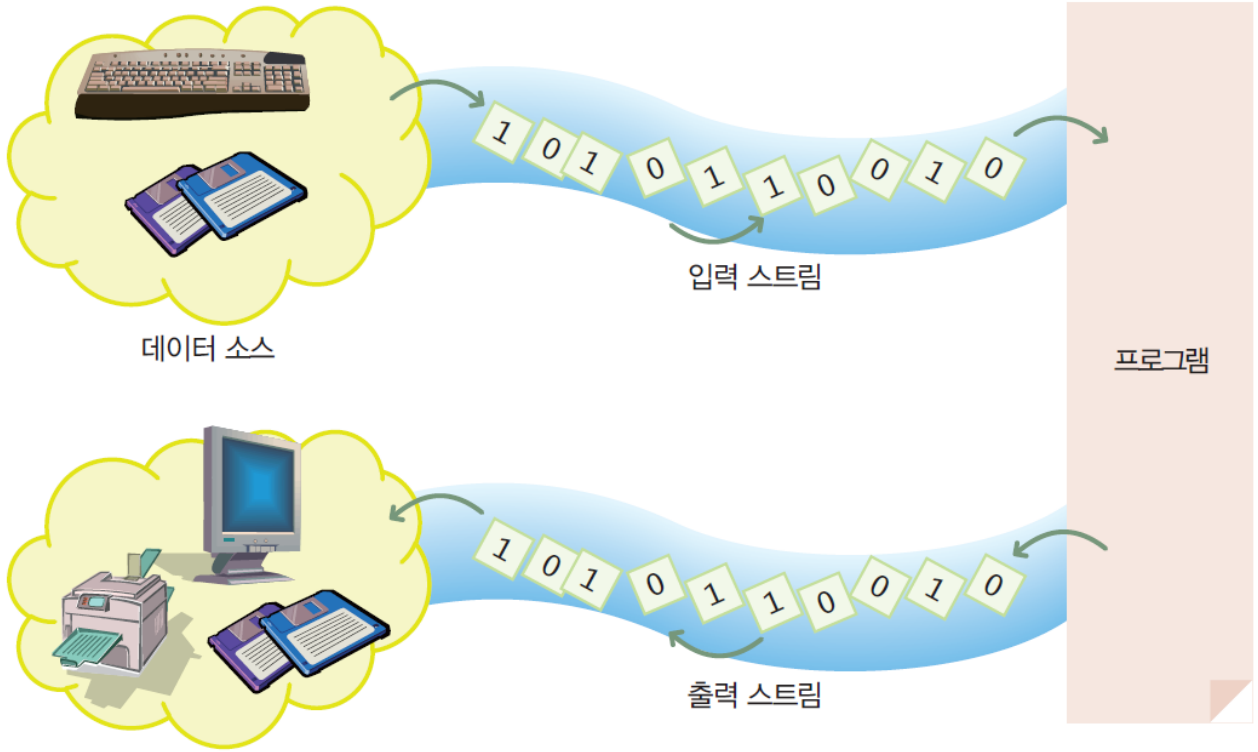
Lec_09 파일 입출력

자바 프로그래밍2_13주

스트림(stream)

- 스트림(stream) : "순서가 있는 데이터의 연속적인 흐름"
- 스트림은 입출력을 물의 흐름처럼 간주하는 것



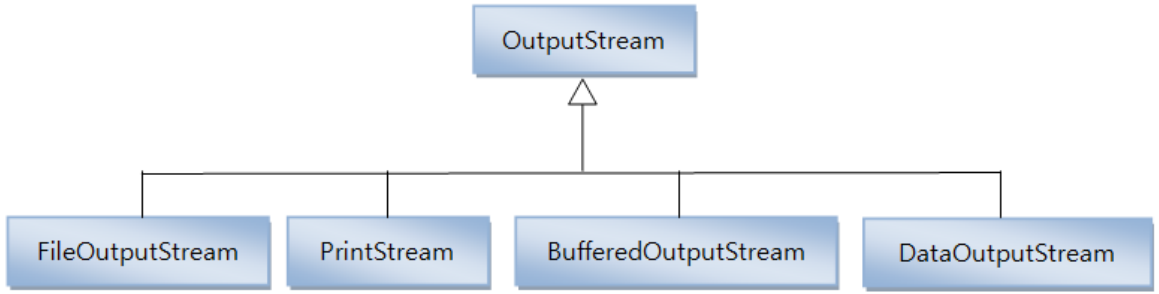
입출력 스트림의 종류

- 입출력의 단위에 따라서 분류
- **바이트 스트림(byte stream)**
  - 8비트의 바이트 단위로 입출력하는 클래스
  - 이진 데이터를 읽고 쓰기 위하여 사용
  - 그림, 멀티미디어, 문자 등 모든 종류의 자료를 다룸
  - 클래스 이름에 InputStream(입력)과 OutputStream(출력)이 붙는다.

