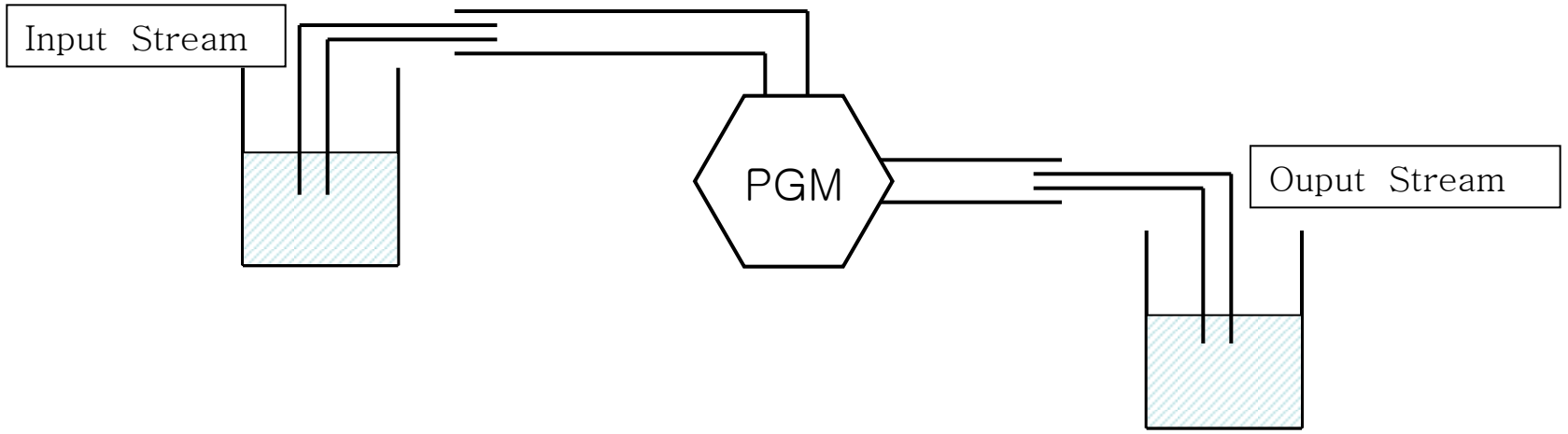
Chap 14. Advanced I/O Streams

학습목표

1. Stream
2. Node Stream / Processing Stream
3. File Read/Write
4. Consol I/O
5. Object Serialization

❑ Stream

- data의 연속 된 흐름을 말한다.
- data 시작(source stream/input stream)과 data의 끝 (sink stream/output stream)을 말한다.
  특히 이것을 Node Stream이라고 한다.
- Node stream : file, memory, 쓰레드나 프로세스 사이의 pipe 등등
- 단방향이다.



- Byte 단위 입/출력 처리 : InputStream, OutputStream
  이진 데이타 처리시 이용

- Character 단위 입/출력 처리 : Reader, Writer
  text 처리시 이용

	Character Streams	Byte Streams
Source Streams	Reader	InputStream
Sink Streams	Writer	OutputStream

❑ Data의 근원이 되는 물리적 장치와 직접 연결하는 stream

Type	Character Streams	Byte Streams
File	FileReader FileWriter	FileInputStream FileOutputStream
Memory Array	CharArrayReader CharArrayWriter	ByteArrayInputStream ByteArrayOutputStream
Memory String	StringReader StringWriter	
Pipe	PipedReader PipedWriter	PipedInputStream PipedOutputStream

❑ Data를 가공해 주는 stream

Type	Character Streams	Byte Streams
Buffering	BufferedReader BufferedWriter	BufferedInputStream BufferedOutputStream
Filtering	FilterReader FilterWriter	FilterInputStream FilterOutputStream
Converting between bytes and character	InputStreamReader OutputStreamWriter	
Object serialization		ObjectInputStream ObjectOutputStream
Data conversion		DataInputStream DataOutputStream
Counting	LineNumberReader	LineNumberInputStream
Peeking ahead	PushbackReader	PushbackInputStream
Printing	PrintWriter	PrintStream

