Buffer 입출력 Stream

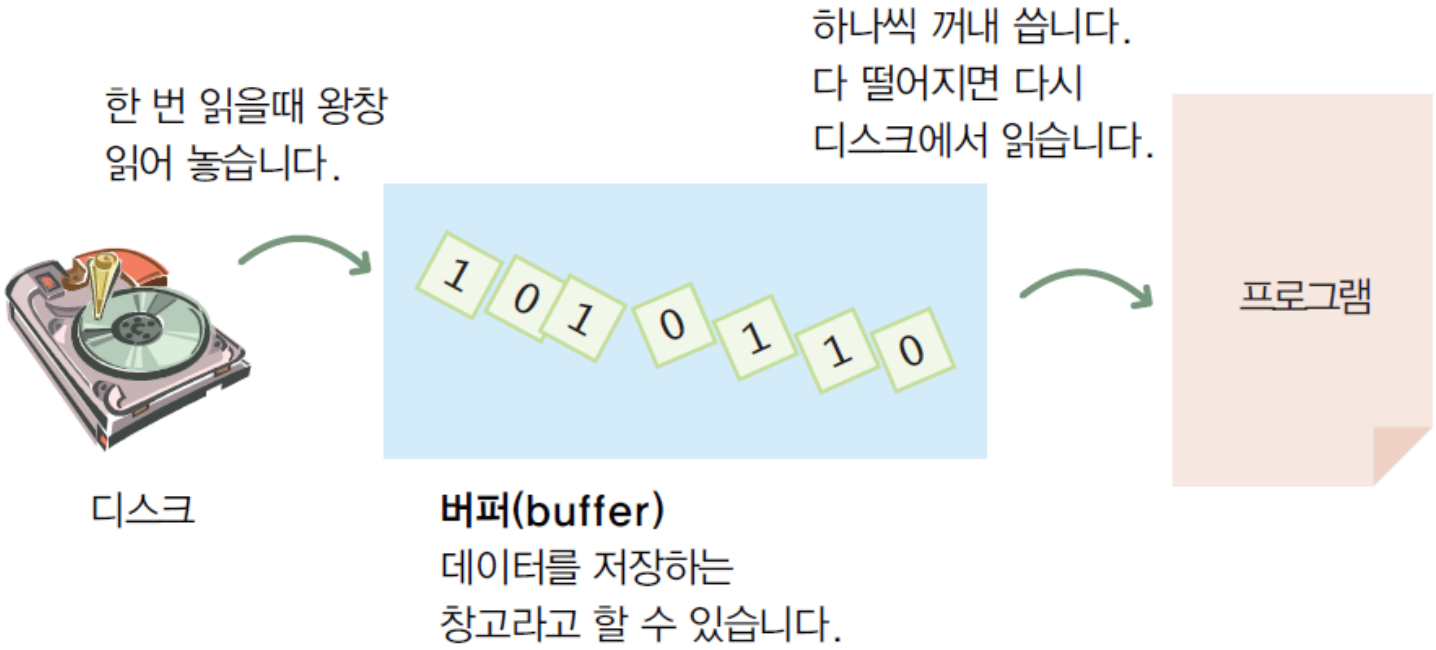
경복대학교
소프트웨어융합과
배희호 교수
010-2369-4112
031-570-9600
hhbae@kbu.ac.kr

Buffer

- Buffer
  - Buffer란 임시 저장 공간을 의미
  - 임시 저장 공간이라고 해서 엉뚱하게 보일 수 있지만 정확히 말하면 A와 B가 서로 입출력을 수행하는 데에 있어서 속도 차이를 극복하기 위해 사용하는 임시 저장 공간을 의미

Buffer Stream

- Buffer Stream 개념
  - Buffer를 가진 Stream
  - JAVA에서는 Buffering된 Stream을 제공
  - 입출력 Data를 일시적으로 저장하는 Buffer를 이용하여 입출력 효율 개선



Buffer Stream

- Buffer 입출력의 목적
  - HDD로부터 Data를 읽는 작업은 많은 시간을 소모하기 때문에 가능한 HDD의 접속 횟수를 줄이는 것이 좋음
  - Buffer는 Memory에 생성되는 임시 기억 장소로써, HDD에 한번 접속할 때 읽어 Memory에 올려놓고 처리하면 Data 처리 속도가 개선
  - 입출력 시 운영체제의 API 호출 횟수를 줄여 입출력 성능 개선
    - 출력 시 여러 번 출력되는 Data를 Buffer에 모아두고 한 번에 장치로 출력
    - 입력 시 입력 Data를 Buffer에 모아두고 한번에 Program에게 전달
  - 문자 하나하나를 처리하기엔 버겁기 때문에(Overhead) Buffer에 만들어진 일정 공간에 담아서 사용하는 것이 효율적 임

Buffer Stream

버퍼 기본 크기 :
8192 char(즉, 16,384 byte)