- **문자 스트림(character stream)**
  - 문자 단위로 입출력하는 클래스 - 문자만 다룸
  - 문자 스트림은 유니코드 단위로 입출력한다.
  - 클래스 이름에 Reader(입력)와 Writer(출력)가 붙는다.

바이트 출력 스트림

- OutputStream
  - 바이트 기반 출력 스트림의 최상위 클래스로 추상 클래스



  - OutputStream의 주요 메소드

리턴타입	메소드	설명
void	wrtie(int b)	출력 스트림으로 1 바이트를 보낸다.
void	write(byte[] b)	출력 스트림에 매개값으로 주어진 바이트 배열 b 의 모든 바이트를 보낸다.
void	write(byte[] b, int off, int len)	출력 스트림에 매개값으로 주어진 바이트 배열 b[off] 부터 len 개까지의 바이트를 보낸다.
void	flush()	버퍼에 잔류하는 모든 바이트를 출력한다.
void	close()	사용한 시스템 자원을 반납하고 출력 스트림을 닫는다.

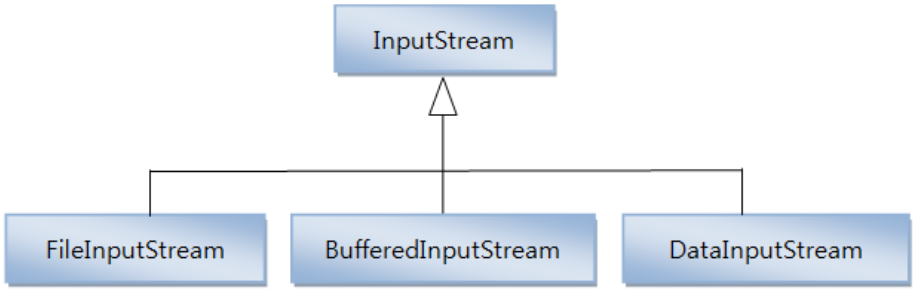
바이트 출력 스트림

import java.io.*; //파일 입출력 시 반드시 추가

```
public class Lec13_01 {
	public static void main(String[] args) throws Exception  {
		OutputStream os=new FileOutputStream("out.txt");
		 //추가하고 싶으면 두번째 인수로 true를 사용 즉, FileOutputStream("output.txt", true);

		byte a=3;
		os.write(a);

		byte[] array= {4,5,6};
		os.write(array);  //배열의 모든 바이트를 출력

		byte[] dim= {10,11,12,13,14,15};
		os.write(dim, 1, 3); //배열 인덱스1부터 3개 출력
		os.flush(); //출력 버퍼의 모든 바이트 출력
		os.close(); //출력 스트림을 닫음
		System.out.println("program stop");
	}
}
```

바이트 입력 스트림

- InputStream
  - 바이트 기반 입력 스트림의 최상위 클래스로 추상 클래스



  - InputStream 클래스의 주요 메소드

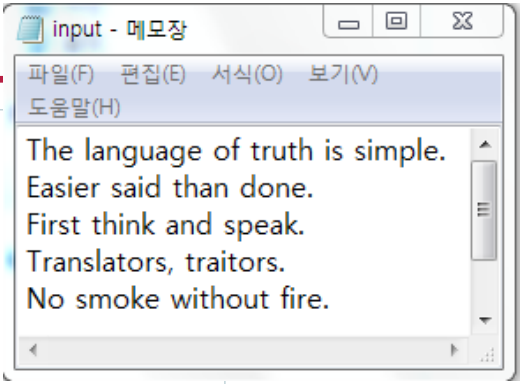
리턴 타입	메소드	설명
int	read()	1byte를 읽고 읽은 바이트 리턴
int	read(byte[] b)	읽은 바이트를 매개값으로 주어진 배열에 저장하고 읽은 바이트 수 리턴
int	read(byte[] b, int off, int len)	Len개의 바이트를 읽고 매개값으로 주어진 배열에서 b[off]부터 len개까지 저장, 읽은 바이트 수 리턴
void	close	입력 스트림을 닫음

