

Java I/O 프로그래밍 보조교재

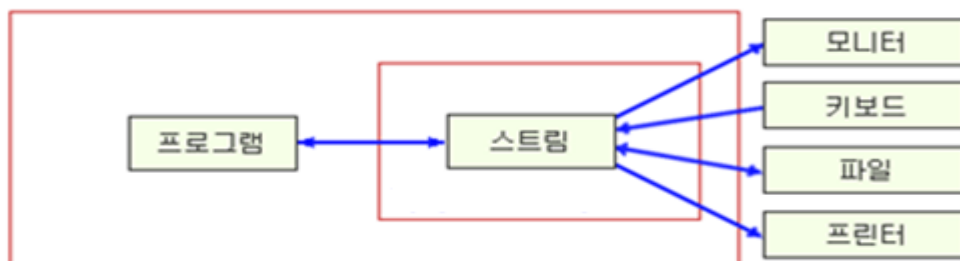
프로그램이 수행하는데 있어서 필요한 데이터를 외부로 부터 읽는 것을 입력, 프로그램의 수행 결과를 외부로 내보내는 것을 출력이라고 한다. 다만 읽는 입력받는 대상과 출력하는 대상에 어디냐에 따라서 사용되는 API가 조금씩 달라지며 Java에서는 java.io. 패키지에서 입출력 기능의 다양한 API들을 제공하고 있다.

java.io 패키지

java.io패키지는 입출력 기능의 API들과 파일을 생성하거나 파일에 대한 정보를 점검하는 기능의 API를 제공하고 있으며 입출력 하는 단위와 입출력 대상에 따라서 다양한 인터페이스와 클래스로 구성되어 있다. java.io 패키지의 API들 중에서 스트림을 이용하여 파일에 대한 입출력을 처리하는 기능의 클래스들인 `FileReader`, `FileWriter`, `FileInputStream`, `FileOutputStream` 클래스와 파일 정보를 점검하거나 파일을 생성 삭제하는 기능을 제공하는 `File` 클래스와 기능과 사용 방법에 대하여 소개한다.

■ 스트림 입출력

Java에서 플랫폼 독립적인 데이터 입출력을 지원하기 위해 스트림이라는 추상적인 구조를 사용한다. 스트림이란 순서가 있고 길이가 정해져 있지 않은 일련의 데이터 흐름을 의미하는 추상적인 개념으로서 프로그램과 입출력 장치 사이에서 입출력 자료들을 중계하는 역할을 담당한다. 즉 데이터를 입력 받거나 출력하려면 먼저 스트림에 일련의 바이트 문자들을 기록한 다음 스트림으로부터 데이터를 읽거나 특정 장치에 데이터를 출력하는 것이다.



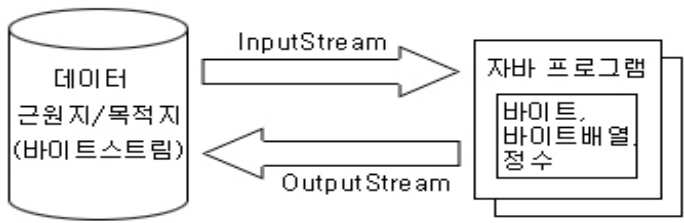
스트림을 이용하여 실제 다양한 하드웨어와 입출력을 수행하는 일은 JVM에 의해 실행되며 데이터의 근원지나 목적지에 상관없이 항상 일관된 방법으로 프로그램을 작성할 수 있다.

스트림은 우선 데이터가 흘러가는 방향에 따라 구분이 되는데 스트림에서 데이터를 읽어서 프로그램으로 가져오는 경우는 **입력 스트림(input stream)**이 사용되고 사용하고 있는 프로그램에서 해당 장치와 관련된 스트림으로 출력하는 경우 **출력 스트림(output stream)**이라 한다.

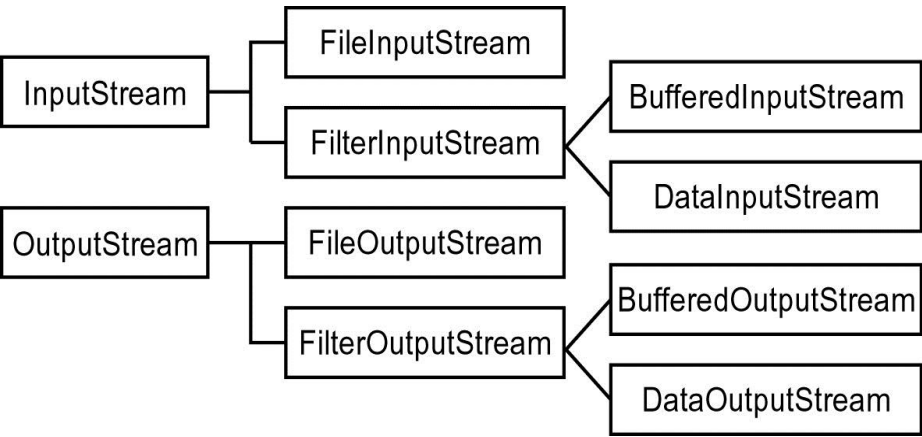
또한 처리 방식(I/O하는 단위)에 따라 **바이트 스트림(byte stream)**과 **문자 스트림(character stream)**으로 나눌 수 있다.