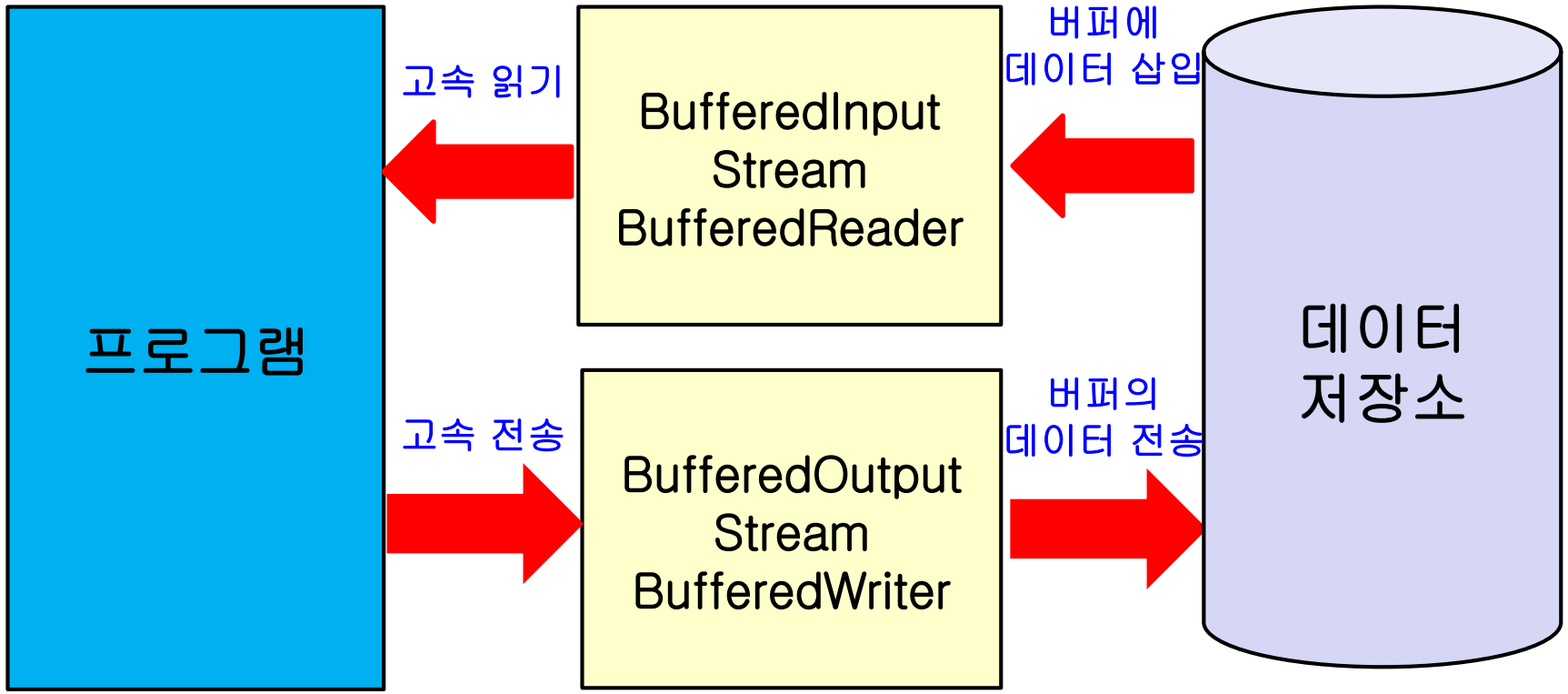
- Buffer Stream 종류

클래스 이름	설명
BufferedInputStream	바이트 입력 스트림을 버퍼링하는 클래스
BufferedOutputStream	바이트 출력 스트림을 버퍼링하는 클래스
BufferedReader	문자 입력 스트림을 버퍼링하는 클래스
BufferedWriter	문자 출력 스트림을 버퍼링하는 클래스

  - Byte Buffer Stream
    - Byte 단위의 Binary Data를 처리하는 Buffer Stream
    - BufferedInputStream와 BufferedOutputStream
  - Character Buffer Stream
    - UNICODE의 문자 Data만 처리하는 Buffer Stream
    - BufferedReader와 BufferedWriter

Buffer Stream

- Buffer 역할



입출력 기능을 향상시키는 클래스들

- 이 클래스들은 모두 java.io 패키지에 속함
- 이 클래스들은 단독으로는 사용될 수 없음

클래스	설명
BufferedReader	데이터를 한꺼번에 읽어서 버퍼에 저장해두는 클래스
BufferedInputStream	
BufferedWriter	데이터를 버퍼에 저장해두었다가 한꺼번에 출력하는 클래스
BufferedOutputStream	
LineNumberReader	텍스트 파일의 각 행에 번호를 붙여가면서 읽는 클래스

BufferedInputStream 클래스

- Byte 단위로 File을 읽어 올 때 사용하는 Buffer Stream
- BufferedInputStream은 File을 읽어올 때 8192 Byte의 Buffer를 두고 작업을 하기 때문에 속도가 굉장히 빨라 짐

BufferedInputStream 클래스

- 생성자

생성자	설명
BufferedInputStream(InputStream in)	InputStream에 대한 BufferedInputStream 객체를 생성
BufferedInputStream (InputStream in, int size)	InputStream에 대한 BufferedInputStream 객체를 생성하고 내부 버퍼의 크기를 Size 값으로 설정

- BufferedInputStream 클래스에 추가되는 메소드는 없으며, 기본 InputStream 클래스에 정의된 메소드를 사용

BufferedInputStream 클래스

- Byte 입력 Stream의 성능을 향상시키기 위하여 BufferedInputStream 클래스를 사용하는 방법
  - 객체를 생성

```
FileInputStream in = new FileInputStream("input.dat");
```

