 1

- 다음과 같은 문자열을 File로 저장하는 프로그램을 만들어 보자

```
KOREA Fighting!
안녕하세요, 경복대학교에서 공부합니다
```

File Write 예제 1

- FileOutputStream

```
public static void main(String[] args) {
    String path = ".₩₩data₩₩output.txt";
    String source = "KOREA Fighting!₩n" +
            "안녕하세요, 경복대학교에서 공부합니다₩n";
    byte[] intxt = source.getBytes();

    try {
        FileOutputStream output = new FileOutputStream(path, false);
        //true 이어서 쓰고 , false면 새로 씀
        output.write(intxt);
        System.out.println("파일 생성 성공");
        output.close();
    } catch (IOException ex) {
        System.err.println(ex.getMessage());
    }
}
```

File Write 예제 1

- FileWriter

```
public static void main(String[] args) {
    String path = ".₩₩data₩₩output.txt";
    String source = "KOREA Fighting!₩n" +
            "안녕하세요, 경복대학교에서 공부합니다₩n";
    char[] intxt = new char[source.length()];
    source.getChars(0, source.length(), intxt, 0);
    try {
        FileWriter writer = new FileWriter(path);
        writer.write(intxt);
        writer.append(source);
        System.out.println("파일 생성 성공");
        writer.close();
    } catch (IOException e) {
        System.err.println(e.getMessage());
    }
}
```

File Write 예제 1

- BufferedWriter

```
public static void main(String[] args) {
    String path = ".₩₩data₩₩output.txt";
    String source = "KOREA Fighting!₩n" +
                    "안녕하세요, 경복대학교에서 공부합니다₩n";
    char[] intxt = new char[source.length()];
    source.getChars(0, source.length(), intxt, 0);
    try {
        FileWriter writer = new FileWriter(path);
        BufferedWriter bufferedWriter = new BufferedWriter(writer);
        bufferedWriter.write(intxt);
        bufferedWriter.append(source);
        System.out.println("파일 생성 성공");
        bufferedWriter.close();
    } catch (IOException e) {
        System.err.println(e.getMessage());
    }
}
```

File Write 예제 1

- PrintWriter

```
public static void main(String[] args) {
    String path = ".₩₩data₩₩output.txt";
    String source = "KOREA Fighting!₩n" +
                    "안녕하세요, 경복대학교에서 공부합니다₩n";
    char[] intxt = new char[source.length()];
    source.getChars(0, source.length(), intxt, 0);

    try {
        FileWriter writer = new FileWriter(path);
        PrintWriter printWriter = new PrintWriter(writer);
        printWriter.print(source);
        printWriter.println(intxt);
        System.out.println("파일 생성 성공");
        printWriter.close();
    } catch (IOException ex) {
        System.err.println(ex.getMessage());
    }
}
```

File Write 예제 1

- Files

```
public static void main(String[] args) {
    String inputfile = ".₩₩data₩₩output.txt";
    String source = "KOREA Fighting!₩n" +
                    "안녕하세요, 경복대학교에서 공부합니다₩n";
    byte[] intxt = source.getBytes();

    Path path = Paths.get(inputfile);
    try {
        Files.write(path, intxt);
        System.out.println("파일 생성 성공");
    } catch (IOException ex) {
        System.err.println(ex.getMessage());
    }
}
```

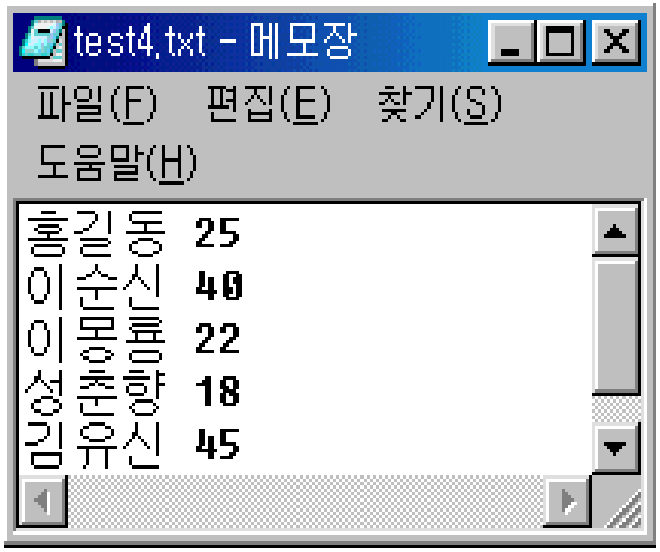
File Write 예제 1

- OutputStreamWriter

```
public static void main(String[] args) {
    String input = ".₩₩data₩₩output.txt";
    String source = "KOREA Fighting!₩n" +
            "안녕하세요, 경복대학교에서 공부합니다₩n";
    char[] intxt = new char[source.length()];
    source.getChars(0, source.length(), intxt, 0);
    try {
        FileOutputStream output = new FileOutputStream(input, false);
        //true면 이어서 쓰고 , false면 새로 씀
        Writer writer = new OutputStreamWriter(output);
        writer.write(intxt);
        writer.append(source);
        System.out.println("파일 생성 성공");
        output.close();
    } catch (IOException ex) {
        System.err.println(ex.getMessage());
    }
}
```