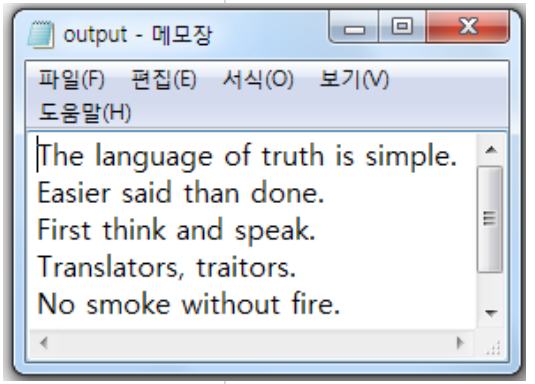
바이트 입력 스트림

```
import java.io.*; //파일 입출력 시 반드시 추가

public class Lec13_02 {
	public static void main(String[] args) throws Exception {
		InputStream in = new FileInputStream("out.txt");

		while(true) {
			int data = in.read();	//한번에 1byte read
			if(data == -1)		//파일의 마지막에 도달하였을 경우
				break;
			System.out.println(data);
		}
		byte[] buffer = new byte[7];
		int readbyte=in.read(buffer, 2, 3);  //파일 처음부터3 byte를 읽어 buffer[2], buffer[3], buffer[4]에 저장
		if(readbyte != -1) {
			for(int i=0; i<buffer.length; i++) {
				System.out.println(i + ")"+ buffer[i]);
			}
		}
	}
}
```

파일 입출력 - 바이트 스트림

```
import java.io.*;  //파일 입출력 시 반드시 추가
public class CopyFile1 {
    public static void main(String[] args) throws IOException {
      // try-with-resources 문장,  SE 7 버전 부터 가능
     try (InputStream in= new FileInputStream("input.txt");
           OutputStream out= new FileOutputStream("output.txt")){
           int c;
            //read()는 int 값 반환, 파일의 마지막이면 -1 반환
            while ((c = in.read() ) != -1) {
                  out.write(c); //파일에서 read 한 값(c)을 파일에 write
            //표준 출력장치로 출력, int 로 반환된 값을 char 로 변환하면 문자로 처리
           //단, 바이트 스트림은 1바이트만 처리(한글은 문자로 변환해도 깨짐)
               System.out.println((char)c); }
        }
    }
}
```





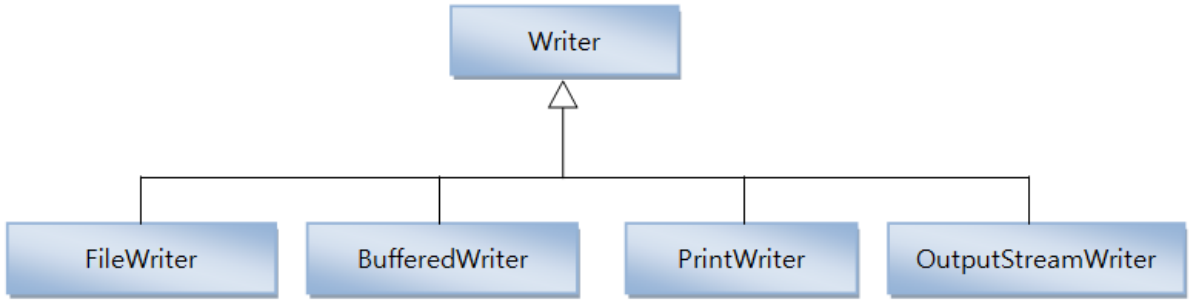
LAB: 이미지 파일 복사하기

- 이미지 파일은 이진 파일

```
//소스 입력 후 Ctrl + Shift + O를 눌러 필요한 파일 포함
public class ByteStreamsLab {
        public static void main(String[] args) throws IOException {
                Scanner scan = new Scanner(System.in);
                System.out.print("원본 파일 이름을 입력하시오: ");  String inFileName = scan.next();
                System.out.print("복사 파일 이름을 입력하시오: ");  String outFileName = scan.next();

                try (InputStream inStream = new FileInputStream(inFileName);
                    OutputStream outStream = new FileOutputStream(outFileName)) {
                        int c;
                        while ((c = inStream.read()) != -1) {
                                outStream.write(c);
                        }
                }
                System.out.println(inFileName + "을 " + outFileName + "로 복사하였습니다. ");
}}
```