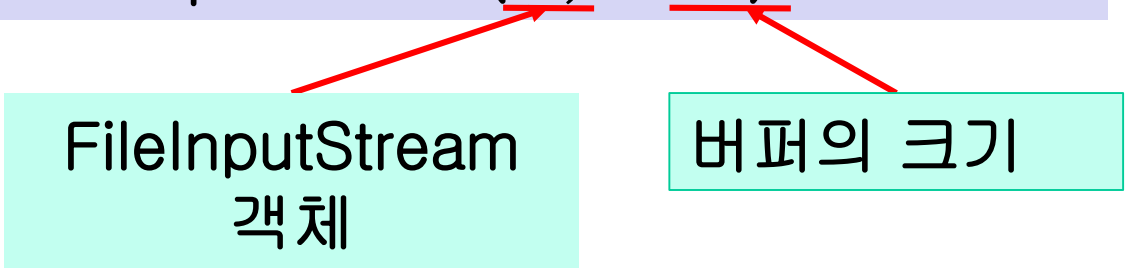
FileInputStream 객체를 생성해서 BufferedInputStream 생성자의 파라미터로 사용

```
BufferedInputStream in2 =
                        new BufferedInputStream(in);
```

BufferedInputStream 클래스

- Buffer 크기를 지정하는 방법

```
FileInputStream in = new FileInputStream("input.dat");

BufferedInputStream input =
                new BufferedInputStream(in, 1024);
```



BufferedInputStream 클래스

- read( ) 메소드를 호출하여 Data를 읽음
  - FileInputStream 클래스의 read( ) 메소드 호출 방법과 동일
- readLine( ) 메소드 호출
  - 문자열을 입력
  - 입력이 끝인 경우는 null을 반환
- File을 닫음
  - FileInputStream 클래스의 close( ) 메소드 호출 방법과 동일

BufferedOutputStream 클래스

- Program의 성능은 입출력이 가장 늦은 장치를 따라가게 됨
- 예) CPU와 Memory의 성능이 아무리 빠르다고 해도 HDD의 입출력 늦다면 Program의 실행 성능은 HDD의 처리 속도에 따라 맞춰 결정됨
- Network도 느린 Network 환경이라면 Computer의 성능이 좋더라도 Messenger 또는 Game의 속도가 느려 짐
- Program이 입출력 소스와 직접 작업하는 대신에 중간에 Memory Buffer와 작업함으로써 실행 성능을 어느 정도 향상 시킬 수는 있음
- BufferedInputStream 과 BufferedOutputStream은 Byte 기반의 성능 향상 보조 Stream
- BufferedInputStream은 파일을 읽어올 때 8192 Byte의 버퍼를 두고 작업을 하기 때문에 속도가 굉장히 빨라 짐

BufferedOutputStream 클래스

- BufferedOutputStream 클래스는 직접적으로 출력 Stream에 쓰지 않음
- 이 클래스는 더 나은 성능으로 Stream에 쓰기 위해 Buffer에 먼저 쓰게 됨
- Buffer의 크기는 생성자에 넘겨 줄 수 있지만, 아무 값도 넘기지 않으면 기본값(8192)으로 설정
- close() 메소드를 호출하면 Buffer의 내용을 Stream에 씀
  - 이것을 확인 하기 위해 close() 메소드를 호출 하지 않으면 File에는 아무것도 쓰지 않게 됨
- 만약 Stream을 닫지 않고 Buffer의 내용을 File에 쓰고 싶을 때에는 flush() 메소드를 사용
- 만약 close() 메소드를 호출 한 뒤 write()나 flush() 메소드를 호출하게 되면 IOException이 발생

BufferedOutputStream 클래스

- 생성자

생성자	설명
BufferedOutputStream (OutputStream out)	OutputStream에 대한 BufferedOutputStream 객체를 생성
BufferedOutputStream (OutputSteam out, int size)	OutputStream에 대한 BufferedOutputStream 객체를 생성하고 내부 버퍼의 크기를 Size 값으로 설정

- BufferedOutputStream 클래스에 추가되는 메소드는 없으며, 기본 OutputStream 클래스에 정의된 메소드를 사용

BufferedOutputStream 클래스

- BufferedInputStream 클래스는 Instance가 만들어질 때 내부에 Buffer 배열을 만듦
- Stream의 Byte가 입력되면 Stream으로 부터 필요에 따라 한번에 다수의 Byte가 내부 Buffer에 추가 됨
- mark() 메소드는 입력 Stream의 특정 위치를 기억함
- reset() 메소드는 입력 Stream의 마지막 mark된 위치로 재설정 됨

BufferedOutputStream 예제 1

```
public static void main(String[] args) {
    String path = ".₩₩data₩₩test.txt";
    String message = " 파일에 저장될 문자열 입니다.₩n Hellow world !!";

    try {
        FileOutputStream outputStream = new FileOutputStream(path);
        BufferedOutputStream output = new BufferedOutputStream(outputStream);
        output.write(message.getBytes());
        output.close();
        outputStream.close();
    } catch (IOException e) {
        System.err.println(e.getMessage());
    }
```

✓ BufferedOutputStream 클래스의 모든 메소드는 IOException이 발생 할 수 있기 때문에 BufferedOutputStream 클래스의 메소드들을 이용할 때에는 try-catch 를 사용해야 함

BufferedOutputStream 예제 1

```
File file = new File(path);
if (file.exists()) {
    try {
        FileInputStream inputStream = new FileInputStream(file);
        BufferedInputStream input = new BufferedInputStream(inputStream);
        StringBuilder builder = new StringBuilder();
        while (input.available() > 0) {
            builder.append((char) input.read());
        }
        System.out.println("내용 : " + builder);
        input.close();
        inputStream.close();
    } catch (IOException e) {
        System.err.println(e.getMessage());
    }
} else {
    System.err.println("입력 파일이 없습니다");
}
```