File Write 예제 2

- Keyboard 로부터 성명, 나이를 입력하여 파일에 쓰기를 연습해보자



File Write 예제 2

- Keyboard에서 입력하는 방법
  - Scanner/ObjectOutputStream 방법
    - Byte 단위 입출력
    - Editor를 이용하여 Data File을 확인 불가능
    - ObjectInputStream으로 읽어야 함
  - Scanner/PrintWriter 방법

File Write 예제 2

```
public static void main(String[] args) {
    String filename = ".₩₩data₩₩test.txt";

    Scanner keyboard = new Scanner(System.in);
    try {
        PrintWriter printWriter = new PrintWriter(new FileWriter(filename));
        int count = 0;
        while (true) {
            System.out.print(" 성명 입력 : ");
            String name = keyboard.next();
            System.out.print(" 나이 입력 : ");
            int age = keyboard.nextInt();
            printWriter.printf("%3s %2d₩n", name, age);
            count++;
            System.out.print(" 계속 하시겠습니까 ? (Y/N) ");
            char enter = keyboard.next().charAt(0);
            if (enter == 'N' || enter == 'n')
                break;
        }
```

File Write 예제 2

```
            System.out.println(count + "개 입력하였습니다");
            printWriter.close();
        } catch (IOException e) {
            System.err.println(e.getMessage());
        }
}
```

File Write 예제 2

```
public class Man implements Serializable {
    private static final long serialVersionUID = 12345L;
    private String name;
    private int age;

    public Man(String name, int age) {
        this.name = name;
        this.age = age;
    }
}
```

File Write 예제 2

```
public static void main(String[] args) {
    String filename = ".₩₩data₩₩test.ser";

    Scanner keyboard = new Scanner(System.in);
    try {
        FileOutputStream stream = new FileOutputStream(filename);
        ObjectOutputStream outputStream = new ObjectOutputStream(stream);
        int count = 0;
        while (true) {
            System.out.print(" 성명 입력 : ");
            String name = keyboard.next();
            System.out.print(" 나이 입력 : ");
            int age = keyboard.nextInt();
            outputStream.writeObject(new Man(name, age));
            count++;
            System.out.print(" 계속 하시겠습니까 ? (Y/N) ");
            char enter = keyboard.next().charAt(0);
            if (enter == 'N' || enter == 'n')
                break;
        }
```

File Write 예제 2

```
            System.out.println(count + "개 입력하였습니다");
            outputStream.close();
        } catch (IOException e) {
            System.err.println(e.getMessage());
        }
}
```

FileOutputStream 이용한 File 쓰기

- 정수 타입의 결과 값을 FileOutputStream을 이용하여 파일에 저장한다
- 다시 이 파일에서 정수형 변수로 읽고 이전에 계산된 결과 값과 같은지 확인하라.

FileOutputStream 이용한 File 쓰기

```
public static void main(String[] args) {
    String path = ".₩₩data₩₩test.out";
    try {
        FileOutputStream fout = new FileOutputStream(path);
        for (int i = 0; i < 10; i++) {
            int n = 10 - i;          // 계산의 결과를 저장
            fout.write(n);           // 파일에 결과값을 바이너리로 저장(바이트)
        }
        fout.close();                // 스트림을 닫음
        FileInputStream fin = new FileInputStream(path);
        int ch;
        while ((ch = fin.read()) != -1) {
            System.out.print(ch + " ");
        }
        fin.close();
    } catch (IOException e) {
        System.err.println(e.getMessage());
    }
}
```

Text 파일 라인수 세기

- Files
- BufferedReader
- LineNumberReader
- Scanner

Text 파일 라인수 세기

- Files
  - Files의 lines()와 Stream의 count() 메소드를 사용하여, File의 라인 수를 구할 수 있음

```
java.nio.file.Files
public static Stream<String> lines(Path path)
```

  - File에서 line을 읽어서 Stream으로 반환

```
java.util.stream.Stream<T>
long count()
```

  - Stream으로 들어온 element의 개수를 세서 반환

Text 파일 라인수 세기

```
public static void main(String[] args) {
	Path path = Paths.get("d:₩₩example₩₩text_file.txt");
	try {	// 파일 라인수 세기
		long lineCount = Files.lines(path).count();
		// 결과 출력
		System.out.println(lineCount);
	} catch (IOException e) {
		 e.printStackTrace();
	}
}
```

Text 파일 라인수 세기

```
public static void main(String[] args) {
    try {
        // BufferedReader 생성
        BufferedReader reader = new BufferedReader(
                        new FileReader("d:₩₩example₩₩text_file.txt"));
        // 라인수 세기
        int lineCount = 0;
        while (reader.readLine() != null) {
            lineCount++;
        }
        // 결과 출력
        System.out.println(lineCount);
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Text 파일 라인수 세기

- LineNumberReader
  - java.io.LineNumberReader 클래스는 줄 번호를 추적하는 buffered character-input stream 임

```
java.io.LineNumberReader
public int getLineNumber()
```

  - getLineNumber() 메소드를 이용하면, 현재 줄 번호를 가져올 수 있음

Text 파일 라인수 세기

```
public static void main(String[] args) {
    try {
      // LineNumberReader 생성
        LineNumberReader reader = new LineNumberReader(
                        new FileReader("d:₩₩example₩₩text_file.txt"));
      // 라인수 세기
        while (reader.readLine() != null)
                        ;
        int lineCount = reader.getLineNumber();
         // 결과 출력
        System.out.println(lineCount);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Text 파일 라인수 세기

```
public static void main(String[] args) {
    try {
     // Scanner 생성
        Scanner scanner = new Scanner(
                          new FileReader("d:₩₩example₩₩text_file.txt"));
      // 라인수 세기
        int lineCount = 0;
        while (scanner.hasNextLine()) {
            scanner.nextLine();
            lineCount++;
        }
      // 결과 출력
        System.out.println(lineCount);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

라인번호 추가하여 복사

- 프로그램 소스 파일을 읽어서 라인 번호를 붙여서 파일로 출력하는 프로그램을 만들어보자

라인번호 추가하여 복사(I)