문자 출력 스트림

- Writer
  - 문자 기반 출력 스트림의 최상위 클래스로 추상 클래스



  - Writer의 주요 메소드

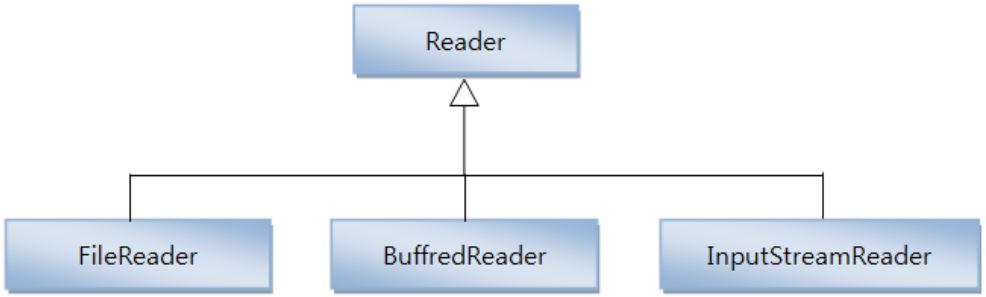
리턴타입	메소드	설명
void	wrtie(int c)	출력 스트림으로 매개값으로 주어진 한 문자를 보낸다.
void	write(char[] cbuf)	출력 스트림에 매개값으로 주어진 문자 배열 cbuf 의 모든 문자를 보낸다.
void	write(char[] cbuf, int off, int len)	출력 스트림에 매개값으로 주어진 문자 배열 cbuf[off] 부터 len 개까지의 문자를 보낸다.
void	write(String str)	출력 스트림에 매개값으로 주어진 문자열을 전부 보낸다.
void	write(String str, int off, int len)	출력 스트림에 매개값으로 주어진 문자열 off 순번부터 len 개까지의 문자를 보낸다.
void	flush()	버퍼에 잔류하는 모든 문자열을 출력한다.
void	close()	사용한 시스템 자원을 반납하고 출력 스트림을 닫는다.

문자 출력 스트림

```
public class Lec13_03 {
	public static void main(String[] args) throws IOException {
		Writer out = new FileWriter("file.txt");
		char ch='b';
		out.write('a'); //4byte에서 끝 2byte만 출력스트림으로 보냄
		out.write(ch);

		char[] array= {'c','d', 'e', 'f', 'g'};
		out.write(array);

		char[] dim= {'h', 'i', 'j', 'k'};
		out.write(dim, 1, 2);

		String str="string";
		out.write(str);

		out.flush();
		out.close();
	}
}
```

문자 입력 스트림

- Reader
  - 문자 기반 입력 스트림의 최상위 클래스로 추상 클래스



  - Reader의 주요 메소드

메소드		설명
int	read()	입력 스트림으로부터 한개의 문자를 읽고 리턴한다.
int	read(char[] cbuf)	입력 스트림으로부터 읽은 문자들을 매개값으로 주어진 문자 배열 cbuf 에 저장하고 실제로 읽은 문자 수를 리턴한다.
int	read(char[] cbuf, int off, int len)	입력 스트림으로부터 len 개의 문자를 읽고 매개값으로 주어진 문자 배열 cbuf[off] 부터 len 개까지 저장한다. 그리고 실제로 읽은 문자 수인 len 개를 리턴한다.
void	close()	사용한 시스템 자원을 반납하고 입력 스트림을 닫는다.

문자 입력 스트림

```
public class Lec13_04 {
	public static void main(String[] args) throws IOException {
		Reader read = new FileReader("file.txt");
		while(true) {
				int data = read.read();  //한번에 한 문자 read
				if(data == -1) //파일의 마지막에 도달하였을 경우
					break;
				System.out.println((char)data);
		}

	  char[] buffer = new char[8];
	  int readbyte=read.read(buffer, 2, 3);  //파일 처음부터 3 byte를 읽어 buffer[2], buffer[3], buffer[4]에 저장
	  if(readbyte != -1) {
		for(int i=0; i<buffer.length; i++) {   System.out.println(i + ")"+ (char)buffer[i]); }
	  }
		read.close();
	}
}
```

보조 스트림

- 보조 스트림
  - 다른 스트림과 연결 되어 여러 가지 편리한 기능을 제공해주는 스트림
    - 문자 변환, 입출력 성능 향상, 기본 데이터 타입 입출력, 객체 입출력 등의 기능을 제공

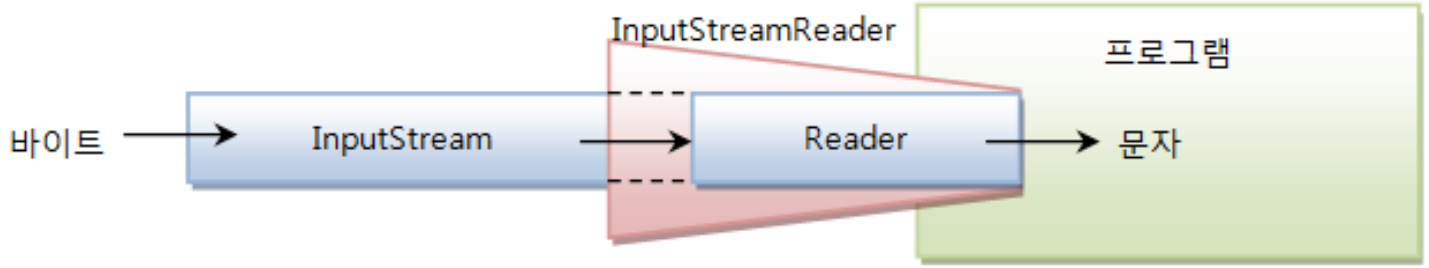
  - 보조 스트림 생성

```
보조스트림 변수 = new 보조스트림(연결스트림)
```

