

# Chapter 03. 자바 IO

**Originally made by Prof. Hanku Lee**

**Modified by Mingyu Lim**

**Collaborative Computing Systems Lab.  
School of Internet & Multimedia Engineering  
Konkuk University, Seoul, Korea**

# 1. 자바 IO 란?

## □ 자바 IO 란

- 자바 IO 프로그램이란 : 자바를 통한 입출력 프로그래밍을 의미
- 입출력 장치의 예
  - 파일 디스크(파일)
  - 키보드, 모니터, 마우스
  - 메모리
  - 네트워크
- Java.io 패키지 개요
  - 자바 IO 와 관련된 기본적인 클래스들을 제공
  - 자바의 입출력은 모든 하드웨어에 독립적으로 설계
  - 자바는 입출력을 스트림(stream)으로 처리
    - : 순서가 있는 일련의 데이터를 의미하는 추상적인 개념
  - 사용자는 스트림을 이용하여 입출력 수행
    - : 스트림을 이용하여 실제 다양한 하드웨어와 입출력을 수행하는 일은 JVM에 의해 실행

## 2. 재사용되기 위해서 설계된 Java IO

- 객체 지향 프로그래밍의 목적

객체를 재사용 함으로써 생산성을 향상 시키는것

- 자바 IO 객체들은 서로가 협력해서 원하는 작업을 처리하도록 설계 되었다.

□ 다음장에 제시되어진 예제에 대한 출력 결과를 생각해 보세요.

ACHROMATIC COLOR  
\*\*\*\*\*

## 2. 재사용되기 위해서 설계된 Java IO(계속)

```
InheritanceTest.java

public class InheritanceTest {

    public static void main(String[] args) {
        FirstChild fc = new FirstChild();
        System.out.println(fc.read());

        SecondChild sc = new SecondChild();
        System.out.println(sc.read());

        ThirdChild tc1 = new ThirdChild(fc);
        System.out.println(tc1.read());

        ThirdChild tc2 = new ThirdChild(sc);
        System.out.println(tc2.read());
    }

    class Parent{
        public String read(){
            return "Parent 입니다.";
        }
    }

    class FirstChild extends Parent{
        public String read(){
            return super.read() + ": firstChild";
        }
    }

    class SecondChild extends Parent{
        public String read(){
            return super.read() + ": secondChild";
        }
    }

    class ThirdChild extends Parent{
        Parent p;

        public ThirdChild(Parent p){
            this.p = p;
        }

        public String read(){
            return p.read() + ": thirdChild";
        }
    }
}
```

### 실행 결과

```
Problems @ Javadoc Declaration Console
<terminated> InheritanceTest [Java Application] C:\WProg
Parent 입니다.: firstChild
Parent 입니다.: secondChild
Parent 입니다.: firstChild: thirdChild
Parent 입니다.: secondChild: thirdChild
```

## 2. 재사용되기 위해서 설계된 Java IO(계속)

### □ 상속과 오버라이딩에 대한 다른 예제 풀기(예제 3-2)

```
Java - 3장/src/ChildTest.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help

P InheritanceTest.java ChildTest.java
src
  (default)
  Bu
  Ct
  Inl
  JRE Syst

class Parent2{
    int i = 7;
    public int get(){
        return i;
    }
}

class Child2 extends Parent2{
    int i = 5;
    public int get(){
        return i;
    }
}

public class ChildTest{
    public static void print(Parent2 p){
        System.out.println(p.i);
        System.out.println(p.get());
    }

    public static void main(String args[]){
        Parent2 p = new Parent2();
        System.out.println("----- 1 -----");
        System.out.println(p.i);
        System.out.println(p.get());

        Child2 c = new Child2();
        System.out.println("----- 2 -----");
        System.out.println(c.i);
        System.out.println(c.get());

        Parent2 p2 = new Child2();
        System.out.println("----- 3 -----");
        System.out.println(p2.i);
        System.out.println(p2.get());

        System.out.println("----- 4 -----");
        print(c);
        print(p2);
    }
}
```

