13장 예외 처리와 입출력

• 학습 목표

- 예외의 개념과 예외 관련 클래스를 학습합니다.
- 두 가지 예외 처리 방법에 관해 학습합니다.
- 자바 시스템에서의 입출력 개요를 학습합니다.
- 파일, 디렉터리와 연관된 클래스를 학습합니다.
- 문자 스트림과 바이트 스트림 개요에 관해 학습합니다.
- 문자 스트림 관련 클래스에 관해 학습합니다. 문자 스트림을 통하여 파일에 입출력하는 방법을 학습합니다.
- 바이트 스트림 관련 클래스에 관해 학습합니다. 바이트 스트림을 통하여 파일에 입출력하는 방법과 기본 자료형의 입출력, 객체의 입출력하는 방법에 관해 학습합니다.

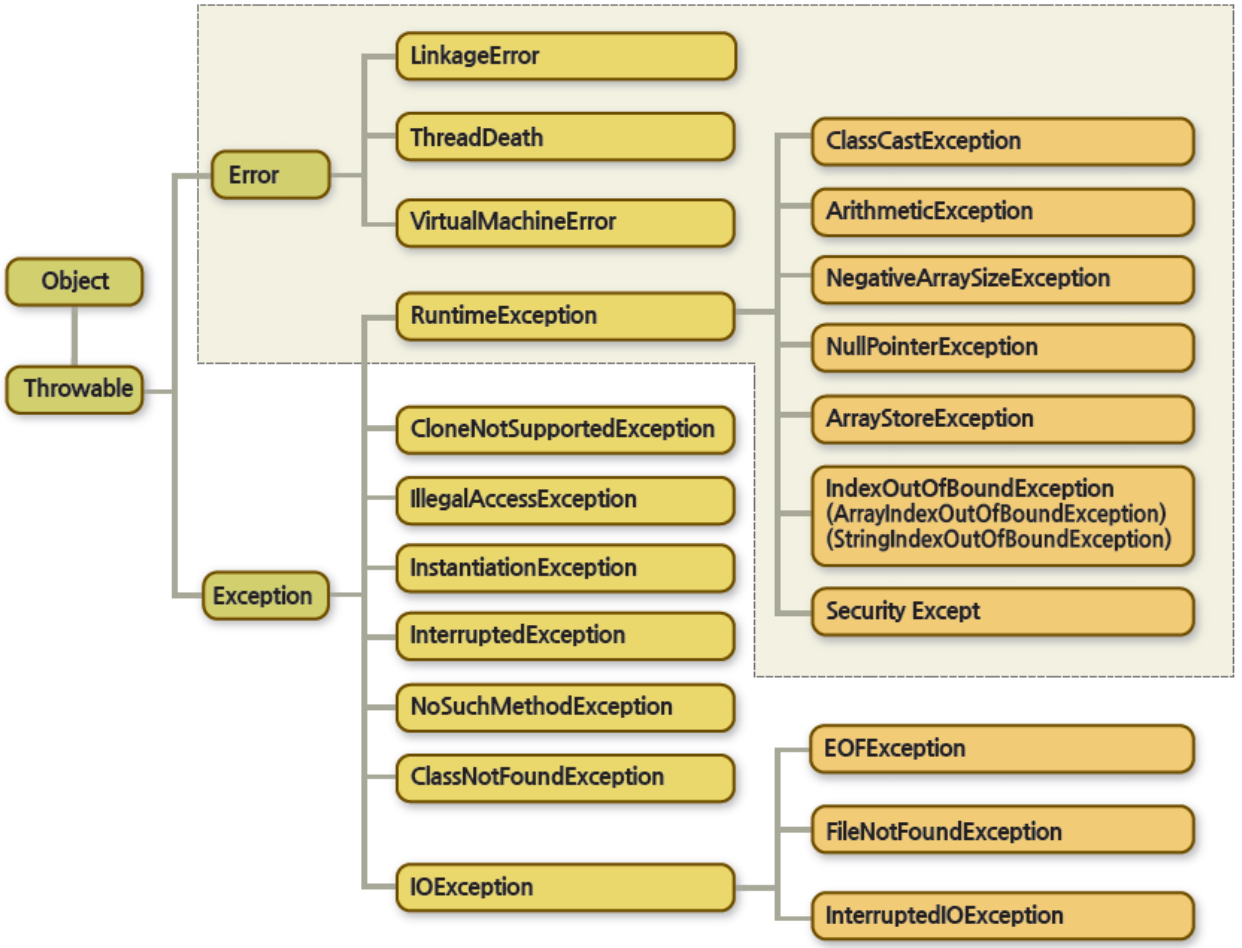
- **예외 : 프로그램 실행 중에 발생하는 예기치 않은 사건**
  - 자바에서는 발생되는 예외를 모두 객체로 취급(객체로 취급한다는 의미는 관련된 클래스가 있다는 의미)
  - 예외의 예
    - 정수를 0으로 나누는 경우
    - 배열의 첨자가 음수 값을 가지는 경우
    - 배열의 첨자가 배열의 크기를 벗어나는 경우
    - 부적절한 형 변환이 일어나는 경우
    - 입출력 시 인터럽트가 발생하는 경우
    - 입출력을 위해 지정한 파일이 없거나 파일 이름이 틀린 경우 등 많은 경우가 존재
- **자바의 뛰어난 능력 중에 하나가 바로 예외를 프로그램에서 처리할 수 있다는 점이다**