✓ 한글 입력이 안됨

BufferedInputStream 예제 1

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

    File file = new File(path);
    if (file.exists()) {
        try {
            InputStream inputStream = new FileInputStream(file);
            BufferedInputStream input = new BufferedInputStream(inputStream);
            int size = inputStream.available();
            byte[] buffer = new byte[size];
            input.read(buffer);
            System.out.println(new String(buffer, StandardCharsets.UTF_8));
            input.close();
        } catch (IOException e) {
            System.err.println(e.getMessage());
        }
    } else {
        System.err.println("입력 파일 존재하지 않음");
    }
}
```

BufferedInputStream/BufferedOutputStream 예제 1

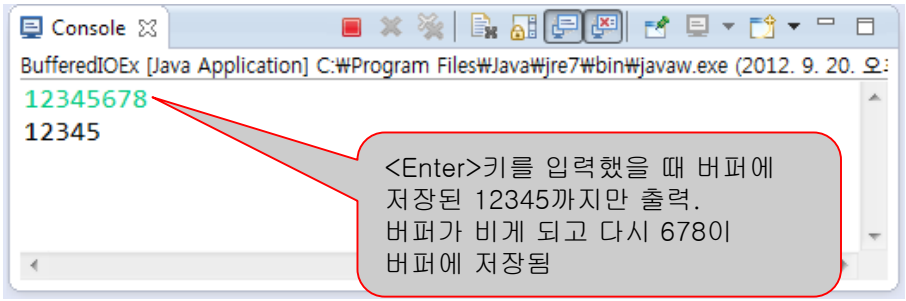
```
public static void main(String[] args) {
    Scanner keyboard = new Scanner(System.in);
    String path = ".₩₩data₩₩";

    String inputname = path + "test.txt";
    System.out.print("복사 파일 이름 : ");
    String filename = keyboard.next();
    String outputname = path + filename;

    File file = new File(inputname);
    if (file.exists()) {
        try {
            InputStream input = new BufferedInputStream(new FileInputStream(file));
            OutputStream output = new BufferedOutputStream(
                                        new FileOutputStream(outputname));
```

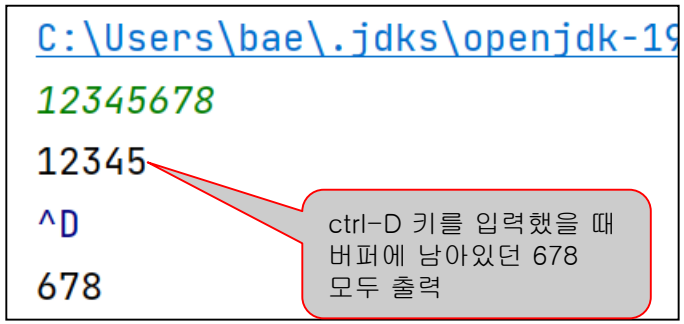
BufferedInputStream/BufferedOutputStream 예제 1

```
            byte ch;
            int count = 0;
            while ((ch = (byte) input.read()) != EOF) {
                count++;
                output.write(ch);
            }
            output.flush();
            input.close();
            output.close();
            System.out.printf("₩n %,d byte 파일 복사 완료₩n", count);
        } catch (IOException e) {
            System.err.println(e.getMessage());
        }
    } else {
        System.err.printf("₩n 입력파일 : %s가 없습니다₩n", inputname);
    }
}
```

BufferedInputStream/BufferedOutputStream 예제 2

- Buffer 크기를 5로 하고, 표준 출력 Stream과 연결된 Buffer 출력 Stream을 생성하라.그리고 Keyboard에서 입력 받은 문자를 출력 Stream에 출력하고, 입력의 끝을 알리면(<Ctrl>-<D>) Buffer에 남아 있는 모든 문자를 출력하는 프로그램을 작성하라.





BufferedInputStream/BufferedOutputStream 예제 2