### 실행 결과

```
Problems @ Javadoc Declaration Cor
<terminated> ChildTest [Java Application] C:\WPro
----- 1 -----
7
7
----- 2 -----
5
5
----- 3 -----
7
5
----- 4 -----
7
5
```

## 2. 재사용되기 위해서 설계된 Java IO(계속)

### □ 예제 소스 코드 분석(예제 3-2)

•Child2 클래스는 Parent2 클래스를 상속 받았다.

```
class Child2 extends Parent2{  
    int i = 5;           //부모 클래스에 있는 필드와 변수명이 같다  
    public int get(){    //부모 클래스에 있는 메소드명과 형태가 같다  
        return i;  
    }  
}
```

- Child2 클래스의 필드와 메소드가 Parent2 와 같다.

⇒ Child2는 Parent2의 필드와 메소드를 **오버라이딩** 하고있다.

## 2. 재사용되기 위해서 설계된 Java IO(계속)

### □ 예제 소스 코드 분석(예제 3-2)

```
Parent2 p = new Parent2();  
System.out.println("----- 1 -----");  
System.out.println(p.i);  
System.out.println(p.get());  
  
Child2 c = new Child2();  
System.out.println("----- 2 -----");  
System.out.println(c.i);  
System.out.println(c.get());
```

### ➡ 출력결과

```
----- 1 -----  
7  
7  
----- 2 -----  
5  
5
```

## 2. 재사용되기 위해서 설계된 Java IO(계속)

### □ 예제 소스 코드 분석(예제 3-2)

```
Parent2 p2 = new Child2();  
System.out.println("----- 3 -----");  
System.out.println(p2.i);  
System.out.println(p2.get());
```

- 힙 메모리에 올린 객체는 Child2 / Child2 객체를 참조하는 변수 형식은 Parent2 이다.
- Child2 객체가 메모리에 인스턴스 되면 Child2 / Parent2 에 있는 필드 i 2개가 메모리에 올라감.
- 객체를 가르키는 참조변수가 Parent2 라면 필드는 Parent2 것을 사용
- 메소드가 오버라이딩되었을 경우는 부모 메소드의 코드는 사라지고 자식 메소드의 코드만 남게 된다.(다형성의 개념)



## 2. 재사용되기 위해서 설계된 Java IO(계속)

### □ 예제 소스 코드 분석(예제 3-2)

- 부모는 자식을 가리킬 수 있다. 조상은 자손을 가리킬 수 있다.
- 만약, 자식이나 자손이 메소드를 오버라이딩 하고 있으면, 메소드의 기능은 자식이나 자손이 구현한 것을 따른다.

### □ 다형성 개념(중요)

- 필드는 무조건 상위 클래스의 필드만 사용할 수 있다.
- 메소드도 상위 클래스의 메소드만 사용할 수 있는데 오버라이딩이 되었다면 하위 클래스의 메소드가 실행된다.

## 2. 재사용되기 위해서 설계된 Java IO(계속)

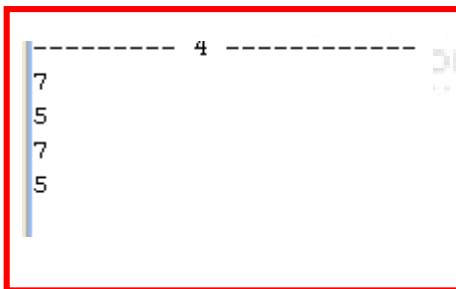
### □ 예제 소스 코드 분석(예제 3-2)

```
public static void print(Parent2 p){
    System.out.println(p.i);
    System.out.println(p.get());
}

public static void main(String args[]){
    .....

    System.out.println("----- 4 -----");
    print(c);
    print(p2);
}
```

### ➡ 출력결과



```
----- 4 -----
7
5
7
5
```

### 3. 특수한 IO 객체

#### □ 표준 입출력 스트림

- 스트림 중에서 기본적으로 가장 많이 사용되는 스트림
- java.lang 패키지의 System 클래스에 스태틱 멤버로 선언 되어 있음
- 자동으로 초기화 된다.