```
public class Main {
    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        String source, destination;
        String path = ".₩₩data₩₩";
        File src, dst;
        while (true) {
            System.out.print("₩n 소스 파일 입력 : ");
            source = keyboard.nextLine( );
            src = new File(path, source);              // 소스 파일
            if (src.exists()) {
                break;
            } else {
                System.err.print("₩n 파일이 없습니다. 다시 입력해 주세요!₩n");
            }
        }
```

라인번호 추가하여 복사

```
System.out.print(" 목적 파일 입력 : ");
destination = keyboard.nextLine( );
dst = new File(path, destination);              // 목적 파일
String line;
String toline;
try {
    BufferedReader input = new BufferedReader(new FileReader(src));
    PrintWriter output = new PrintWriter(new FileWriter(dst));
    int lineNumber = 1;
    while ((line = input.readLine()) != null) {
        toline = String.format("line %03d : %s", lineNumber++, line);
        System.out.printf("%s₩n", toline);
        output.printf("%s%n", toline);
    }
```

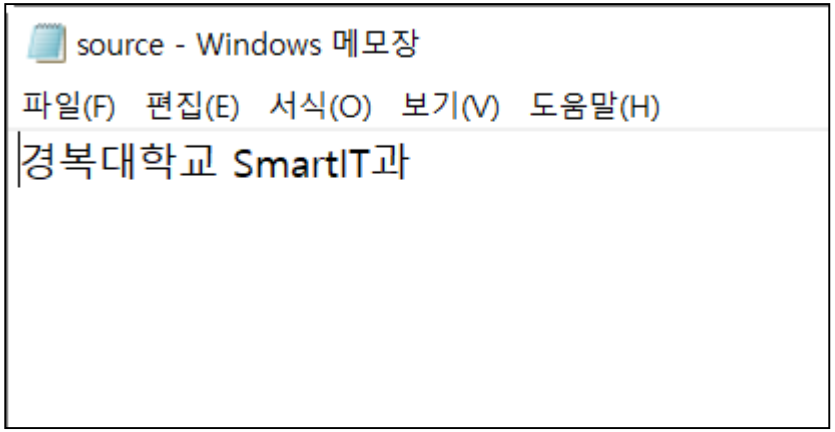
라인번호 추가하여 복사

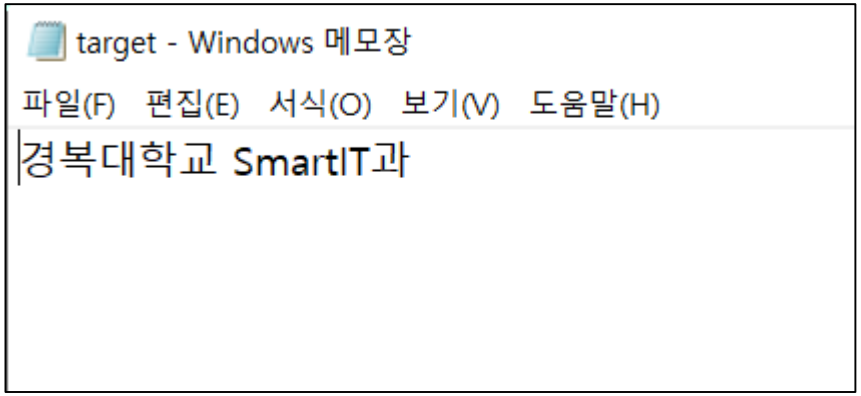
```
            input.close();
            output.close();
            Runtime.getRuntime().exec("notepad " + dst);
        } catch (IOException e) {
            System.out.println("파일의 오류");
        }
    }
}
```

File 복사

- data 디렉토리에 존재하는 source.txt 파일의 내용을 target.cpy로 복사하는 프로그램을 작성하시오

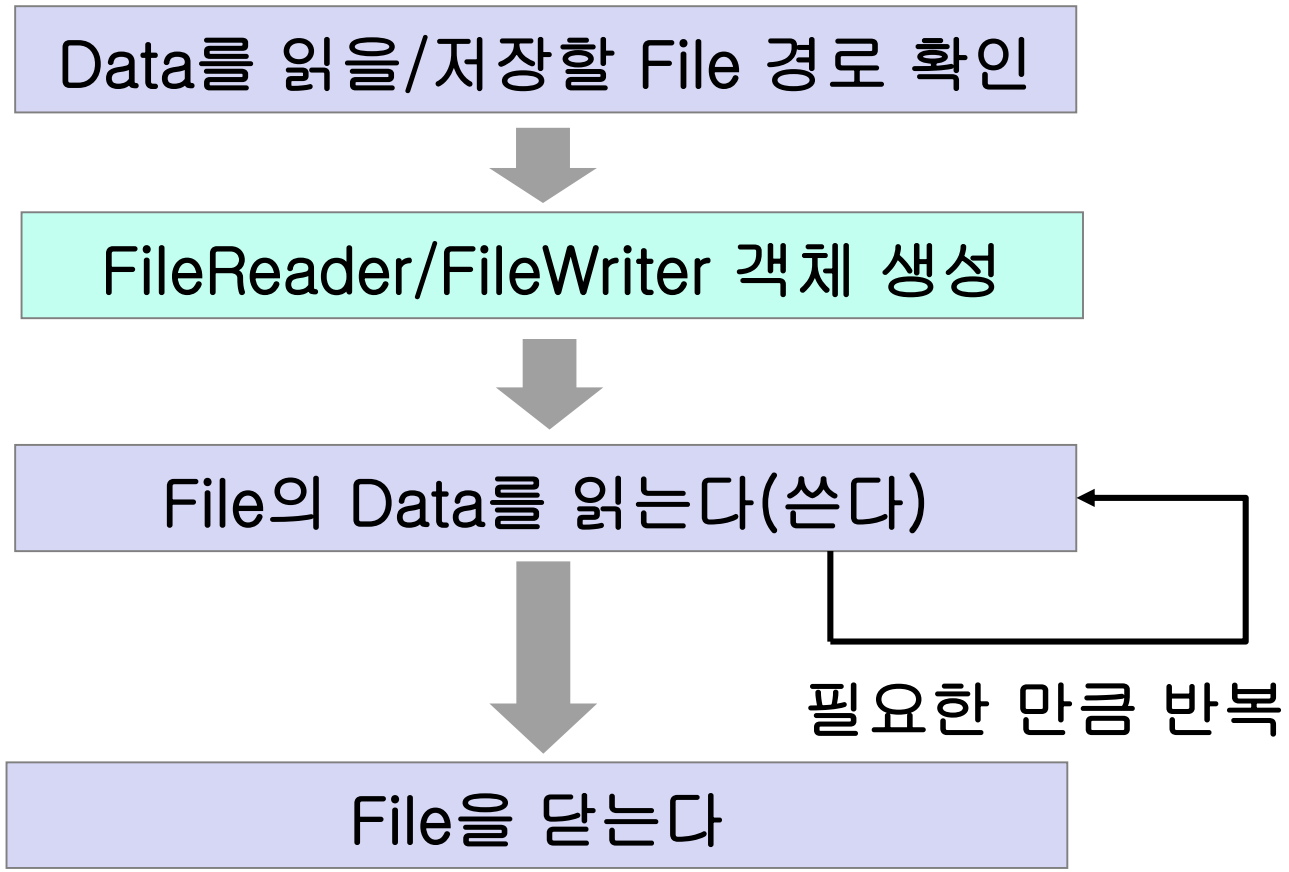
- 목적
  - 파일의 생성과 파일을 읽고 쓰는 방법의 이해





File 복사

- File로부터 Data를 읽는(쓰는) 3단계



File 복사

- File을 복사하는 방법
  - FileInputStream, FileOutputStream
  - BufferedInputStream, BufferedOutputStream
  - InputStreamReader, OutputStreamWriter
  - FileReader, FileWriter
  - BufferedReader, BufferedWriter
  - BufferedReader. PrintWriter
  - Scanner, PrintWriter
  - Files.copy()
  - FileChannel
  - Apache Commons IO (외부 Library)

File 복사(I)