"c:₩data₩text.txt"에서 문자열을 읽기 위한 스트림 작성

File 지정

```
String fileName ="c:/data/text.txt";
```

Stream생성

3가지 방법 모두 동일

```
1) FileReader fr = new FileReader(fileName);
   BufferedReader br = new BufferedReader( fr );
2) BufferedReader br
              = new BufferedReader( new FileReader(fileName) );
3) BufferedReader br = new BufferedReader(
         new InputStreamReader(new FileInputStream(fileName)) );
```

READ

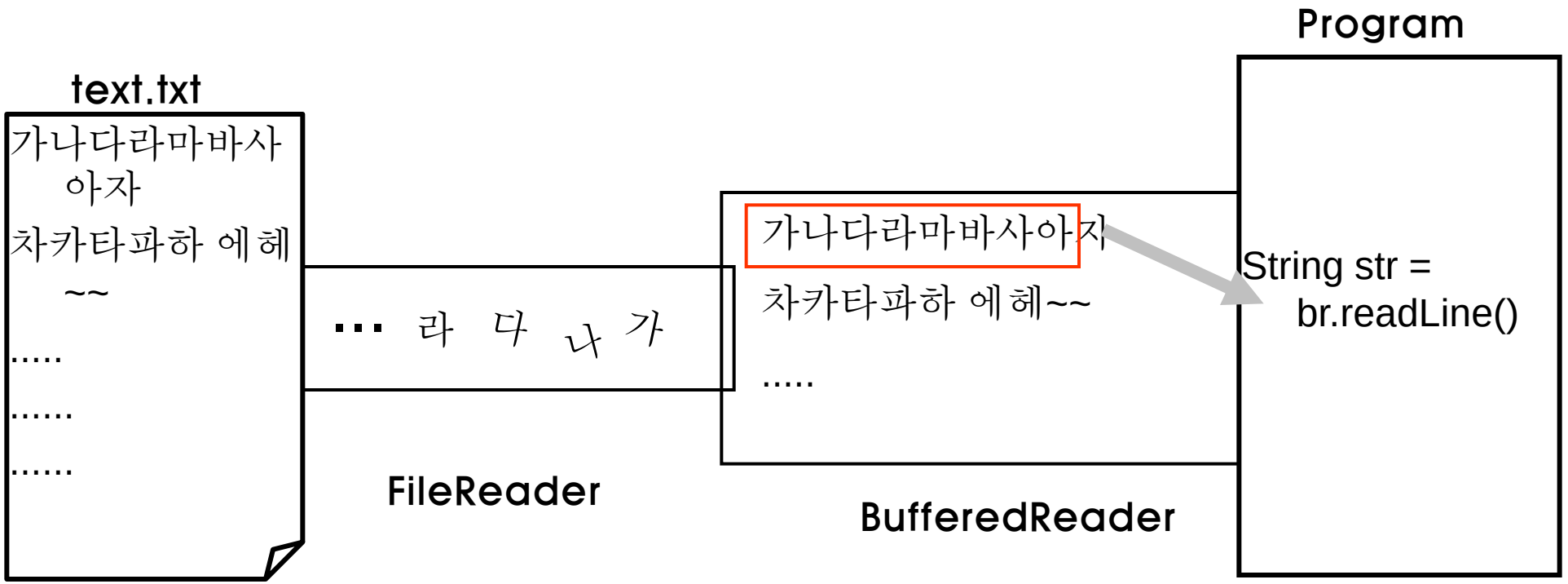
```
s = br.readLine();  //결과의 null여부로 파일을 다 읽었는지 확인
```

Stream닫기

```
br.close();
```

File I / O – Read

```
BufferedReader br = new BufferedReader( new FileReader("text.txt") );
```



File I / O – Read Example


```
import java.io.*;
public class ReadFile {
    public static void main ( String args[] ) {

        try {
            BufferedReader in
                 = new BufferedReader(new FileReader("c:/data/text.txt"));
            String s;

            while ( ( s = in.readLine() ) != null ) {
                System.out.println( s );
            }

            in.close();
        } catch ( FileNotFoundException e1 ) {
            System.err.println( "File not found: " + "c:/data/text.txt" );
        } catch ( IOException e2 ){
            e2.printStackTrace();
        }
    }
}
```

"c:₩data₩text2.txt"에 문자열을 쓰기 위한 스트림 작성

File 지정

```
String file ="c:/data/text2.txt";
```

Stream생성

2가지 방법 모두 동일

```
1) FileWriter fw = new FileWriter(fileName);
    PrintWriter  pw = new PrintWriter ( fw );
2) PrintWriter pw
          = new PrintWriter( new FileWriter(fileName) );
```