#### □ 특수한 IO 객체

```
public static PrintStream out // 표준 입력
public static InputStream in // 표준 출력
public static PrintStream err // 표준 오류 출력
```

- 표준 입력이란 보통 키보드의 입력, 표준 출력은 화면출력, 표준 오류 출력은 화면상의 오류 메시지 출력을 의미

### 3. 특수한 IO 객체(계속)

#### □ 표준 입출력 스트림

- System.out

- System.out.println("에러메시지");  
: 콘솔 화면에 문자열을 출력하기 위한 용도

- System.in

- int d = System.in.read(); **//한 바이트 읽어내기**  
: 키보드의 입력을 받아들이기 위해서 사용하는 표준 입력 스트림  
: 키보드에 문자를 입력한 후 엔터키를 눌렀을 때 System.in으로 문자들이 들어오며 read()를 이용해서 한 바이트씩 읽어낸다.

- System.err

- System.err.println("데이터")  
: out과 같은 PrintStream이지만 에러 메시지의 출력을 위해서 사용하는 표준 에러 스트림(Standard Error Stream)

## 4. 자바 IO 클래스

### □ 문자 스트림 VS 바이트 스트림 (4장 구체적 설명)

- 문자 스트림 클래스

- 16비트 문자나 문자열을 읽고 쓰기 위한 스트림
- Reader, Writer 클래스와 그 하위 클래스를 이용

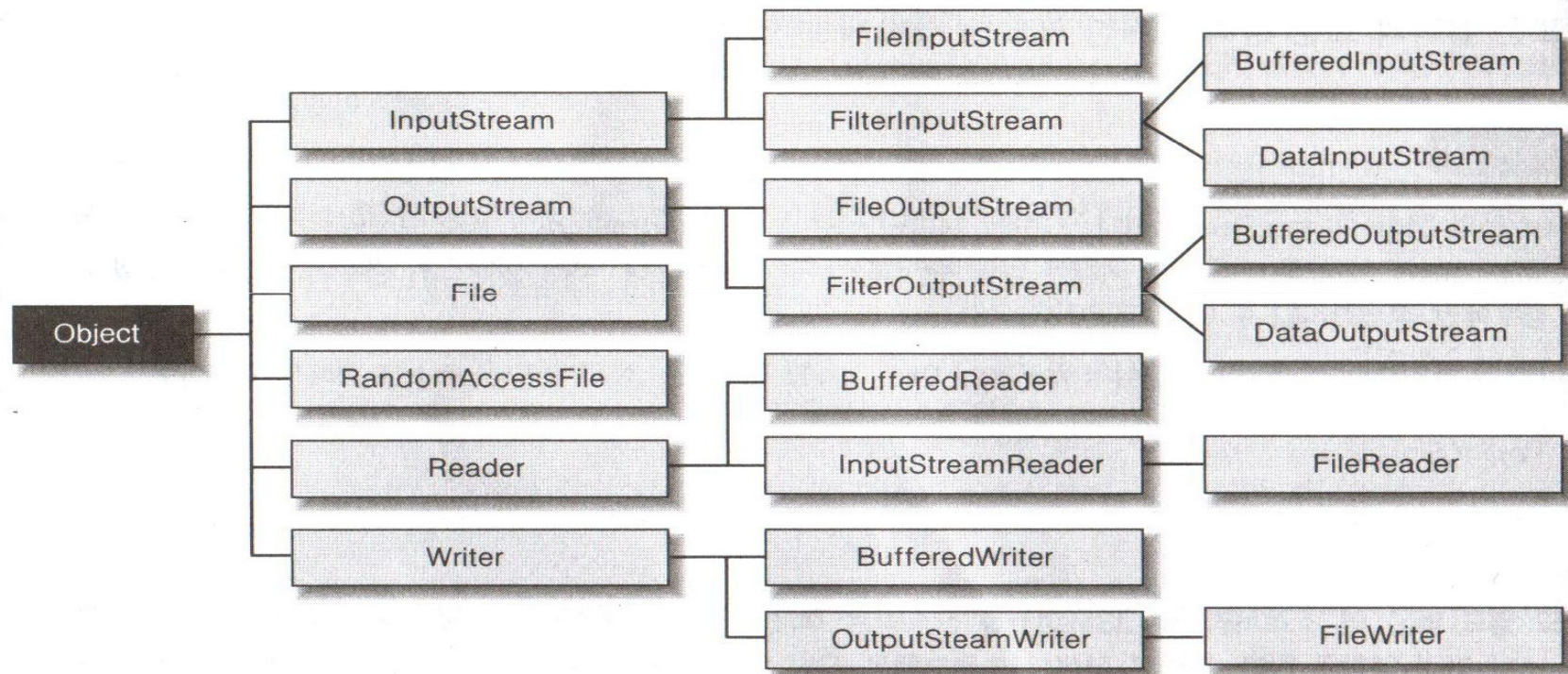
- 바이트 스트림 클래스

- 바이트(8비트)를 읽고 쓰기 위한 스트림
- InputStream, OutputStream 클래스와 그 하위 클래스를 이용

- 자바가 처음 등장 했을때에는 바이트 단위로 읽고 쓸 수 있는 바이트 스트림 클래스만 존재