```
public static void main(String[] args) throws IOException {
    final String inputfile = ".₩₩data₩₩source.txt";
    final String outputfile = ".₩₩data₩₩target.txt";

    File file = new File(inputfile);
    if (file.exists()){
        FileInputStream input = new FileInputStream(file);
        FileOutputStream output = new FileOutputStream(outputfile);
        int ch;
        int count = 0;
        while ((ch = input.read()) != EOF) {
            output.write(ch);
            count++;
        }
        input.close();
        output.close();
        System.out.printf("₩n %,d Bytes 파일 복사 완료₩n", count);
    } else {
        System.err.printf("₩n 입력파일 : %s가 없습니다₩n", inputfile);
    }
}
```

File 복사(I)

- ch = (char) input.read()
  - File(입력 스트림)으로부터 1 Byte를 읽고 4 Byte int 타입으로 반환
  - 반환된 4 Byte 중 1 Byte에만 Data가 들어 있음
  - 입력 Stream으로부터 더 이상 Byte를 읽을 수 없으면 EOF(-1)을 반환
  - 읽을 수 있는 마지막 Byte까지 루프를 돌며 한 Byte씩 읽을 수 있음
- output.write(ch)
  - 매개 변수로 주어진 int 값에서 끝에 있는 1 Byte만 출력 Stream으로 출력
  - 매개 변수가 정수 타입이지만 4 Byte를 모두 보내는 것은 아님

File 복사(I-1)

```
public static void main(String[] args) throws IOException {
    final String inputfile = ".₩₩data₩₩source.txt";
    final String outputfile = ".₩₩data₩₩target.txt";