```
public static void main(String[] args) {
    BufferedInputStream in = new BufferedInputStream(System.in);
    BufferedOutputStream out = new BufferedOutputStream(System.out, 5);
    try {
        int ch;
        while ((ch = in.read()) != -1) {
            out.write(ch);
        }
        out.flush();
        in.close();
        out.close();
    } catch (IOException e) {
        System.err.println(e.getMessage());
    }
}
```

BufferedReader 클래스

- Reader 클래스의 하위 클래스
- Character Stream의 입력 시 Buffer를 사용
- InputStreamReader에 Buffering 기능이 추가된 Class
- 사용자가 요청할 때마다 Data를 읽어 오는 것이 아니라 일정한 크기의 Data를 한번에 읽어 Buffer에 보관 한 후, 사용자의 요청이 있을 때 Buffer에서 Data를 읽어오는 방식으로 동작
- 속도가 향상되고 시간 부하가 적다는 장점
- BufferedReader는 입력을 Line 단위로 입력 받기 때문에 String으로 받아들이므로 다른 Type의 Data라면 이후 형 변환이 필요함

BufferedReader 클래스

- 생성자

생성자	내용
BufferedReader(Reader in)	in을 연결하는 디폴트 크기의 문자 입력 버퍼 스트림 생성
BufferedReader(Reader in, int sz)	in을 연결하는 sz 크기의 문자 입력 버퍼 스트림 생성

BufferedReader 클래스

- 메소드

메소드	설명
int read()	스트림으로부터 한 문자를 읽어서 int형으로 반환
int read(char[] buf)	문자배열 buf의 크기만큼 문자를 읽어들임. 읽어들인 문자수를 반환
int read(char[] buf, int offset, int length)	buf의 offset위치에서부터 length길이만큼 문자를 스트림으로부터 읽어 들임
String readLine()	스트림으로부터 한 줄을 읽어 문자열로 반환
void mark()	현재위치를 마킹, 차후 reset()을 이용하여 마킹위치부터 시작
void reset()	마킹이 있으면 그 위치에서부터 다시 시작, 그렇지 않으면 처음부터 다시 시작
long skip(int n)	n개의 문자를 건너 뜀
void close()	스트림 닫음

BufferedWriter 클래스

- Writer 클래스의 하위 클래스
- Character Stream의 출력 시 Buffer를 사용
- BufferedWriter 역시 File 쓰기를 담당
- FileWriter 객체를 인자로 BufferedWriter 객체를 생성하고, 그 사용법은 FileWriter와 동일
- 이름에서 느껴지는 것처럼 BufferedWriter는 Buffer를 갖고 있음
- BufferedWriter가 다음 두 가지 조건에서 FileWriter보다 효과적
  - Buffer보다 작은 크기의 쓰기일 경우
  - 한 곳이 아닌 여러 곳에서 쓰기가 이루어 지는 경우
- 항상 BufferedWriter가 효과적인 것은 아님

BufferedWriter 클래스

- 생성자

생성자	내용
BufferedWriter(Writer out)	out을 연결하는 디폴트 크기의 문자 출력 버퍼 스트림 생성 out : Writer 타입의 객체로서 출력 스트림을 의미
BufferedWriter(Writer out, int sz)	out을 연결하는 sz 크기의 문자 출력 버퍼 스트림 생성 sz : 버퍼의 크기를 지정. 지정하지 않으면 묵시적인 크기의 버퍼 사용

BufferedWriter 클래스

- 메소드

메소드	기능
void close()	스트림 닫음
void write(int c)	int형으로 문자 데이터를 출력문자스트림으로 출력
void write(char[] buf, int offset, int length)	문자배열 buf의 offset 위치부터 length 길이만큼을 출력스트림으로 출력
void newLine()	줄 바꿈 문자열 출력
void flush()	남아있는 데이터를 모두 출력

BufferedWriter 클래스

- write() 메소드를 사용하여 출력할 내용을 담고, flush()을 통해서 Buffer를 비워내어야 함
- 주의할 점은 write()만 사용한다고 출력이 되는 것은 아니고, 반드시 flush()을 써 주어야 함
- 그리고 출력이 끝났으면, close()을 통해서 스트림을 닫음
- BufferedWriter도 BufferedReader와 마찬가지로 IOException 예외 처리를 반드시 해 주어야 함

BufferedWriter 클래스

- BufferedWriter 클래스
  - 문자 출력 Stream의 성능을 향상시키는 클래스
  - 다음과 같은 방법으로 객체를 생성

```
FileWriter writer1 = new FileWriter("output.txt");

BufferedWriter writer2 = new BufferedWriter(writer1);
```

    - FileWriter 객체를 생성해서 BufferedWriter 생성자의 Parameter로 사용
  - write() 메소드를 호출하여 Data를 출력
    - FileWriter 클래스의 write() 메소드 호출 방법과 동일
  - File을 닫음
    - FileWriter 클래스의 close() 메소드 호출 방법과 동일

BufferedWriter 클래스

- 버퍼에 있는 데이터를 파일에 즉시 출력하는 flush 메소드

```
writer.flush();
```

↑ 호출되는 즉시 버퍼의 데이터를 모두 출력하는 메소드

BufferedWriter 예제 1

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

    try {
        Writer writer = new FileWriter(path);
        BufferedWriter output = new BufferedWriter(writer);
        output.write("Hello World!₩n");
        output.newLine(); //개행 즉 엔터 역할
        output.write("경복대학교에서 공부합니다₩n"); //개행과 함께 출력
        output.flush();
        output.close();
    } catch (IOException e) {
        System.err.println(e.getMessage());
    }
}
```

BufferedWriter 예제 1

```
    File file = new File(path);
    if (file.exists()) {
        try {
            Reader reader = new FileReader(file);
            BufferedReader input = new BufferedReader(reader);
            String line;
            while((line = input.readLine()) != null) {
                System.out.println(line);
            }
            input.close();
        } catch (IOException e) {
            System.err.println(e.getMessage());
        }
    } else {
        System.err.println("입력 파일이 존재하지 않아요");
    }
}
```

BufferedWriter 예제 2

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

    try {
        FileWriter outputStream = new FileWriter(path);
        BufferedWriter writer = new BufferedWriter(outputStream);
        writer.write(intxt);
        writer.append(source);
        writer.flush();
        System.out.println("파일 생성 성공");
        writer.close();
    } catch (IOException e) {
        System.err.println(e.getMessage());
    }
}
```

BufferedWriter 예제 3

```
public static void main(String[] args) {
    BufferedWriter outputFile;
    try {
        outputFile = new BufferedWriter(new PrintWriter(System.out));
        outputFile.write("경복대학교");
        outputFile.close();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}
```

BufferedReader 예제 1

```
public static void main(String[] args) throws IOException {
    InputStreamReader keyboard = new InputStreamReader(System.in);
    BufferedReader reader = new BufferedReader(keyboard);
    String address, temp;
    char gender;