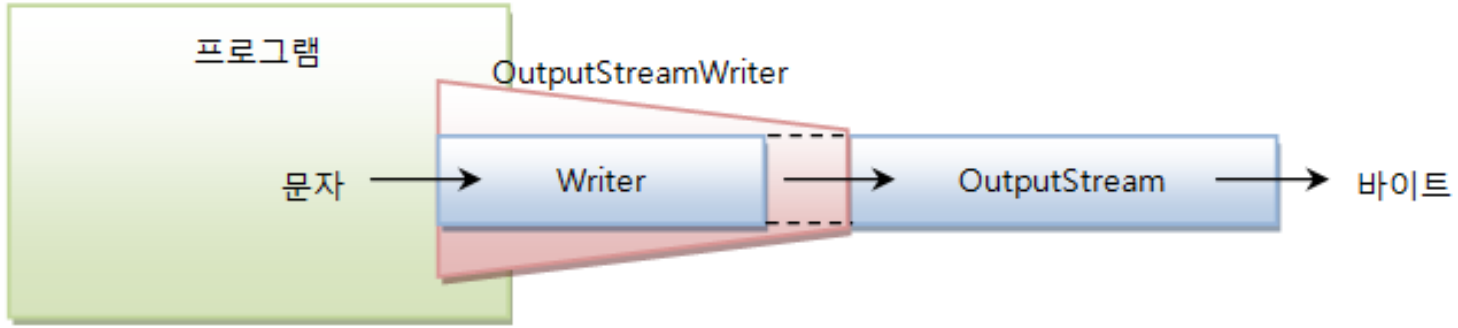
  - 보조 스트림 체인 – 다른 보조 스트림과 연결되어 역할 수행

보조 스트림

- 문자 변환 보조 스트림
  - 소스 스트림이 바이트 기반 스트림이지만 데이터가 문자일 경우 사용
    - Reader와 Writer는 문자 단위로 입출력 - 바이트 기반 스트림보다 편리
    - 문자셋의 종류를 지정할 수 있기 때문에 다양한 문자 입출력 가능
  - InputStreamReader



  - OutputStreamWriter



문자 변환 보조 스트림

```
public class Lec13_05 {
	public static void main(String[] args) throws Exception{
		write("문자 변환 스트림을 사용합니다");
		String data=read();
		System.out.println(data);
	}
	public static void write(String str) throws Exception{
		FileOutputStream fos=new FileOutputStream("test.txt");
		Writer wr=new OutputStreamWriter(fos);
		wr.write(str);		//OutputStreamWriter 보조 스트림을 이용해서 문자 출력
		wr.flush();
		wr.close();
	}
	public static String read() throws Exception{
		FileInputStream fis=new FileInputStream("test.txt");
		Reader rd=new InputStreamReader(fis);
		char[] buffer=new char[100];
		int readNum=rd.read(buffer); //InputStreamReader 보조 스트림을 이용해서 문자 입력
		rd.close();
		String data = new String(buffer, 0, readNum); //char  배열에서 읽은 수만큼 문자열로 반환
		return data;
	}
}
```

문자 엔코딩

- 문자 엔코딩
  - 자바에서는 StandardCharsets 클래스 안에 각 엔코딩 방법이 StandardCharsets.UTF_8, StandardCharsets.UTF_16과 같이 상수로 정의
  - String s = **new** String(100, StandardCharsets.UTF_8);
  - 파일에서 읽을 때는 InputStreamReader 클래스 사용