    File file = new File(inputfile);
    if (file.exists()){
        FileInputStream input = new FileInputStream(file);
        FileOutputStream output = new FileOutputStream(outputfile);
        int count = 0;
        byte[] buffer = new byte[512];
        int data;
        while ((data = input.read(buffer)) != EOF) {
            output.write(buffer, 0, data);
            count += data;
        }
        input.close();
        output.close();
        System.out.printf("₩n %,d Bytes 파일 복사 완료₩n", count);
    } else {
        System.err.printf("₩n 입력파일 : %s가 없습니다₩n", inputfile);
    }
}
```

File 복사(I-1)

- data = input.read(buf);
  - buffer(버퍼)에 File의 데이터를 읽어서 넣고, buffer로 읽은 총 Byte 수를 반환
  - 파일 끝에 도달하여, 더 이상 읽을 데이터가 없을 경우 EOF(-1)을 반환
- output.write(buffer, 0, data);
  - buffer(버퍼)에 담긴 데이터를,
  - 0번째 offset부터,
  - data 길이 만큼,
  - output stream에 출력

- read() 메소드는 Byte 수만큼 반복해서 읽지만, read(byte[] b)는 한 번 읽을 때 주어진 Byte 배열의 길이 만큼 읽기 때문에 반복 횟수가 현저히 적어 효율적 임

File 복사(I-2)

```
public static void main(String[] args) {
    Scanner keyboard = new Scanner(System.in);
    System.out.print(" 소스 파일 이름 입력 : ");
    File inputfile = new File(".₩₩data₩₩" + keyboard.next());
    System.out.print(" 복사 파일 이름 입력 : ");
    File outputfile = new File(".₩₩data₩₩" + keyboard.next());
    if (inputfile.exists()) {
        try {
            FileInputStream input = new FileInputStream(inputfile);
            FileOutputStream output = new FileOutputStream(outputfile, true);
            byte[] buffer = new byte[input.available()];
            int data = input.read(buffer);
            output.write(buffer, 0, data);
            System.out.printf("₩n %d bytes 파일 복사 완료₩n", data);
            input.close();
            output.close();
            Runtime.getRuntime().exec("notepad.exe " + outputfile);
        } catch (IOException e) {
            System.err.println(e.getMessage());
        }
```

File 복사(I-2)

```
    } else {
        System.out.printf("입력 파일 %s가 없습니다.₩n", inputfile);
    }
}
```

- ✓ Java Runtime Class
  - ✓ JVM이 작동하는 System 운영체제와의 Interface로 작동하는 클래스
- ✓ 주로 운영체제 기반의 Program을 실행하거나 정보를 가져오는 기능을 사용
- ✓ System 침입의 주요 경로가 될 수 있으므로 실행 시 보안적 요소를 고려해야 함

File 복사(I - 3)

```
public static void main(String[] args) throws IOException {
    FileInputStream input;
    FileOutputStream output;
    Copy copy;
    if (args.length < 1) {
        System.out.println("CopyFile01-3 filename₩n파일명을 입력하세요..");
        System.exit(0);
    } else if (args.length == 1) {
        output = new FileOutputStream(args[0]);
        copy = new Copy(System.in, output);
        copy.DataCopy();
        System.out.println("키보드로 입력한 내용이  " +
                                            args[0] + "파일에 복사되었습니다.");
    } else if (args.length == 2) {
        input = new FileInputStream(args[0]);
        output = new FileOutputStream(args[1]);
        copy = new Copy(input, output);
        copy.DataCopy();
        System.out.println(args[0] + "파일이 " + args[1] + "파일에 복사되었습니다.");
    }
}
```

File 복사(I-3)

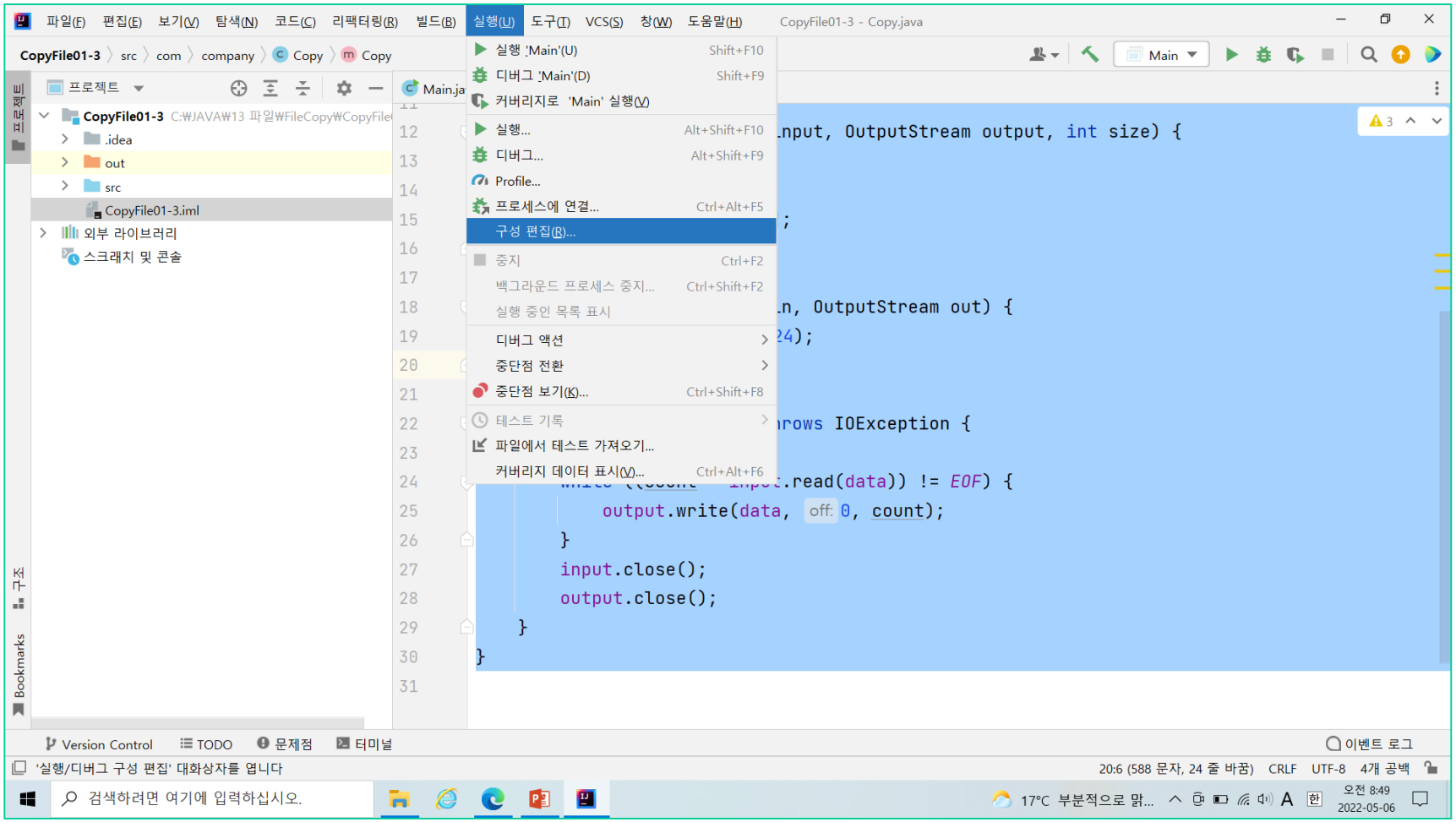
```
public class Copy {
    private InputStream input;
    private OutputStream output;
    private byte[] data;