    System.out.printf("주소 입력 : ");
    address = reader.readLine( );
    System.out.printf("정수 입력 : ");
    temp = reader.readLine();
    System.out.print("성별(M 또는 F) ? ");
    gender = (char) reader.read( );

    System.out.printf("나의 집 주소 : %s 입니다₩n", address);
    System.out.printf("성별: (아스키코드) : %d, ", (int) gender);
    System.out.printf("(문자) : %c₩n", gender);
    System.out.println(" value : " + Integer.parseInt(temp));
    reader.close();
    keyboard.close();
}
```

BufferedReader 예제 2

```
public static void main(String[] args) {
    String path = ".//data//test.txt";
    File file = new File(path);
    if (file.exists()) {
        try {
            BufferedReader reader = new BufferedReader(new FileReader(file));
            String line;
            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }
            reader.close();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

BufferedReader 예제 3

- Member.java

```
public class Member {
    private String name;
    private String location;
    private String mail;
    private int age;

    public Member(String name, String location, String mail, int age) {
        this.name = name;
        this.location = location;
        this.mail = mail;
        this.age = age;
    }

    public String getName() {
        return name;
    }
```

BufferedReader 예제 3

- Member.java

```
    public String getLocation() {
        return location;
    }

    public String getMail() {
        return mail;
    }

    public int getAge() {
        return age;
    }

    @Override
    public String toString() {
        return "name='" + name + '₩'' + ", location='" + location + '₩'' +
                ", mail='" + mail + '₩'' + ", age=" + age;
    }
}
```

BufferedReader 예제 3

- member.txt 저장

```
public static void main(String[] args) {
    String path = ".₩₩data₩₩member.txt";
    Member[] persons = {new Member("홍길동","서울","hong@test.com",30),
            new Member("이길남", "부산", "kil@homg.com", 25)};
    try {
        FileOutputStream outputStream = new FileOutputStream(path);
        for (int i = 0; i < persons.length; i++) {
            String data = String.format(("%s,%s,%s,%d"),
                    persons[i].getName(), persons[i].getLocation(),
                    persons[i].getMail(), persons[i].getAge());
            outputStream.write(data.getBytes());
            outputStream.write('₩n');  // 레코드 구분자
        }
        System.out.println("File 저장 완료");
        outputStream.close();
    }catch(Exception e) {
        System.err.println(e.getMessage());
    }
}
```

BufferedReader 예제 3

- Main.java

```
ArrayList<Member> members = new ArrayList<>();
File file = new File(path);
if (file.exists()) {
    try {
        FileReader inputStream = new FileReader(file);
        BufferedReader reader = new BufferedReader(inputStream);
        String line;
        while ((line = reader.readLine()) != null) {
            String[] data = line.split(",");
            Member member = new Member(data[0], data[1], data[2],
                                                    Integer.parseInt(data[3]));
            members.add(member);
        }
```

BufferedReader 예제 3

- Main.java

```
            inputStream.close();
            reader.close();
            for (int i = 0; i < members.size(); i++)
                System.out.println(members.get(i).toString());
        } catch (Exception e) {
            System.err.println(e.getMessage());
        }
    } else {
        System.out.println("입력 파일이 없습니다");
    }
```

BufferedReader/BufferedWriter 예제 1

```
public static void main(String[] args) {
    Reader reader = new InputStreamReader(System.in);
    BufferedReader keyboard = new BufferedReader(reader);
    Writer writer = new OutputStreamWriter(System.out);
    BufferedWriter monitor = new BufferedWriter(writer);

    while (true) {
        String line;
        try {
            monitor.write(" 입력해주세요 ");
            monitor.flush();
            if ((line = keyboard.readLine()) == null)
                break;
            monitor.write(line+ "₩n");
            monitor.flush();
        } catch (IOException e) {
            System.err.println(e.getMessage());
        }
    }
```

BufferedReader/BufferedWriter 예제 1

```
    try {
        keyboard.close();
        monitor.close();
    } catch (IOException e) {
        System.err.println(e.getMessage());
    }
}
```

BufferedReader/BufferedWriter 예제 2

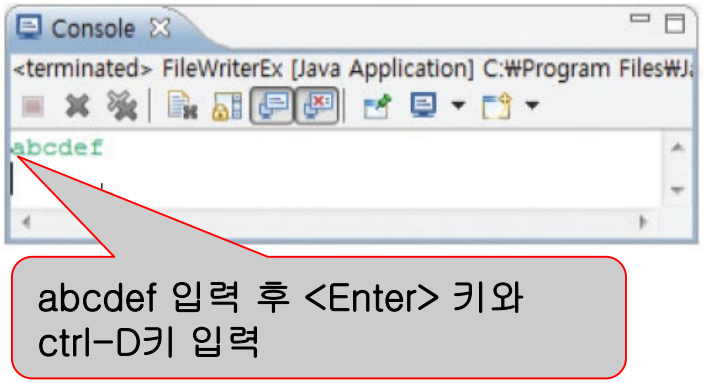
```
public static void main(String[] args) {
    String source = ".₩₩data₩₩test.txt";
    String taget = ".₩₩data₩₩test.cpy";

    File file = new File(source);
    if (file.exists()) {
        try {
            BufferedReader reader = new BufferedReader(new FileReader(source));
            BufferedWriter writer = new BufferedWriter(new FileWriter(taget, false));

            Date date = new Date();
            long start = date.getTime();
            String line;
            while ((line = reader.readLine()) != null) {
                writer.write(line);
                writer.newLine();
                writer.flush();
            }
```