▶ 바이트 스트림

바이트 스트림은 byte타입, byte타입 배열, 기본형 데이터 그리고 이진 파일 형식의 객체형 데이터의 입출력을 처리할 수 있다. 1바이트 단위로 입출력을 처리한다.



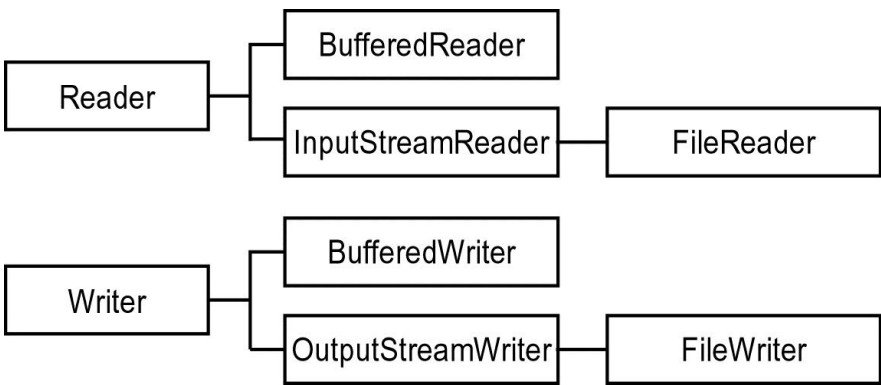
다음 제시된 API들이 바이트 스트림을 사용하는 주요 API이다.



▶ 문자 스트림

문자 스트림은 char 타입, char 타입 배열과 같은 문자형 데이터와 문자열 데이터의 입출력을 처리할 수 있다. 영어 이외의 문자에 대한 처리와 문자 인코딩을 내부에서 처리하며 유니코드를 지원하는 Java 특성에 맞게 2바이트 크기로 입출력을 수행한다.

다음 제시된 API들이 문자 스트림을 사용하는 주요 API이다.



► FileReader와 FileWriter

문자스트림을 사용하여 파일에 대한 입출력을 처리할 수 있는 클래스로서 텍스트 파일을 오픈하여 입출력을 하려는 경우에 사용된다. FileReader와 FileWriter는 파일에 저장된 바이트를 유니코드 문자로 변환해서 읽어들이거나 출력할 유니코드 문자를 디폴트 문자 인코딩의 바이트로 변환해서 파일에 저장하는 데 사용되는 입출력 클래스이다.

FileReader는 파일을 입력 모드로 오픈하여 입력을 위한 스트림을 생성하는 클래스로 다음은 FileReader의 주요 생성자 메서드이다. 생성자의 파라미터로 지정된 파일이 존재하지 않는 경우 FileNotFoundException이 발생한다.

FileReader의 생성자	설 명
FileReader(String filepath) throws FileNotFoundException	filepath로 지정한 파일에 대한 입력 스트림을 생성한다.
FileReader (File fileObj) throws FileNotFoundException	fileObj로 지정한 파일에 대한 입력 스트림을 생성한다.

FileWriter는 파일로 데이터를 출력하기 위한 출력 스트림을 오픈하는데 사용된다.

FileReader와 마찬가지로 출력할 파일에 대한 정보를 인자로 하는 생성자 함수를 갖는다. 출력할 목적으로 파일을 오픈할 때 파일이 존재하지 않으면 새로이 생성된다.

FileWriter의 생성자	설 명
FileWriter(String filepath) throws IOException	filepath로 지정한 파일에 대한 출력 스트림을 생성한다.
FileWriter(String filepath, boolean append) throws IOException	지정한 파일로 출력 스트림을 생성한다. append 인자로 출력할 때 파일에 대한 append 모드를 설정한다.
FileWriter(File fileObj) throws IOException	fileObj로 지정된 파일에 대한 출력 스트림을 생성한다.