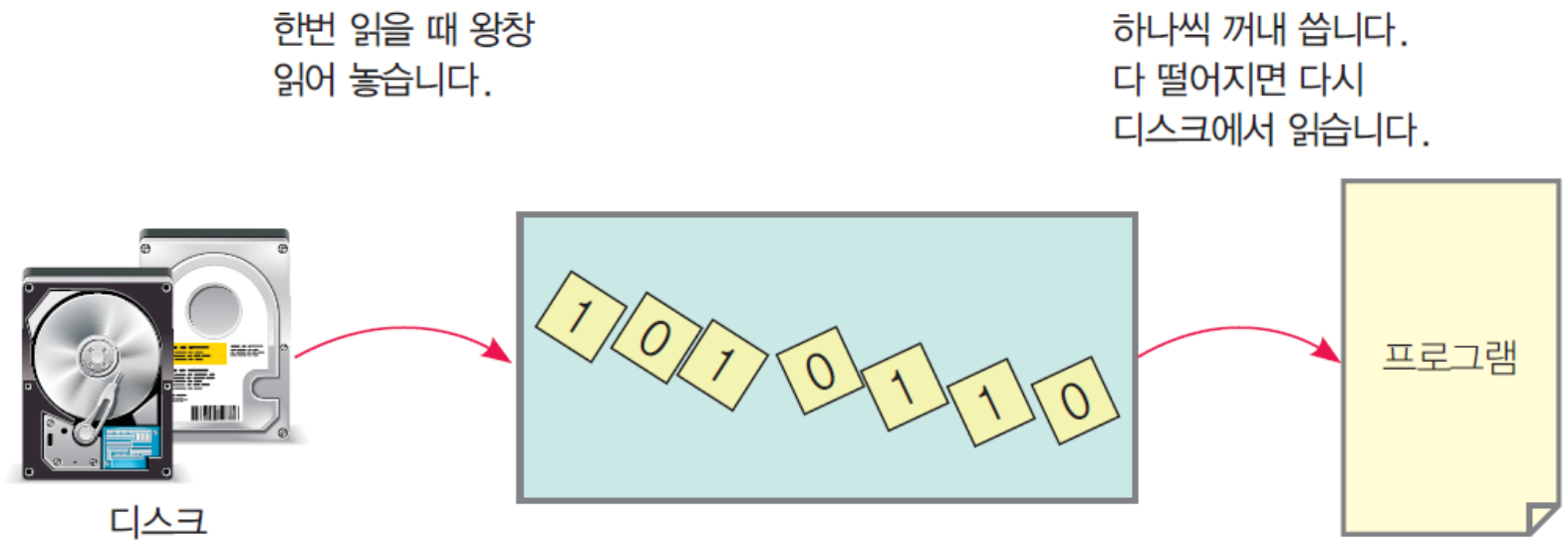
```
public class CharEncodingTest {
    public static void main(String[] args) throws IOException {
        File fileDir = new File("input.txt");
        BufferedReader in = new BufferedReader(new  InputStreamReader(new FileInputStream(fileDir), "UTF8"));
        String str;
        while ((str = in.readLine()) != null) {
            System.out.println(str);
        }
    }
}
```

성능향상 보조 스트림

- 버퍼를 이용해서 입출력 성능을 향상시키는 클래스들
  - File* 클래스는 read, write 메소드 호출할 때마다 파일로부터 직접 데이터를 read/write 때문에 프로그램 성능을 저하시킬 수 있다
  - 이런 문제를 해결하기 위해 필요한 데이터를 한꺼번에 읽어 두거나 출력데이터를 모아두었다 한꺼번에 출력(버퍼)
  - 스트림의 종류에 따라 4개의 클래스가 있음

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

성능향상 보조 스트림



버퍼 스트림의 개념

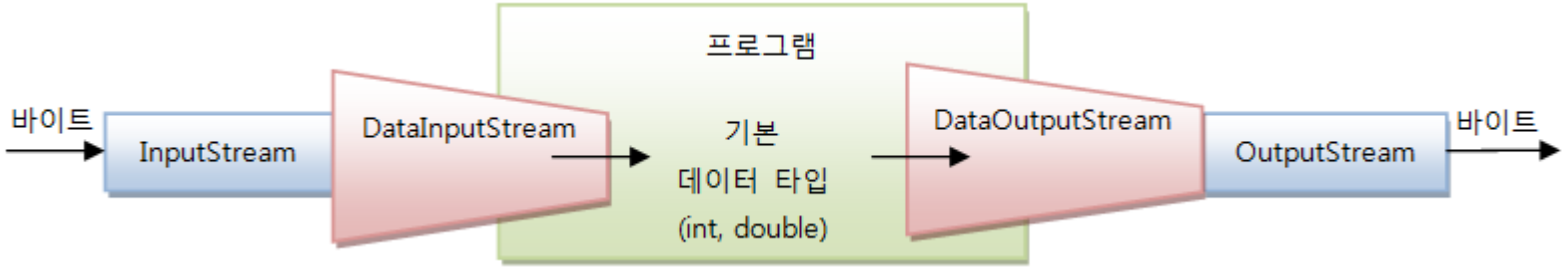
성능향상 보조 스트림

- 버퍼가 없는 스트림을 버퍼가 있는 스트림으로 변경
  - 버퍼 스트림 객체를 생성하면서 생성자의 인수로 버퍼가 없는 스트림 객체를 전달

```
import java.io.BufferedReader;
import java.io.FileReader;

public class Lec13_06 {
	public static void main(String[] args) throws Exception {
		BufferedReader  br=new BufferedReader(new FileReader("test.txt"));
		while(true) {
			String data=br.readLine(); //라인 단위로 문자열을 읽어 반환
			if(data == null)                //파일 끝에 도달했을 경우
				break;
			System.out.println(data);
		}
		br.close();
	}
}
```

기본 타입 입출력 보조 스트림

- DataInputStream 과 DataOutputStream 클래스
  - 기초 자료형 단위로 데이터를 읽고 쓸 수 있다.
  - 쓸 때와 동일한 순서로 읽어야 함