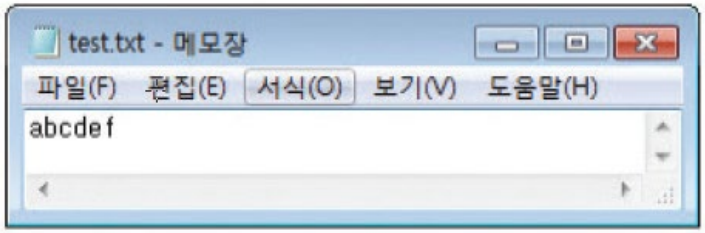
BufferedReader/BufferedWriter 예제 2

```
            date = new Date();
            long end = date.getTime();
            System.out.println("복사 시간 : " + (end - start));
            reader.close();
            writer.close();
        } catch (IOException e) {
            System.err.println(e.getMessage());
        }
    }
}
```

File에 문자 단위 Data를 쓰는 예제

- Keyboard에서 입력 받은 Data를 test.txt 파일에 저장하는 코드를 작성하라.





실행 결과 test.txt 파일 생성

File에 문자 단위 Data를 쓰는 예제

- Keyboard 지정 방법
  - Scanner로 지정

```
Scanner keyboard = new Scanner(System.in);
```

  - InputStream으로 지정

```
InputStream reader = System.in;
```

  - InputStreamReader로 지정

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

File에 문자 단위 Data를 쓰는 예제

- 출력 File을 지정하는 방법
  - FileWriter
  - FileWriter with append
    (File을 덮어 쓰지 않고 이어서 작성)
  - BufferedWriter
  - PrintWriter
    - PrintWriter와 BufferedWriter의 차이는 단 하나로 개행을 하는 문자열을 넣을 수 있음
    - 다른 쓰기 메소드들은 개행을 하는 문자를 따로 적어주어야 했는데, PrintWriter는 println() 메소드를 통해 파일에 개행 된 문자열을 넣을 수 있음
    - print()는 write() 메소드와 동일

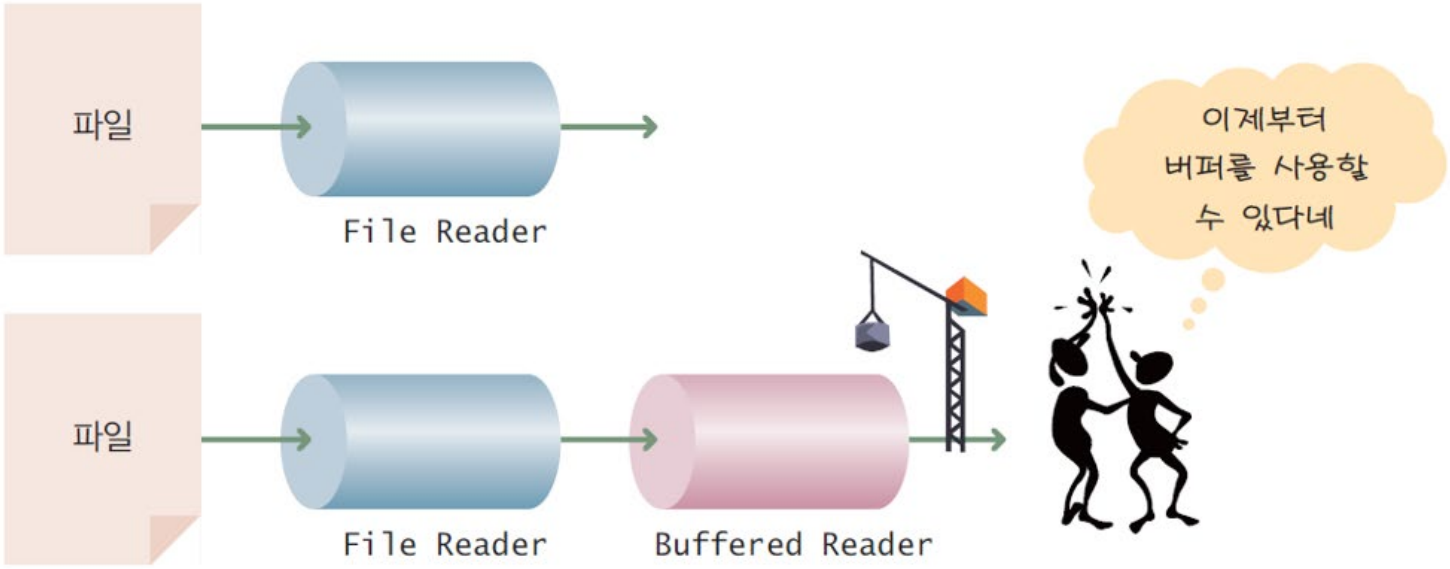
File에 문자 단위 Data를 쓰는 예제

- Buffer Stream의 연결

```
FileWriter writer1 = new FileWriter("output.txt");
BufferedWriter writer2 = new BufferedWriter(writer1);
```

↓

```
BufferedWriter writer = new BufferedWriter(
                            new FileWriter("output.txt"));
```



File에 문자 단위 Data를 쓰는 예제

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

    Scanner keyboard = new Scanner(System.in);
    String message = keyboard.nextLine();
    char[] charArr = message.toCharArray();
    try {
        FileWriter writer = new FileWriter(path);
        for (int i = 0; i < message.length(); i++)  {
            writer.write(charArr[i]);
        }
        writer.close();
    } catch (IOException e) {
        System.err.println(e.getMessage());
    }
    keyboard.close();
}
```

File에 문자 단위 Data를 쓰는 예제