● 예외의 발생과 처리의 예(프로그램에서 예외를 처리하지 않는 경우) : JVM이 처리

```
01: class Error {
02:     public static void main(String[] args) {
03:         x();  <-------------------- x() 메소드 호출
04:     }
05:     static void x() {
06:         y();  <-------------------- y() 메소드 호출
07:     }
08:     static void y() {
09:         z();  <-------------------- z() 메소드 호출
10:     }
11:     static void z() {
12:         int i = 1;
13:         int j = 0;
14:         System.out.println(i/j);  <------ 1을 0으로 나눈다. 예외 발생
15:     }
16: }
```

```
01: Exception in thread "main" java.lang.ArithmeticException: / by zero  <----
02:     at Error.z(Error.java:14)                                              |
03:     at Error.y(Error.java:9)                                               |
04:     at Error.x(Error.java:6)                                               |
05:     at Error.main(Error.java:3)                                            |
                                                  예외가 발생한 원인과 관련된 클래스 ----
```

- 자바는 예외 관련 클래스를 java.lang 패키지에서 제공



- **자바는 RuntimeException과 관련된 예외와 Error 클래스 예외를 프로그램에서 처리하지 않습니다.**
  - 그 이유는 예외 지정에 소요되는 노력이 예외를 지정하여 얻는 효율보다 더 크기 때문. 즉 RuntimeException 클래스와 관련된 예외를 모두 처리하기 위해서는 프로그램의 작성 노력 대부분을 예외 처리에 보내야 하기 때문

- **Error 클래스 관련 예외와 RuntimeException 클래스 관련 예외는 처리 효율보다는 처리에 드는 노력이 너무 커서 자바 프로그램에서 직접 처리하지 않고 JVM에 처리를 맡긴다**
- **그러나 나머지 예외들은 자바 프로그램에서 직접 처리하여야 한다.**
  - 자바에서의 예외 처리 방법
    - 예외가 발생한 메소드 내에서 처리하는 방법
    - 발생된 예외를 자신을 호출한 메소드에 넘겨주는 방법

● 예

```
01: import java.io.*;
02: public class IOExceptionError {
03:     public static void main(String args[]) {
04:         FileReader file = new FileReader("a.txt");
05:         int i;
06:         while((i = file.read()) != -1) {
07:             System.out.print((char)i);
08:         }
09:         file.close();
10:     }
11: }
```

a.txt 파일로부터 읽어 들이기 위한 객체 생성

읽은 하나의 문자를 출력

```
01: Exception in thread "main" java.lang.Error: Unresolved compilation problems:
02:     Unhandled exception type FileNotFoundException
03:     Unhandled exception type IOException
04:     Unhandled exception type IOException
05:
06:     at IOExceptionError.main(IOExceptionError.java:4)
```

지정된 파일이 없을 경우를 대비한 예외 루틴을 기술해야 한다.