다음은 파일을 출력 모드로 오픈하여 문자 데이터, 문자 타입 배열 그리고 문자열 데이터를 출력하는 부분예제이다.

```
FileWriter writer = null;
try {
    writer = new FileWriter("test.txt");
    char arr[] = { '객', '체', '지', '향', '언', '어', 'J', 'a', 'v', 'a' };
    for (int cnt = 0; cnt < arr.length; cnt++)
        writer.write(arr[cnt]);           // write(char) 호출
    writer.write('\n');
    writer.write(arr);                     // write(char[]) 호출
    writer.write('\n');
```

```

        writer.write("OCJP 시험 대비");           // write(String) 호출
    } catch (IOException ioe) {
        System.out.println("파일로 출력할 수 없습니다.");
    } finally {
        try {
            writer.close();
        } catch (Exception e) { }
    }
}

```

다음은 파일을 입력 모드로 오픈하여 한 문자씩 문자 데이터를 읽는 부분예제이다.

```

FileReader reader = null;
try {
    reader = new FileReader("test.txt");
    while (true) {
        int data = reader.read();
        if (data == -1)           // 더 이상 읽을 데이터가 없으면 -1을 리턴
            break;
        char ch = (char) data;
        System.out.print(ch);
    }
} catch (FileNotFoundException fnfe) {
    System.out.println("파일이 존재하지 않습니다.");
} catch (IOException ioe) {
    System.out.println("파일을 읽을 수 없습니다.");
} finally {
    try {
        reader.close();
    } catch (Exception e) { }
}

```

■ Scanner를 사용한 파일 내용 읽기

Java SE 5.0부터 추가된 Scanner클래스는 java.util패키지에서 제공되는 클래스로서 12차시에서 학습한 내용과 같이 문자열의 내용을 원하는 패턴으로 분리하여 토큰형태로 읽는 기능 외에도 표준 입력 스트림 또는 읽기 모드로 오픈된 파일로부터 원하는 토큰 단위로 데이터를 읽는 기능을 제공한다. 다음은 Scanner클래스의 주요 생성자 메서드 정보이다.

```

Scanner(File source)           // 오픈된 파일에서 스캔하고자 할 때
Scanner(InputStream source)    // 표준 입력 스트림(System.in)에서 스캔하고자 할 때
Scanner(Readable source)       // 문자 스트림으로부터 스캔하고자 할 때
Scanner(String source)         // 문자열의 내용에서 스캔하고자 할 때

```

Scanner객체를 통해서 데이터를 읽어들이는 것을 스캔한다고 한다. 이 때는 기본 분리자(공백), 단순 분리자(하나의 문자 또는 워드) 그리고 정규 패턴("WW.WWs*")을 적용하여 토큰이라고 불리는 단위로 읽게 된다. 다음은 Scanner클래스에서 제공되는 메서드들로서 데이터를 분리하여 각각의 기본형 데이터 또는 문자열형 데이터로 텍스트를 스캔하여 리턴하는 주요 메서드들이다.

```

String next()
    스케너에서 다음 토큰을 찾아서 리턴한다.

String next(Pattern pattern)
    지정된 패턴에 매칭되는 다음 토큰을 찾아서 리턴한다.

String next(String pattern)
    지정된 문자열에 의해 구성된 패턴에 매칭되는 다음 토큰을 찾아서 리턴한다.

double nextDouble()
    다음 토큰을 double타입으로 리턴한다.

int nextInt()
    다음 토큰을 int타입으로 리턴한다.

long nextLong()
    다음 토큰을 long타입으로 리턴한다.

String nextLine()
    행 구분자로 구분되는 하나의 행을 리턴한다.

```

▶ Scanner를 활용한 다양한 예제들

다음은 Scanner클래스를 이용하여 키보드로부터 데이터를 읽고 연산을 처리하는 예제이다. Java의 표준 입력 스트림 객체를 참조하는 System.in을 가지고 Scanner객체를 생성하게 되면 표준 입력 스트림을 통해 스캔하게 되는 결과가 된다. 대표적인 표준 입력 스트림은 키보드이다.

즉, 프로그램의 수행 중간에 키보드로부터 int타입의 데이터 2개, double타입의 데이터 2개를 각각 읽어서 연산을 처리하는 예제이다.

```

Scanner scanner = new Scanner(System.in);
System.out.print("두 개의 숫자(정수)를 입력해 주세요 : ");
int number1 = scanner.nextInt();
int number2 = scanner.nextInt();
System.out.println("합 : " + (number1+number2));

```

```

System.out.print("두 개의 숫자(실수)를 입력해 주세요 : ");
double number3 = scanner.nextDouble();
double number4 = scanner.nextDouble();
System.out.println("합 : " + (number3+number4));