    public Copy(InputStream input, OutputStream output, int size) {
        this.input = input;
        this.output = output;
        data = new byte[size];
    }
    public Copy(InputStream in, OutputStream out) {
        this(in, out, 1024);
    }
    public void DataCopy() throws IOException {
        int count;
        while ((count = input.read(data)) != EOF) {
            output.write(data, 0, count);
        }
        input.close();
        output.close();
    }
}
```

File 복사(I-3)

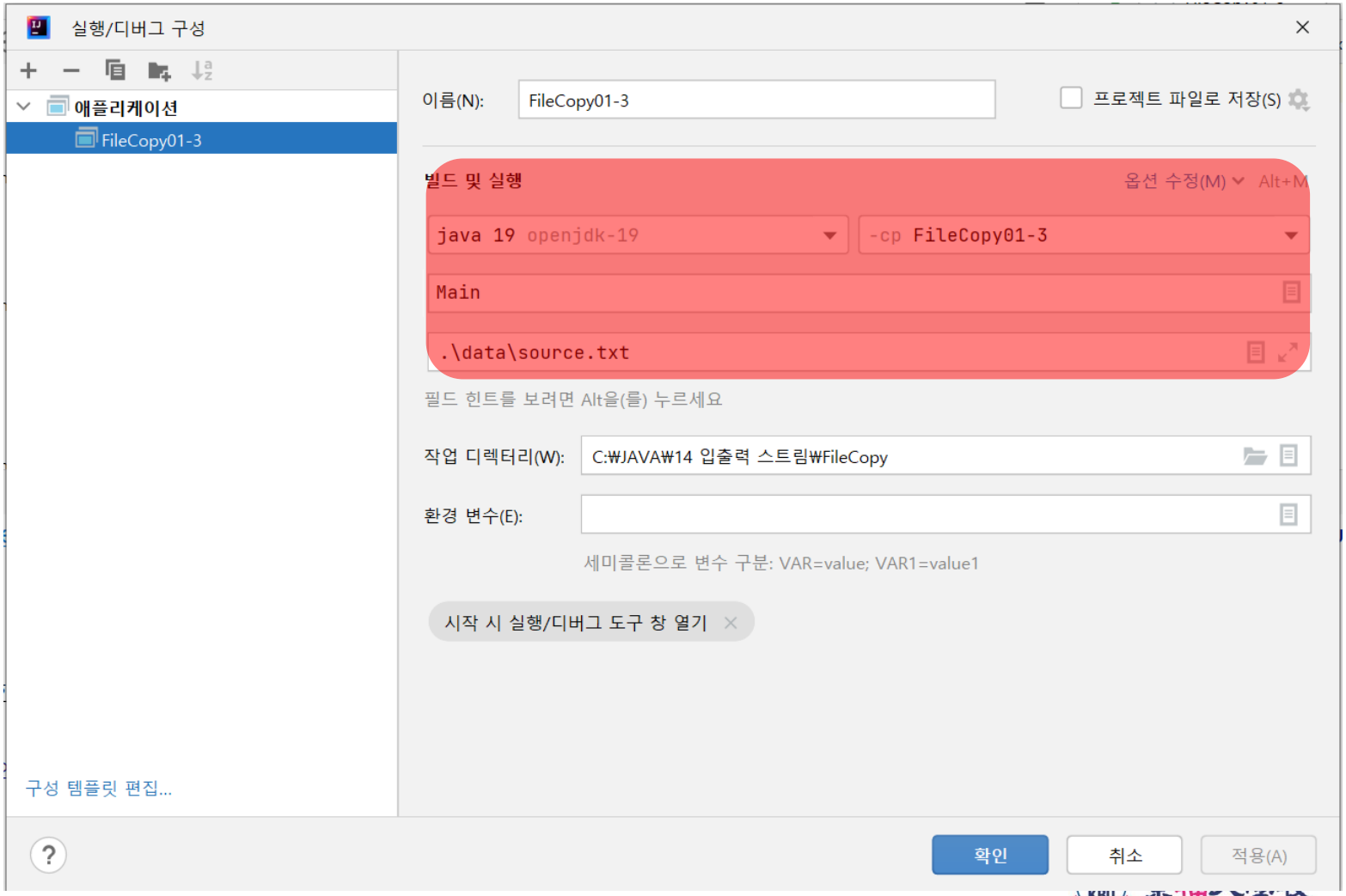
- Command Line Input (String[] args 용도)
  - "java 클래스"로 Program을 실행하면 JVM은 길이가 0인 String 배열을 먼저 생성하고, main( ) 메소드를 호출할 때 매개값으로 전달
  - public static void main(String args[]) {... 에서
  - main() 메소드의 기본으로 항상 있는 args[]라는 "문자열 배열"은 JAVA Program 실행 시에 주어진 "옵션"들이 자동으로 담겨지는 곳
  - Arguments 즉, "인수" = "매개 변수" = "파라미터" 라는 뜻

File 복사(I-3)

실행 – 구성 편집



File 복사(I-3)


실행/디버그 구성
애플리케이션
FileCopy01-3
구성 템플릿 편집...
이름(N):
FileCopy01-3
프로젝트 파일로 저장(S)
빌드 및 실행
옵션 수정(M) Alt+M
java 19 openjdk-19
-cp FileCopy01-3
Main
.\data\source.txt
필드 힌트를 보려면 Alt을(를) 누르세요
작업 디렉터리(W):
C:₩JAVA₩14 입출력 스트림₩FileCopy
환경 변수(E):
세미콜론으로 변수 구분: VAR=value; VAR1=value1
시작 시 실행/디버그 도구 창 열기
확인
취소
적용(A)

파일 복사(II)