- 2바이트 문화권인 한국, 일본, 중국 등의 영향으로 문자 스트림 클래스가 추후 추가

## 4. 자바 IO 클래스(계속)

### □ 자바 IO 클래스의 상속도



[그림 3-2] 자바 IO 클래스의 상속도



## 4. 자바 IO 클래스(계속)

### □ 자바 IO 클래스명에 사용된 단어의 의미

클래스에 사용된 단어	의미
Stream으로 끝나는 클래스	바이트 단위 IO 클래스이다
InputStream으로 끝나는 클래스	바이트 단위로 입력받는 클래스이다.
OutputStream으로 끝나는 클래스	바이트 단위로 출력하는 클래스이다.
Reader 로 끝나는 클래스	문자단위로 입력받는 클래스이다.
Writer로 끝나는 클래스	문자단위로 출력하는 클래스이다.
File로 시작할 경우(File 클래스제외)	파일로부터 입력이나 출력하는 클래스이다.
ByteArray로 시작할 경우	입력 클래스의 경우, 바이트 배열로부터 읽어 들이고, 출력클래스의 경우 클래스 내부의 자료구조에 출력한 후 결과를 바이트 배열로 반환하는 기능이 있다.
CharArray로 시작할 경우	입력 클래스의 경우, char 배열로부터 읽어 들이고, 출력클래스의 경우 클래스 내부의 자료구조에 출력한 후 결과를 char 배열로 반환하는 기능이 있다.
Filter로 시작할 경우	Filter로 시작하는 IO클래스는 직접 사용하는 것보다는 상속을 받아 사용하며 사용자가 원하는 내용만 필터링할 목적으로 사용된다.
Data로 시작할 경우	다양한 데이터 형식을 입출력할 목적으로 사용한다. 특히 기본형 값(int, float, Double등)을 출력하는데 유리하다.
Bufferde로 시작할 경우	프로그램에서 Buffer라는 말은 메모리를 의미한다. 입출력 시에 병목현상을 줄이고 싶을 경우에 사용한다.
RandomAccessFile	입력이나 출력을 모두 할 수 있는 클래스로써, 파일에서 임의 위치의 내용을 읽거나 쓸수 있는 기능을 제공한다.