```

다음은 Scanner클래스를 이용하여 키보드로부터 행 단위로 문자열 데이터를 읽고 표준 출력하는 예제이다. Java의 표준 입력 스트림 객체를 참조하는 System.in을 가지고 Scanner객체를 생성하게 되면 표준 입력 스트림을 통해 스캔하게 되는 결과가 된다. 대표적인 표준 입력 스트림은 키보드이다.

```

Scanner scan = new Scanner(System.in);
String input = "";
do {
    System.out.print(" 값을 입력하세요. 입력을 마치려면 Q를 입력하세요 : ");
    input = scan.nextLine();
    System.out.println("입력하신 값은 " + input + "입니다.");
} while(!input.equalsIgnoreCase("Q"));

```

다음은 Scanner 클래스를 이용하여 파일로부터 행 단위로 문자열 데이터를 읽고 읽은 문자열을 가지고 콤마(",")를 분리자로 사용하여 스캔하여 행 별로 합과 평균을 연산하고 화면에 표준 출력하는 예제이다. Scanner객체를 두 번 생성하고 있다. sc1이 참조하는 Scanner객체는 data.txt라는 파일로부터 스캔하기 위한 생성한 객체이고 sc2로 참조하는 Scanner객체는 data.txt 파일에서 행 단위로 읽어들이 문자열로부터 스캔하기 위해 생성한 객체이다.

```

Scanner sc1 = new Scanner(new File("data.txt"));
int cnt = 0;
int totalSum = 0;

while (sc1.hasNextLine()) {
    String line = sc1.nextLine();
    Scanner sc2 = new Scanner(line).useDelimiter(",");
    int sum = 0;
    while(sc2.hasNextInt()) {
        sum += sc2.nextInt();
    }
    System.out.println(line + ", sum = " + sum);
    totalSum += sum;
    cnt++;
}
System.out.println("Line: " + cnt + ", Total: " + totalSum);

```

■ File 클래스

File클래스는 파일 및 디렉터리를 관리할 수 있는 기능을 제공하는 클래스이다. 이 클래스를 이용하면 특정 파일의 존재유무, 정보, 복사, 이름변경 그리고 생성 등 파일에 관련된 다양한 작업을 수행할 수 있다. 하지만 한 가지 유의할 점은 File 클래스 자체에서는 파일의 데이터를 입출력하기 위한 메서드는 제공하지 않는다는 것이다.

Java에서는 모든 데이터의 입출력을 스트림 기반으로 수행하기 때문에 File 클래스 내에는 입출력을 위한 메서드는 제공하지 않고 있다. 주로 파일에 대한 정보를 점검하는 용도 그리고 파일을 제어하는 용도로 사용되는 클래스이다.

다음은 File클래스의 객체를 생성하는데 사용되는 주요 생성자 정보이다.

File(File parent, String child)

File객체로 지정된 부모 패스명과 문자열로 지정된 자식 패스명을 사용하여 File객체를 생성한다.

File(String pathname)

문자열로 지정된 패스명을 사용하여 File객체를 생성한다.

File(String parent, String child)

문자열로 지정된 부모 패스명과 자식 패스명을 사용하여 File객체를 생성한다.

File(URI uri)

URI 객체로 지정된 패스명을 사용하여 File객체를 생성한다.

다음은 File 클래스의 주요 메서드 리스트이다.

File 클래스의 메서드	설 명
boolean canRead()	파일을 읽을 수 있는지 여부를 리턴한다.
boolean canWrite()	파일에 기록할 수 있는지 여부를 리턴한다.
boolean createNewFile()	주어진 이름의 파일이 없으면 새로 생성한다.
boolean delete()	파일이나 폴더를 삭제한다. 단, 폴더가 비어 있지 않으면 삭제할 수 없다.
void deleteOnExit()	자바가상머신이 끝날 때 파일을 삭제한다.
boolean exists()	파일의 존재 여부를 리턴한다.
File getAbsoluteFile()	파일의 절대 경로를 넘겨준다.
String getAbsolutePath()	파일의 절대 경로를 문자열로 넘겨준다.
String getCanonicalPath()	파일의 Canonical 경로를 문자열로 넘겨준다.
String getName()	파일이나 폴더의 이름을 넘겨준다.
String getParent()	부모 경로에 대한 경로명을 문자열로 넘겨준다.
File getParentFile()	부모 폴더를 File의 형태로 리턴한다.
String getPath()	파일의 경로를 문자열의 형태로 리턴한다.
boolean isDirectory()	폴더인지 여부를 리턴한다.