```
public static void main(String[] args) throws IOException {
    Scanner keyboard = new Scanner(System.in);
    final String path = ".₩₩data₩₩";
    final String inputfile = path + "source.txt";
    System.out.print("복사 파일 이름 : ");
    String filename = keyboard.next();
    String outputfile = path + filename;

    File file = new File(inputfile);
    if (file.exists()){
        BufferedInputStream input = new BufferedInputStream(
                                                    new FileInputStream(file));
        BufferedOutputStream output = new BufferedOutputStream(
                                                    new FileOutputStream(outputfile));
        byte[] buffer = new byte[512];
        int data;
        int count = 0;
```

파일 복사(II)

```
        while ((data = input.read(buffer)) != EOF) {
            count += data;
            output.write(buffer, 0, data);
        }
        input.close();
        output.close();
        System.out.printf("₩n %,d bytes 파일 복사 완료₩n", count);
    } else {
        System.err.printf("₩n 입력파일 : %s가 없습니다₩n", inputfile);
    }
}
```

파일 복사(II-1)

- 수정 내용

```
 ..........................................
int ch;
int count = 0;
while ((ch = input.read()) != EOF) {
    count++;
    output.write(ch);
}
 ..........................................
```

File 복사 예제(III)

```
public static void main(String[] args) throws IOException {
    final String inputfile = ".₩₩data₩₩source.txt";
    final String outputfile = ".₩₩data₩₩target.txt";
    File file = new File(inputfile);
    if (file.exists()){
        InputStreamReader input = new InputStreamReader(
                                                    new FileInputStream(file));
        OutputStreamWriter output = new OutputStreamWriter(
                                                new FileOutputStream(outputfile));
        int ch;
        int count = 0;
        while ((ch = input.read()) != EOF) {
            count++;
            output.write(ch);
        }
        input.close();
        output.close();
        System.out.printf("₩n %,d Bytes 복사 완료₩n", count);
    } else {
        System.err.printf("₩n 입력파일 : %s가 없습니다₩n", inputfile);
    }
}
```

File 복사 예제(IV)

```
public static void main(String[] args) throws IOException {
    final String inputfile = ".₩₩data₩₩source.txt";
    final String outputfile = ".₩₩data₩₩target.txt";
    File file = new File(inputfile);
    if (file.exists()){
        FileReader input = new FileReader(file);
        FileWriter output = new FileWriter(outputfile);
        int ch;
        int count = 0;
        while ((ch = input.read()) != EOF) {
            count++;
            output.write(ch);
        }
        input.close();
        output.close();
        System.out.printf("₩n %,d Bytes 복사 완료₩n", count);
    } else {
        System.err.printf("₩n 입력파일 : %s가 없습니다₩n", inputfile);
    }
}
```

File 복사 예제(IV)

- ch = read()
  - Text File에서 한 글자씩 읽어서, 하나의 char를 반환
  - 더 이상 읽을 글자가 없으면, EOF(-1)을 반환
- write(int a)
  - int a로 지정된 단일 문자를 출력
- write(String str, int pos, int length)
  - 위치 pos에서 문자 length 수까지 문자열의 일부를 출력
- write(char ch[], int pos, int length)
  - 배열 ch[]의 문자 위치를 pos에서 length 길이 수까지 출력
- write(char ch[])
  - ch[]로 지정된 문자 배열을 출력
- write(String st)
  - 'st'로 지정된 문자열 값을 파일에 출력

File 복사 예제(V)