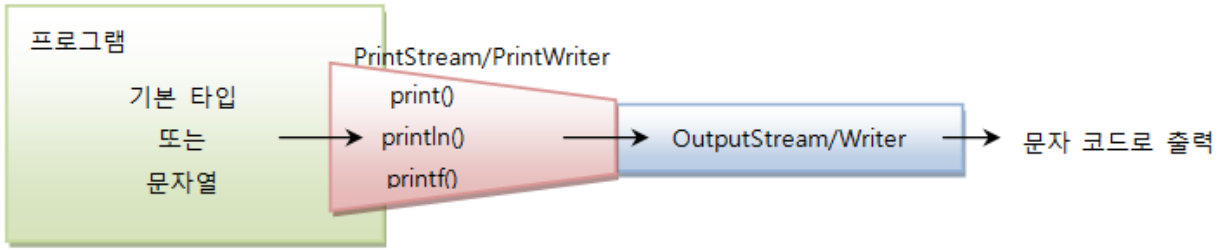
DataInputStream		DataOutputStream	
boolean	readBoolean()	void	writeBoolean(boolean v)
byte	readByte()	void	writeByte(int v)
char	readChar()	void	writeChar(int v)
double	readDouble()	void	writeDouble(double v)
float	readFloat()	void	writeFloat(float v)
int	readInt()	void	writeInt(int v)
long	readLong()	void	writeLong(long v)
short	readShort()	void	writeShort(int v)
String	readUTF()	void	writeUTF(String str)

기본 타입 입출력 보조 스트림

```
public class Lec13_07 {
    public static void main(String[] args) throws IOException {
            DataInputStream in = null;
            DataOutputStream out = null;
            int c;

            // 파일 스트림->버퍼 스트림->데이터 스트림
            out = new DataOutputStream(new BufferedOutputStream(new FileOutputStream("data.bin")));
            out.writeDouble(3.14);
            out.writeInt(100);
            out.writeUTF("자신의 생각을 바꿀 수 없다.");
            out.flush();
            out.close();
            in = new DataInputStream(new BufferedInputStream(new FileInputStream("data.bin")));
            System.out.println(in.readDouble());
            System.out.println(in.readInt());
            System.out.println(in.readUTF());
            in.close();
    }
}
```

프린터 보조 스트림

- PrintStream : 바이트 입력 ->텍스트 형태로 출력
- PrintWriter : 문자 입력 -> 텍스트 형태로 출력

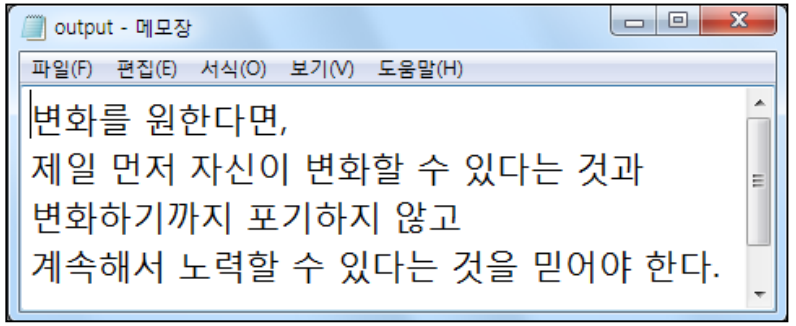


- println()은 데이터 끝에 개행문자 추가

PrintStream / PrintWriter			
void	print(boolean b)	void	println(boolean b)
void	print(char c)	void	println(char c)
void	print(double d)	void	println(double d)
void	print(float f)	void	println(float f)
void	print(int i)	void	println(int i)
void	print(long l)	void	println(long l)
void	print(Object obj)	void	println(Object obj)
void	print(String s)	void	println(String s)
		void	println()