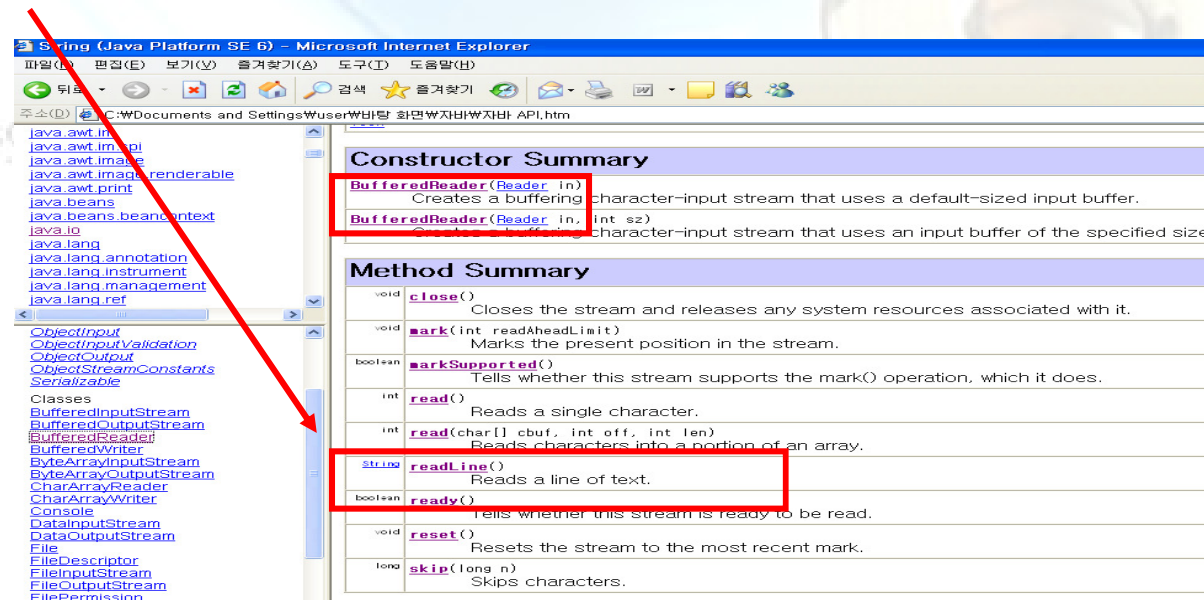
## 5. 자바 IO 프로그래밍을 잘하려면(계속)

### □ 예제를 통한 자바 IO 프로그래밍

“키보드로부터 한 줄씩 입력받아 화면에 한줄씩 출력하는 프로그램”

#### • 필요한 객체 예상

- 키보드로 입력받는 것은 표준입력 장치 : System.in 사용
- 화면에 출력하려면 표준 출력 장치 : System.out 사용
- **메소드** 필요 : “한 줄씩 입력받는다.”





## 5. 자바 IO 프로그래밍을 잘하려면(계속)

- 사용할 객체 3가지
  - BufferedReader : readLine() 이라는 한줄을 입력받는 메소드
  - System.in : 키보드로부터 입력 받으려면 사용 (InputStream 형식)
  - System.out : 화면에 출력하는데 사용
- BufferedReader 객체 메모리에 올리기
  - BufferedReader 클래스의 생성자는 기본 생성자가 존재 하지 않는다.
  - BufferedReader 클래스의 생성자를 사용하려면 Reader 객체를 인자로 지정해야한다.

```
BufferedReader br = new BufferedReader(Reader 객체나 Reader의 자손)
```

- Reader 클래스는 abstract 클래스(new 연산자이용 생성불가능)  
: **BufferedReader** 클래스 생성자 인자로는 **Reader** 클래스를 상속받는 클래스만 올수있다.

## 5. 자바 IO 프로그래밍을 잘하려면(계속)

### ★ Reader 클래스의 직계자손 클래스

java.io

**Class Reader**

[java.lang.Object](#)

└ [java.io.Reader](#)

**All Implemented Interfaces:**

[Closeable](#), [Readable](#)

**Direct Known Subclasses:**

[BufferedReader](#), [CharArrayReader](#), [FilterReader](#), [InputStreamReader](#), [PipedReader](#), [StringReader](#)

- **BufferedReader** : 중복해서 사용할 필요가 없음
- **CharArrayReader** : CharArray 단어가 붙은 클래스는 문자 배열에게 읽거나 쓰기때문에 사용불가
- **InputStreamReader** : Reader 의 자식 클래스로서 BufferedReaer 생성자의 인자로 전달 가능