```
public static void main(String[] args) throws IOException {
    final String inputfile = ".₩₩data₩₩source.txt";
    final String outputfile = ".₩₩data₩₩target.txt";
    File file = new File(inputfile);
    if (file.exists()){
        BufferedReader input = new BufferedReader(new FileReader(file));
        BufferedWriter output = new BufferedWriter(new FileWriter(outputfile));
        String line;
        while ((line = input.readLine()) != null) {
            output.append(line + "₩n");
        }
        input.close();
        output.close();
        System.out.println("₩n 파일 복사 완료");
    } else {
        System.err.printf("₩n 입력파일 : %s가 없습니다₩n", inputfile);
    }
}
```

File 복사 예제(V-1)

```
public static void main(String[] args) throws IOException {
    final String inputfile = ".₩₩data₩₩source.txt";
    final String outputfile = ".₩₩data₩₩target.txt";
    File file = new File(inputfile);
    if (file.exists()){
        BufferedReader input = new BufferedReader(new FileReader(file));
        BufferedWriter output = new BufferedWriter(new FileWriter(outputfile));
        String line;
        StringBuilder builder = new StringBuilder();
        while ((line = input.readLine()) != null) {
            builder.append(line + "₩n");
        }
        output.write(builder.toString());
        input.close();
        output.close();
        System.out.printf("₩n %,d Bytes 복사 완료", builder.length());
    } else {
        System.err.printf("₩n 입력파일 : %s가 없습니다", inputfile);
    }
}
```

File 복사 예제(VI)

```
public static void main(String[] args) throws IOException {
    final String inputfile = ".₩₩data₩₩source.txt";
    final String outputfile = ".₩₩data₩₩output1.txt";
    File file = new File(inputfile);
    if (file.exists()){
        BufferedReader input = new BufferedReader(new FileReader(inputfile));
        PrintWriter output = new PrintWriter(new FileWriter(outputfile));
        String line;
        while ((line = input.readLine()) != null) {
            System.out.println(line);
            output.println(line);
        }
        System.out.println("라인 단위로 복사 완료");
        input.close();
        output.close();
    } else {
        System.err.printf("₩n 입력파일 : %s가 없습니다", inputfile);
    }
}
```

File 복사 예제(VII)

```
public static void main(String[] args) throws IOException {
    final String inputfile = ".₩₩data₩₩source.txt";
    final String outputfile = ".₩₩data₩₩output21.txt";
    File file = new File(inputfile);
    if (file.exists()){
        Scanner input = new Scanner(file);
        PrintWriter output = new PrintWriter(new FileWriter(outputfile));
        while (input.hasNextLine()) {
            String line = input.nextLine();
            System.out.println(line);
            output.println(line);
        }
        System.out.println("라인 단위로 복사 완료");
        input.close();
        output.close();
    } else {
        System.err.printf("₩n 입력파일 : %s가 없습니다", inputfile);
    }
}
```

File 복사 예제(VIII)

```
public static void main(String[] args) throws IOException {
    String inputfile = ".₩₩data₩₩source.txt";
    String outputfile = ".₩₩data₩₩target11.txt";

    Path source = Paths.get(inputfile);
    if (Files.exists(source)) {
        Path target = Paths.get(outputfile);

        Files.copy(source, target, StandardCopyOption.REPLACE_EXISTING);
    } else {
        System.err.printf("₩n 입력파일 : %s가 없습니다", inputfile);
    }
}
```

File 복사 예제(VIII)

- Files.copy(source, traget,
  StandardCopyOption.REPLACE_EXISTING);
  - source.txt 파일을 traget.txt로 복사
  - REPLACE_EXISTING 옵션을 주었기 때문에, 기존에 traget.txt 파일이 존재하더라도 그대로 덮어 씀

File 복사 예제(IX)

```
public static void main(String[] args) throws IOException {
    String inputfile = ".₩₩data₩₩source.txt";
    String outputfile = ".₩₩data₩₩target.txt";
    File file = new File(inputfile);
    if (file.exists()) {
        RandomAccessFile input = new RandomAccessFile(file, "r");
        RandomAccessFile output = new RandomAccessFile(outputfile, "rw");

        FileChannel source = input.getChannel();
        FileChannel target = output.getChannel();
        source.transferTo(0, source.size(), target);
        // target.transferFrom(source, 0, source.size());
    } else {
        System.err.printf("₩n 입력파일 : %s가 없습니다", inputfile);
    }
}
```

File 복사 예제(IX)

- source.transferTo(0, source.size(), target)
- target.transferFrom(source, 0, source.size())
  - FileChannel 클래스의 transferTo() 또는 transferFrom() 메소드를 이용하여 데이터를 복사할 수 있음
  - 이 메소드는 데이터를 source.size() 만큼 target에 써주기 때문에, 만약, target 파일이 이미 존재하고, target 파일의 데이터가 더 길면, 나머지 부분은 그대로 남아있게 됨

File 복사 예제(X)

```
public static void main(String[] args) throws IOException {
    String inputfile = ".₩₩data₩₩source.txt";
    String outputfile = ".₩₩data₩₩target.txt";

    File input = new File(inputfile);
    if (input.exists()) {
        File output = new File(outputfile);

        FileUtils.copyFile(input, output);
    } else {
        System.err.printf("₩n 입력파일 : %s가 없습니다₩n", inputfile);
    }
}
```

File 복사 예제(X)

- Apache Commons IO Library 사용
  - FileUtils 클래스는 copyFile(file, newFile)이라는 메소드 이용

File 복사 예제(X)

- Apache Commons IO 검색 – jar 파일 다운로드

File 복사 예제(X)

- 프로젝트 구조에서 library 추가