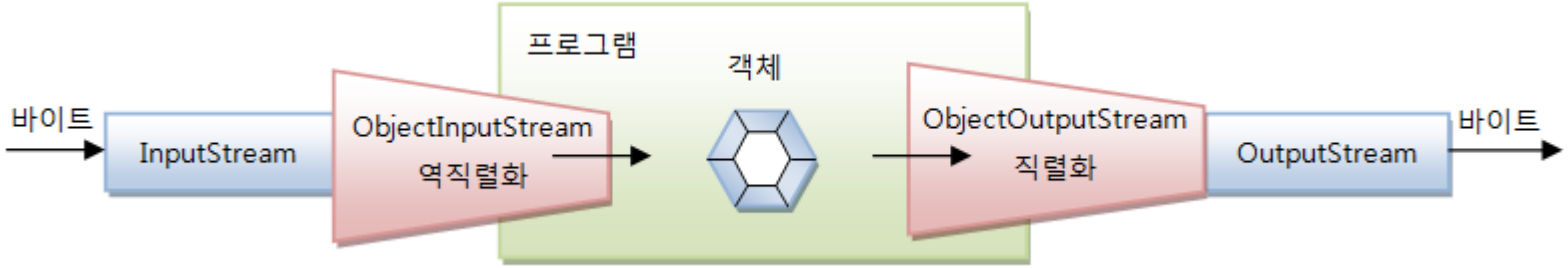
줄 단위 입출력

```
import java.io.*;
public class CopyFile2 {
	public static void main(String[] args) throws IOException {
		BufferedReader in = null;
		PrintWriter out = null;
		out = new PrintWriter(new FileWriter("output.txt"));
		out.println("변화를 원한다면,");	//라인 단위로 저장
		out.println("제일 먼저 자신이 변화할 수 있다는 것과");
		out.println("변화하기까지 포기하지 않고");
		out.println("계속해서 노력할 수 있다는 것을 믿어야 한다.");
		out.flush();  //버퍼를 수동으로 비움
		in = new BufferedReader(new FileReader("output.txt"));
		String line;
		while ((line = in.readLine()) != null) {   //라인단위로 읽어옴
			System.out.println(line);
		}
		if (in != null)		in.close();
		if (out != null)	out.close();
	}
}
```



객체 입출력 보조 스트림

- 객체를 파일 또는 네트워크로 입출력 할 수 있는 기능 제공
- 객체 직렬화
  - 객체는 문자가 아니므로 바이트 기반 스트림으로 데이터 변경 필요
- ObjectInputStream, ObjectOutputStream



- 직렬화가 가능한 클래스(Serializable)
  - 자바에서는 **Serializable 인터페이스를 구현한 클래스만 직렬화** 할 수 있도록 제한, transient 필드는 제외
  - 객체 직렬화 할 때 private 필드 포함한 모든 필드를 바이트로 변환 가능

객체 입출력 보조 스트림

- writeObject()와 readObject() 메소드
  - writeObject(객체)
    - 객체를 직렬화하여 출력 스트림으로 보냄
  - (객체 타입)readObject()
    - 입력 스트림에서 읽은 바이트를 역 직렬화하여 객체로 복원하여 반환
    - 강제 형 변환 필요

객체 입출력 보조 스트림

```
import java.io.*; //Serializable 인터페이스를 구현하기 위해 필요
import java.util.*;

class Pet implements Serializable {
   private String name;        // 이름
   private int age;            // 나이
   private double weight;      // 몸무게

   public Pet(String name, int age, double weight) {
          this.name = name;
          this.age = age;
          this.weight = weight;
   }

   public String toString() {
          return "Name = " + name + "  " + "Age = " + age + "  " + "Weight = " + weight;
   }
}
```

객체 입출력 보조 스트림

```
public class Lec13_08 {
    public static void main(String[] args) throws Exception {
            writeList();
            List<Pet> list = readList();
            for(Pet obj : list) System.out.println(obj);
    }
    public static List<Pet> readList() throws Exception{
            ObjectInputStream in = new ObjectInputStream(new FileInputStream("object.dat"));
            @SuppressWarnings("unchecked")
            List<Pet> list = (List<Pet>)in.readObject();
            return list;
    }
    public static void writeList() throws Exception {
            List<Pet> list = new ArrayList<>();
            ObjectOutputStream out = null;
            list.add(new Pet("해바라기", 7, 10.2));    list.add(new Pet("금강초롱", 4, 4.2));
            // FileOutputStream 객체를 생성하여 사용: 데이터를 파일에 저장하는 기능이 없기 때문
            out = new ObjectOutputStream(new FileOutputStream("object.dat"));
            out.writeObject(list);  out.flush();  out.close(); }
}
```

객체 입출력 보조 스트림

```
import java.io.*;
public class ClassIODemo {
    public static void main(String[] args) throws IOException, ClassNotFoundException{
        ObjectOutputStream out = null;    ObjectInputStream in=null;
        out = new ObjectOutputStream(new FileOutputStream("object.dat"));
        PetRecord one = new PetRecord("해바라기", 7, 10.2);
        PetRecord two = new PetRecord("금강초롱", 4, 4.2);
        out.writeObject(one);    out.writeObject(two);  //파일에 객체 저장
        in = new ObjectInputStream(new FileInputStream("object.dat"));
        while(true){
            PetRecord obj=(PetRecord)in.readObject();  //파일에서 객체를 읽어 옴
            if(obj == null ) break;                    //파일에서 마지막 데이터를 읽으면
            System.out.println(obj); }                  //파일에서 읽어 온 객체 출력
        }
        out.close();
        in.close();
        System.out.println("프로그램 종료");
}}
```