► File을 활용한 다양한 예제들

다음은 File클래스를 이용하여 명령 행 데이터로 전달된 명칭의 파일이 존재하는지의 여부와 존재한다면 디렉토리 형식의 파일인지 일반 파일인지 등의 정보를 점검하는 예제이다.

```
File che = new File(args[0]);
boolean dresult = che.isDirectory();
boolean fresult = che.isFile();

if (che.exists() == false)
    System.out.println(args[0]+" 명칭의 파일 혹은 디렉토리는 존재하지 않는다.");
else if (dresult == true)
    System.out.println(args[0]+" 명칭의 파일은 디렉토리이다.");
else if(fresult == true)
    System.out.println(args[0]+" 명칭의 파일은 일반 파일이며 크기는 "+
        che.length()+"바이트이다.");
```

다음은 File클래스를 이용하여 명령 행 데이터로 전달된 명칭의 파일이 디렉토리로 존재하는지의 여부를 점검하고 임시 파일을 생성하는 예제이다.

```
File dir = new File(args[0]);
if(!dir.exists() || !dir.isDirectory()) {
    System.out.println("유효하지 않은 디렉토리입니다.");
    System.exit(0);
}
long time = new Date().getTime();
String newFileName = "temp"+time+".tmp";

File file = new File(dir,newFileName);
if (file.createNewFile())
    System.out.println(newFileName+"명의 파일이 생성되었습니다.");
else
    System.out.println(newFileName+"명의 파일이 이미 존재합니다.");
```


표준 입출력

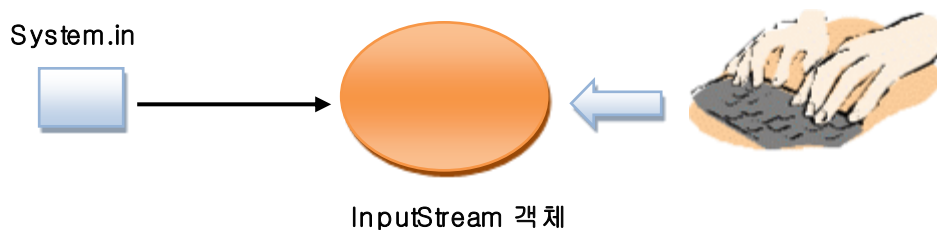
데이터는 다양한 곳에 존재할 수 있다. 데이터가 파일 일 수도 있으며, 메모리일 수도 있다. 그리고 네트워크의 특정 장소일 수도 있다. 이렇게 다양한 장소에 존재하는 데이터의 입출력(I/O)을 하기 위해서 일관된 처리 방법이 있어야 한다. Java에서는 이를 위해 스트림(Stream)이라는 구조를 제공하며 스트림을 이용하면 입출력 장치가 무엇이든 상관하지 않고 데이터를 읽거나 기록할 수 있다. 스트림 중에서 기본적으로 가장 많이 사용되는 스트림은 바로 표준 입출력 스트림이다.

■ System.in과 System.out

콘솔을 통한 데이터 입출력은 시스템에 장착되어 있는 키보드나 모니터 화면으로의 입출력을 의미하며 일반적으로 표준 입출력이라고 한다. Java에서 표준 입출력을 하려면 `java.lang.System` 클래스에서 제공되는 `in`, `out` 그리고 `err`이라는 static형 멤버 변수를 사용한다.

▶ System.in

표준 입력 스트림으로 사용되는 참조형 변수이다. JVM이 기동될 때 표준 입력 장치인 키보드로 부터의 입력을 처리할 수 있도록 `InputStream` 객체로 자동 초기화된다.



표준입력으로부터 바이트 단위로 하나의 문자 값만 읽을 때는 다음과 같이 `System.in`이 참조하는 `InputStream` 객체의 `read()`를 호출한다. `InputStream` 객체는 바이트 스트림으로 데이터를 읽어들인다.

```
int inputbyte = System.in.read();
```

표준입력으로부터 문자 단위로 하나의 문자 값만 읽을 때는 다음과 같이 `System.in`을 가지고 `InputStreamReader` 객체를 추가 생성한 후에 `read()`를 호출한다. `InputStreamReader`는 바이트 스트림으로 문자 스트림으로 변환하는 역할을 수행하는 API이다.

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

```
int inputchar = isr.read();
```

표준입력으로부터 한 행을 읽을 때는 다음과 같이 `System.in`을 가지고 `InputStreamReader` 객체를 추가 생성한 후 다시 `BufferedReader` 객체를 추가 생성한 다음에 `readLine()`이라는 메서드를 호출한다.