READ

```
pw.println( msg[inx] );
```

Stream닫기

```
pw.close();
```

File I / O – Write Example


```
import java.io.*;
public class WriteFile {
    public static void main ( String args[] ) {
        try {
            BufferedReader in
                = new BufferedReader(new FileReader("c:/data/test.txt"));
            PrintWriter out
                = new PrintWriter(new FileWriter("c:/data/test2.txt"));
            String s;
            while ( ( s = in.readLine() ) != null ) {
                out.println( s );
            }
            in.close();
            out.close();
        } catch ( IOException e ) {
            e.printStackTrace();
        }
    }
}
```

System.out

PrintStream 객체로 Terminal 참조

System.in

InputStream 객체로 키보드 참조

System.err

PrintStream 객체로 Terminal 참조

- 표준 출력은 콘솔(모니터) 출력이며, System.out의 메소드를 사용한다.
  - println() : 출력 후, new line
  - print() : new line 없이 출력
  - print() 와 println()은 다양한 타입을 화면에 출력하기 위해 overload 되어 있음

Consol I / O – Standard Input


```
import java.io.*;
public class KeyboardInput {
    public static void main ( String [] args ) {
        String s;
        InputStreamReader ir = new InputStreamReader( System.in );
        BufferedReader in = new BufferedReader( ir );
        System.out.println( "Type 'exit' to exit." );
        try {
            while ( !(s = in.readLine()).equals("exit") ) {
                System.out.println( "Read: " + s );
            }
            in.close();
        } catch ( IOException e ) {
            e.printStackTrace();
        }
    }
}
```

Serialization (객체 직렬화)

- 객체 메모리를 한 줄로 늘어선 byte 형태로 만드는 것

- 객체 직렬화 대상
  - 멤버 변수
  - Class variable(static), 메소드, 생성자는 Serialization 대상에서 제외

- 장점
  - 완전한 객체 데이터 보장
  - 객체를 얻는 즉시 사용

- 관련 Stream (Processing Stream)
  - ObjectInputStream : Unserialization
  - ObjectOutputStream : Serialization

Serialization 대상 클래스 선언

- Serialization 대상이 되는 객체는 반드시 implements Serializable

```
import java.io.Serializable;
public class EmpInfo implements Serializable {
    private String empId;
    private String name;

    public EmpInfo( String empId, String name ) {
        this.empId = empId;
        this.name = name;
    }
    public String getEmpId() {
        return empId;
    }
    public void setEmpId( String empId ) {
        this.empId = empId;
    }
    public String getName() {
        return name;
    }
    public void setName( String name ) {
        this.name = name;
    }
}
```

Serialization – Example

```
import java.io.*;
public class SerializeTest {
    public static void main( String[] args ) {
        FileOutputStream empOutputFile = null;
        ObjectOutputStream empOutputStream = null;
        EmpInfo[] empList = { new EmpInfo( "11111", "홍길동" ),
                              new EmpInfo( "22222", "고길동" ),
                              new EmpInfo( "33333", "김길동" ), };
        try {
            empOutputFile = new FileOutputStream("d:/data/emplist.ser");
            empOutputStream = new ObjectOutputStream( empOutputFile );
            empOutputStream.writeObject( empList );
            empOutputStream.close();
        } catch ( IOException ioe ) {
            ioe.printStackTrace();
        }
    }
}
```

Unserialization – Example

```
import java.io.*;
public class SerializeTest {
    public static void main( String[] args ) {
        FileInputStream empInputFile = null;
        ObjectInputStream empInputStream = null;
        try {
            empInputFile = new FileInputStream("d:/data/emplist.ser");
            empInputStream = new ObjectInputStream( empInputFile );
            EmpInfo[] readList = (EmpInfo[])empInputStream.readObject();
            empInputStream.close();

            for ( int inx = 0 ; inx < readList.length ; inx++ ) {
                System.out.println( "사번 : " + readList[inx].getEmpId() +
                                    "  이름 : " + readList[inx].getName() );
            }
        } catch ( ClassNotFoundException ce ) {
            ce.printStackTrace();
        } catch ( IOException ioe ) {
            ioe.printStackTrace();
        }
    }
}
```