## 5. 자바 IO 프로그래밍을 잘하려면(계속)

- BufferedReader 객체 생성하는 방법

```
InputStreamReader isr = new InputStreamReader(System.in)
BufferedReader br = new BufferedReader(isr)
```

- InputStreamReader의 객체를 생성하려면 인자로 InputStream 을 지정
- System.in 이 InputStream 이기때문에 생성자의 인자로 사용가능

### Constructor Summary

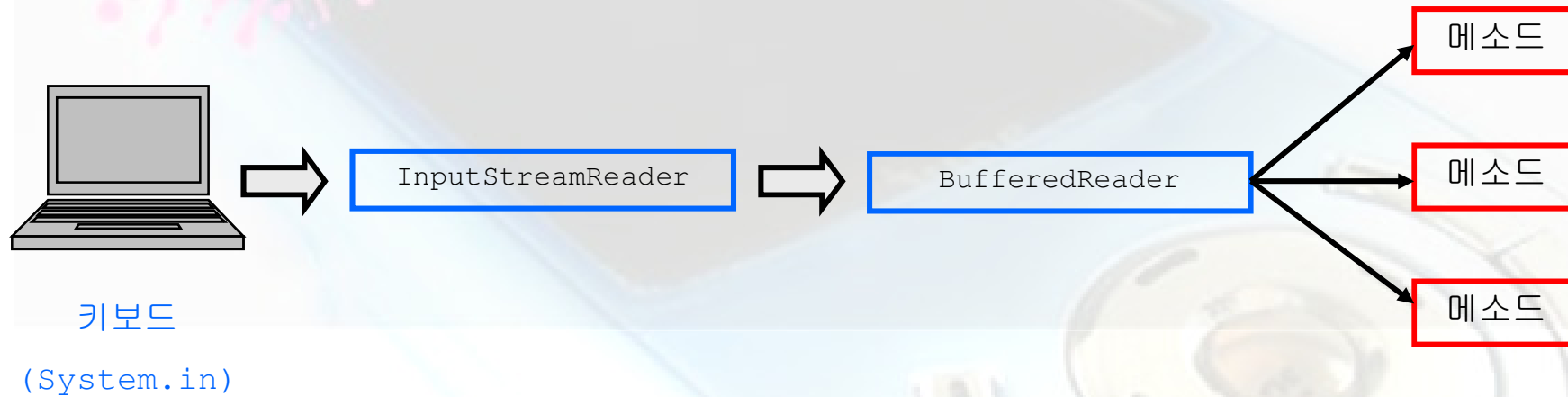
**InputStreamReader**(InputStream in)  
Creates an InputStreamReader that uses the default charset.

**InputStreamReader**(InputStream in, Charset cs)  
Creates an InputStreamReader that uses the given charset.

**InputStreamReader**(InputStream in, CharsetDecoder dec)  
Creates an InputStreamReader that uses the given charset decoder.