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

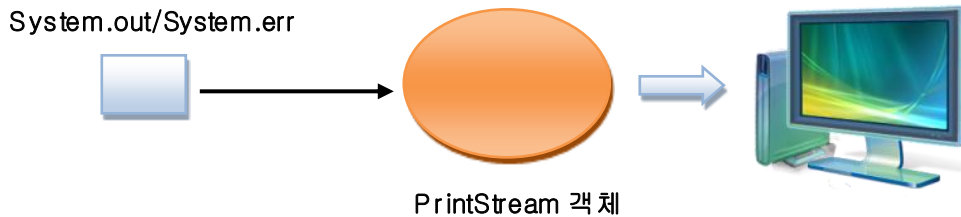
```
BufferedReader br = new BufferedReader(isr);
```

String inputline = br.readLine();

앞에서 학습한 System.in 을 가지고 Scanner객체를 생성하는 방법을 사용할 수도 있다.

▶ System.out 와 System.err

표준 출력 스트림으로 사용되는 참조형 변수이다. JVM이 기동될 때 표준 출력 장치인 모니터로 출력을 처리할 수 있도록 PrintStream 객체로 자동 초기화된다. PrintStream은 출력을 편하게 하기 위해서 제공되는 출력 전용의 바이트 스트림이다.



System.out 은 프로그램의 일반적인 수행 결과를 출력하는 스트림으로서 표준 출력 스트림이라고 하고 System.err은 오류 메시지를 출력하는 스트림으로서 표준 오류 출력 스트림이라고 한다.

수행 결과상의 차이점은 거의 없기 때문에 대부분 System.out으로 사용한다.

```
System.out.println(123.456);  
system.out.printf("%.2f", 123.456);  
system.err.println("오류 발생");
```

■ java.io.Console

java.io.Console클래스는 Java SE 6.0에서 새로 추가된 클래스로서 JVM에 의해 인식되는 콘솔 장치로의 입출력을 처리하는 기능의 클래스이다.

Console클래스가 추가되면서 Java에서의 콘솔 입출력 즉, 표준 입출력이 좀 더 편해졌다.

출력할 때는 PrintStream에서 사용했던 printf()메서드를 그대로 사용할 수 있으며 입력받을 때는 readLine()과 readPassword()를 사용할 수 있다.

readLine()메서드는 사용자가 입력한 하나의 행을 String으로 리턴하며 readPassword()메서드는 사용자가 입력한 내용을 char타입의 배열로 리턴한다.

[Console 클래스의 객체 생성]

Console클래스는 생성자를 통해서 객체를 생성할 수 없는 final클래스이다. 이 클래스의 객체를 생성하기 위해서는 다음과 같이 System클래스에서 제공되는 console()이라는 메서드를 사용한다.

```
Console console = System.console();
```

다음은 Console클래스를 활용을 보여주는 부분 예제이다.

```
Console console = System.console();    // 콘솔 클래스의 객체 생성
if (console == null) {
    System.out.println("Console을 사용할 수 없습니다.");
    System.exit(1);
}

String username = console.readLine("이름 입력 : ");
console.printf("입력된 이름 : %s\n", username);

char[] password = "mustang".toCharArray();
// readPassword()를 사용하면
// 입력되는 패스워드의 내용이 화면에 출력되지 않는다.
char[] passwordEntered = console.readPassword("패스워드 입력 : ");
if (Arrays.equals(password, passwordEntered)) {
    System.out.println("\n접근 허용 \n");
    Arrays.fill(passwordEntered, ' ');
    System.out.println("OK ...");
} else {
    System.out.println("접근 실패");
    System.exit(1);
}
```

객체직렬화

Java프로그램이 수행되는 동안 힙이라는 메모리 공간에 할당되어 있는 객체의 내용을 지속적으로 보관하고자 할 때 사용되는 기술로서 객체를 구성하는 인스턴스 멤버 변수들의 내용을 바이트 스트림으로 변환하는 것을 직렬화라 하며 반대로 바이트 스트림의 내용을 읽고 다시 객체를 만들어 내는 것을 역 직렬화라고 한다. 객체 직렬화 기능 구현하려면 java.io패키지의 API 중에서 FileInputStream, FileOutputStream, ObjectInputStream, ObjectOutputStream, 그리고 Serializable 인터페이스의 기능을 알고 있어야 한다.

■ 직렬화 가능한 객체 구현

1. **Serializable** 또는 **Externalizable** 인터페이스를 상속을 해야 한다. (NotSerializableException 발생)

모든 객체들이 직렬화 가능한 것이 아니며 java.io 패키지에서 제공되는 Serializable 또는 Externalizable을

추가 상속하고 있는 클래스의 객체만이 직렬화가 가능하다.