```
public static void main(String[] args) {
    String path = ".₩₩data₩₩test.txt";
    int ch;
    try {
        InputStream reader = System.in;
        FileWriter writer = new FileWriter(path);
        while ((ch = reader.read()) != EOF) {
            writer.write(ch);
        }
        writer.flush();
        reader.close();
        writer.close();
    } catch (IOException e) {
        System.err.println(e.getMessage());
    }
}
```

File에 문자 단위 Data를 쓰는 예제

```
public static void main(String[] args) {
    String path = ".₩₩data₩₩test.txt";
    int ch;
    try {
        InputStreamReader reader = new InputStreamReader(System.in);
        PrintWriter writer = new PrintWriter(path);
        while ((ch = reader.read()) != EOF) {
            writer.print((char) ch);
        }
        writer.flush();
        reader.close();
        writer.close();
    } catch (IOException e) {
        System.err.println(e.getMessage());
    }
}
```

String 변환 방법

- String을 char[]로 변환하는 방법
  - toCharArray() 메소드
    - java.lang.String 클래스의 toCharArray() 메소드는 주어진 문자열을 char 배열 형태로 반환

```
String str = "Hello World";
char[] charArr = str.toCharArray();
{ 'H', 'e', 'l', 'l', 'o', 'W', 'o', 'r', 'l', 'd' }
```

  - getChars() 메소드
    - String에서 지정 범위의 단일 문자를 char 배열로 복사

```
String str = "Hello World";
char[] charArr = new char[str.length()];
str.getChars(0, str.length(), charArr, 0);
```

String 변환 방법

- char[]을 String으로 변환하는 방법
  - String 생성자
  - String.valueOf() 메소드
  - StringBuilder (append() 메소드)
  - Stream

  - String 생성자
    - char 배열을 String 생성자의 인자로 넣으면 문자열로 변환되어 String 객체가 생성

```
char[] charArr = {'H', 'e', 'l', 'l', 'o', 'W', 'o', 'r', 'l', 'd' };
String str = new String(charArray);
```

String 변환 방법

- String.valueOf() 메소드
  - String.valueOf()의 인자로 char[]를 전달하면 문자열로 변환된 String 객체가 생성

```
char[] charArr = { 'H', 'e', 'l', 'l', 'o', 'W', 'o', 'r', 'l', 'd' };
String str = String.valueOf(charArray);
```

- StringBuilder
  - StringBuilder의 append()를 이용하여 다음과 같이 char[]를 문자열로 변환할 수 있음

```
char[] charArr = { 'H', 'e', 'l', 'l', 'o', 'W', 'o', 'r', 'l', 'd' };
StringBuilder builder = new StringBuilder();
for (char ch : charArray) {
    builder.append(ch);
}
```

String 변환 방법

- Stream
  - Stream으로 char[]를 String으로 변환하고, String들을 합쳐서 하나의 String으로 만들 수 있음

```
Character[] charArr = { 'H', 'e', 'l', 'l', 'o', 'W', 'o', 'r', 'l', 'd' };
Stream<Character> charStream = Arrays.stream(charArray);
String str = charStream.map(String::valueOf)
                                    .collect(Collectors.joining());
```

다양한 입력 방법

```
public static void main(String[] args) throws IOException {
    InputStreamReader inputStream = new InputStreamReader(System.in);
    BufferedReader reader = new BufferedReader(inputStream);

    System.out.print(" 집 주소 입력 : ");
    String address = reader.readLine( );          // 문자열 입력
    System.out.println("나의 집주소는 " + address + "입니다.");
    System.out.print("성별(M 또는 F)? ");
    int gender = inputStream.read( );
    System.out.println("성별 : " + "(아스키코드) " + gender + "₩n₩t" + " (문자) "
            + (char) gender);
    reader.close();
    inputStream.close();
}
```

LineNumberReader 클래스

- BufferedReader 클래스를 상속 받은 클래스
- Line 번호를 추적해 관리하는 Buffering된 문자 입력 Stream
- 이 클래스에는 현재의 Line 번호를 설정하는 setLineNumber(int) 메소드와 현재의 행 번호를 반환하는 getLineNumber() 메소드가 정의되어 있음
- Text를 한 Line씩 번호를 매기면서 읽는 클래스
- 사용 방법
  - 다음과 같은 방법으로 객체를 생성

LineNumberReader 클래스

- 사용 방법
  - readLine() 메소드를 호출해서 Text를 읽음

```
String str = reader2.readLine();
```

텍스트 한 줄을 읽어서
리턴하는 메소드

  - 읽어 들인 행의 번호는 getLineNumber() 메소드를 호출해서 가져옴

```
int num = reader2.getLineNumber();
```

바로 전에 읽은 행 번호를
리턴하는 메소드

LineNumberReader 예제 1

- 텍스트 파일에 행 번호를 붙이면서 읽는 프로그램

```
public static void main(String[] args) {
    String path = ".//data//test.txt";
    File file = new File(path);
    if (file.exists()) {
        try {
            LineNumberReader reader = new LineNumberReader(new FileReader(file));
            String line;
            while ((line = reader.readLine()) != null) {
                int lineNo = reader.getLineNumber();
                System.out.printf("%03d : %s₩n", lineNo, line);
            }
            reader.close();
        } catch (FileNotFoundException e) {
            System.out.println("파일이 존재하지 않습니다.");
        } catch (IOException e) {
            System.out.println("파일을 읽을 수 없습니다.");
        }
    }
}
```