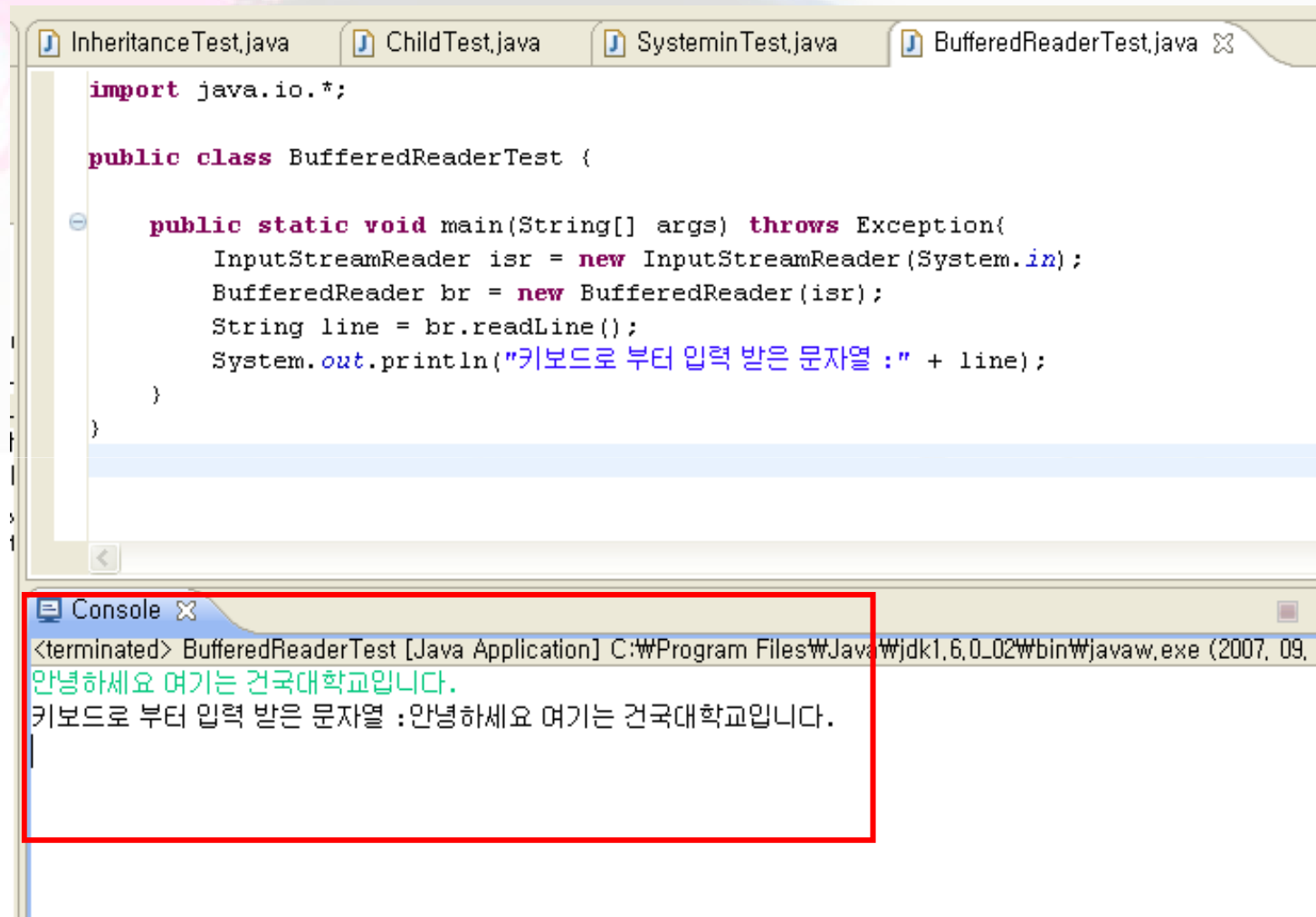
**InputStreamReader**(InputStream in, String charsetName)  
Creates an InputStreamReader that uses the named charset.

## 5. 자바 IO 프로그래밍을 잘하려면(계속)



InputStreamReader는 System.in 으로부터 읽어 들이고 BufferedReader는 InputStreamReader로 부터 읽어 들인다는 것을 의미

## 5. 자바 IO 프로그래밍을 잘하려면(계속)



The screenshot shows a Java IDE with four tabs: InheritanceTest.java, ChildTest.java, SystemInTest.java, and BufferedReaderTest.java. The BufferedReaderTest.java tab is active, displaying the following code:

```
import java.io.*;

public class BufferedReaderTest {

    public static void main(String[] args) throws Exception{
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);
        String line = br.readLine();
        System.out.println("키보드로 부터 입력 받은 문자열 : " + line);
    }
}
```

Below the code editor is a console window titled "Console". It contains the following output:

```
<terminated> BufferedReaderTest [Java Application] C:\Program Files\Java\jdk1.6.0_02\bin\javaw.exe (2007. 09. 1
안녕하세요 여기는 건국대학교입니다.
키보드로 부터 입력 받은 문자열 :안녕하세요 여기는 건국대학교입니다.
```