Externalizable은 해당 클래스 안에 직접 객체에 대한 직렬화 알고리즘을 구현하고자 할 때 상속하며 Serializable은 Java에서 기본으로 제공되는 직렬화 알고리즘을 사용하고자 할 때 상속한다. 그러므로 대부분 Serializable을 상속하게 되며 이 때는 반드시 오버라이딩 해야하는 메서드는 없고 단지 이 객체는 직렬화가 가능하다는 정도를 알리는 용도이다.

2. 조상이 Serializable을 상속하고 있으면 자손에도 그대로 적용된다.
3. 자손은 Serializable을 상속하고 있지만 조상은 Serializable을 상속하고 있지 않으면 자손에서만 직렬화가 일어난다.
4. non-static, non-transient 멤버 변수들만 직렬화 대상이 된다.
직렬화할 때는 객체의 모든 상태정보 즉, 인스턴스형 멤버변수의 값들은 직렬화 대상이 되지만 static변수(클래스 변수)는 객체의 상태 정보가 아니므로 기본적으로 직렬화 대상에서 자동으로 제외된다. 하지만 인스턴스형 멤버 변수라 하더라도 일시적으로만 사용하려는 경우 즉, 직렬화 시에 직렬화에서 제외되는 멤버변수를 선언하고자 할 때 사용되는 변수 제어자가 바로 **transient** 키워드이다.
5. 직렬화의 대상이 되는 멤버 변수가 참조형일 때는 참조하는 객체도 직렬화 가능한 객체여야 한다. 그렇지 않으면 실행 시 NotSerializableException이 발생된다.

■ ObjectOutputStream을 사용한 직렬화

ObjectOutputStream은 기본형 타입의 데이터와 객체를 직렬화하여 출력할 수 있도록 지원하는 보조스트림 클래스이다. 다음과 같은 생성자를 사용하여 객체를 생성하며 writeInt(), writeDouble(), writeChar() 그리고 writeObject() 등의 메서드를 사용하여 출력할 수 있다.

ObjectOutputStream(OutputStream out)

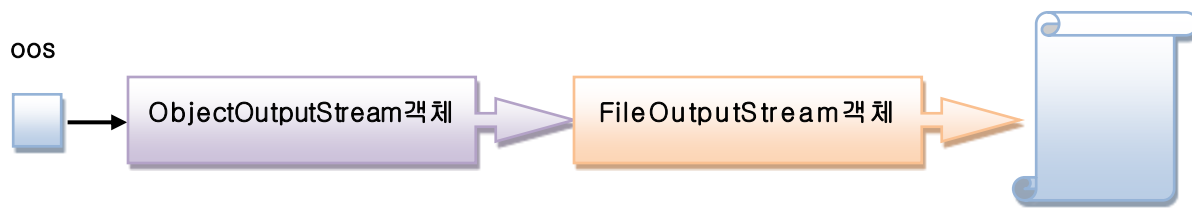
지정된 OutputStream객체로 출력하는 ObjectOutputStream객체를 생성한다.

다음은 ObjectOutputStream의 객체를 생성하는 부분 소스이다. 직렬화한 결과를 파일에 출력하고자 한다면 먼저 바이트 스트림으로 동작하는 FileOutputStream클래스의 객체를 생성하여 파일을 출력모드로 었다.

```
FileOutputStream fos = new FileOutputStream("test1.ser");
ObjectOutputStream oos = new ObjectOutputStream(fos);
oos.writeInt(100);
oos.writeDouble(3.14);
oos.writeObject(new Date());
```

다음은 위의 소스가 수행될 때의 ObjectOutputStream객체와 FileOutputStream객체의 관계를 소개하고 있는 그림

이다.



직렬화 자격을 갖추고 있지 않은 클래스의 객체를 writeObject()로 출력하면 예외가 발생한다.

■ ObjectInputStream을 사용한 역 직렬화

ObjectInputStream은 ObjectOutputStreamdp 의해 출력된 기본형 타입의 데이터와 객체를 역 직렬화하여 입력받을 수 있도록 지원하는 보조스트림 클래스이다. 다음과 같은 생성자를 사용하여 객체를 생성하며 readInt(), readDouble(), readChar() 그리고 readObject() 등의 메서드를 사용하여 출력할 수 있다.

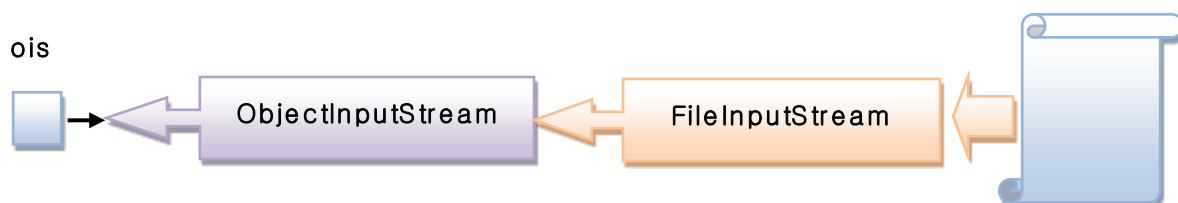
ObjectInputStream(InputStream out)

지정된 InputStream객체로부터 입력받는 ObjectInputStream객체를 생성한다.

다음은 ObjectInputStream의 객체를 생성하는 부분 소스이다. 파일에 출력된 직렬화한 결과를 읽고자 한다면 먼저 바이트 스트림으로 동작하는 FileInputStream클래스의 객체를 생성하여 파일을 입력모드로 었다.

```
FileInputStream fis = new FileInputStream("test1.ser");
ObjectInputStream ois = new ObjectInputStream(fis);
int value1 = ois.readInt();
double value2 = ois.readDouble();
Date value3 = (Date)ois.readObject();
```

다음은 위의 소스가 수행될 때의 ObjectInputStream객체와 FileInputStream객체의 관계를 소개하고 있는 그림이다.



System.out.printf() 메서드를 활용한 포맷적용 출력 (JDK 5.0)

- JDK 5.0 부터 지원
- PrintStream 의 printf() 메서드 사용
- %.. 으로 출력 포맷 지정
- Java2 에서 문자열 결합 연산자(Concatenation) + 를 이용하던 불편을 다소 해결

문법 구조

printf(format, arguments)

format : %[argument_index\$][+|-flag][width]conversion

포맷 문자열안에서 포맷 지정자에 의해
참조되어지는 매개변수 인덱스

출력되는 값들의 타입(문자타입, 정수타입,
실수타입....)을 지정

```
int a = 20;
long b = 20000L;
float c = 34.56F;
double d = 123.123;
char e = 'A';
GregorianCalendar today = GregorianCalendar.getInstance();
```

System.out.println("1 : "+a + " "+b+" "+c+" "+d+" "+ e); → JAVA2 방식

System.out.printf("2 : %d %d %f %f %c %n", a, b, c, d, e);

System.out.printf("3 : %f %1\$a %1\$e %1\$f %1\$g Wn", 32.3453453453);

System.out.printf("4 : %1\$h %1\$d %2\$f %3\$c Wn", a, d, e);

System.out.printf("5 : %1\$20d%n", a);

System.out.printf("6 : %-20d%n", a);

System.out.printf("7 : %1\$20.10f%n", 2345.0123456789);

System.out.printf("8 : 오늘의 날짜는 %1\$tY 년 %1\$tm 월 %1\$td 일 입니다.Wn", today);

Flags

Flag	Description	Example
+	Prints sign for positive and negative numbers	+3333.33
space	Adds a space before positive numbers	3333.33
0	Adds leading zeroes	003333.33
-	Left-justifies field	3333.33
(Encloses negative number in parentheses	(3333.33)
,	Adds group separators	3,333.33
# (for f format)	Always includes a decimal point	3,333.
# (for x or o format)	Adds 0x or 0 prefix	0xcafe
\$	Specifies the index of the argument to be formatted; for example, %1\$d %1\$x prints the first argument in decimal and hexadecimal	159 9F
<	Formats the same value as the previous specification; for example, %d %<x prints the same number in decimal and hexadecimal	159 9F

Conversion characters

Conversion Character	Description	Example
d	Decimal integer	159
x	Hexadecimal integer	9f
o	Octal integer	237
f	Fixed-point floating-point	15.9
e	Exponential floating-point	1.59e+01
g	General floating-point (the shorter of e and f)	
a	Hexadecimal floating-point	0x1.fccdp3
s	String	Hello
c	Character	H
b	boolean	true
h	Hash code	42628b2
tx	Date and time	See the next table
%	The percent symbol	%
n	The platform-dependent line separator	

Formatted Output with MessageFormat, continued

Format	Subformat	Example
number	none	1,234.567
	integer	1,235
	currency	\$1,234.57
	percent	123,457%
date	none or medium	Jan 15, 2009
	short	1/15/09
	long	January 15, 2009
	full	Thursday, January 15, 2009
time	none or medium	3:45:00 PM
	short	3:45 PM
	long	3:45:00 PM PST
	full	3:45:00 PM PST
choice	<p>List of choices, separated by . Each choice has</p> <ul style="list-style-type: none"> ▪ a lower bound (use -\u221E for -∞) ▪ a relational operator: < for "less than", # or \u2264 for ≤ ▪ a message format string <p>For example, {1,choice,0#no houses 1#one house 2#{1} houses}</p>	<p>no house</p> <p>one house</p> <p>5 houses</p>