3-1 메소드에서 예외 처리

- **발생되는 예외를 메소드 내에서 직접 처리하기 위해 자바 언어는 try, catch, finally 구문을 제공**

[형식] 메소드에서의 예외 처리

```
try {
..... // try 블록
}
catch(예외타입1 매개 변수 1) {
..... // 예외 처리 블록1
}
catch(예외타입2 매개 변수 2) {
..... // 예외 처리 블록 2
}
.....
catch(예외타입N 매개 변수 N) {
..... // 예외 처리 블록 N
}
finally {
..... // finally블록
}
```

try 블록은 예외가 발생할 가능성이 있는 범위를 지정하는 블록입니다

catch 블록은 try 블록 바로 다음에 위치하여야 합니다. catch 블록과 try 블록 사이에 다른 문장이 있으면 오류가 발생

finally 블록은 필요에 따라 선택적으로 사용되며, 예외의 발생과 상관없이 무조건 수행되는 블록

3-1 메소드에서 예외 처리

예제 13.1 ExceptionTest1.java

```
01: import java.io.*;
02: public class ExceptionTest1 {
03:     public static void main(String args[]) {
04:         try {
05:             FileReader file = new FileReader("a.txt");
06:             int i;
07:             while((i = file.read()) != -1)
08:                 System.out.print((char)i);
09:             file.close();
10:         }
11:         catch(Exception e) {
12:             System.out.println("예외 처리 루틴 : " + e + " 파일이 존재하지
    않는다.");
13:         }
14:     }
15: }
```

입출력 패키지 사용을 위한 io 패키지 import

try 블록 지정

파일 객체 생성

catch 블록 지정

실행 결과

– a.txt 파일이 없는 경우 실행
예외 처리 루틴 : java.io.FileNotFoundException: a.txt (지정된 파일을 찾을 수 없습니다) 파일이 존재하지 않는다.

– a.txt 파일에 "처음 시작하는 자바를 사랑합니다"를 저장하고 실행
* a.txt 파일은 이클립스의 프로젝트명 폴더에 저장해야 합니다.
처음 시작하는 자바를 사랑합니다

3-2 호출한 메소드에 예외를 넘겨주는 방법

● 발생된 예외를 호출한 메소드에 넘겨주는 방법

- 이러한 방법은 처리해야 하는 모든 예외를 하나의 메소드에 처리하게 하거나, 자바 가상 기계(JVM)에 처리를 맡길 때 유용한 방법

[형식] 예외를 호출 메소드에 넘겨주는 방법

```
public void c() throws 예외 클래스[,예외 클래스,...]
```

```
01: import java.io.*;
02: public class ExceptionTest1 {
03:     public static void main(String args[]) throws Exception {
04:         FileReader file = new FileReader("a.txt");
05:         int i;
06:         while((i = file.read()) != -1)
07:             System.out.print((char)i);
08:         file.close();
09:         catch(Exception e) {
10:         System.out.println("예외 처리 루틴 : " + e + " 파일이 존재하지 않는다.");
11:     }
12: }
```

main 메소드에서 throws 절을 이용하여 예외를 넘김

- **프로그램에서 사용된 데이터를 영구적으로 저장하여 후에 다른 프로그램에서 사용하게 하기위해 필요한 기능이 입출력 기능**
  - 자바 언어는 입출력을 위해 스트림(stream)을 사용. 스트림이란 순서가 있는 일련의 데이터를 의미하는 추상적인 개념

그림 13-2 스트림을 통한 입출력의 예

- **자바는 입출력을 위해 스트림을 생성하고 다루는 클래스들을 java.io 패키지로 제공**
  - 이 패키지를 이용하면 사용자는 다양하고 복잡한 입출력 장치와 상관없이 일관된 방법으로 입출력을 수행할 수 있다.

- java.io 패키지

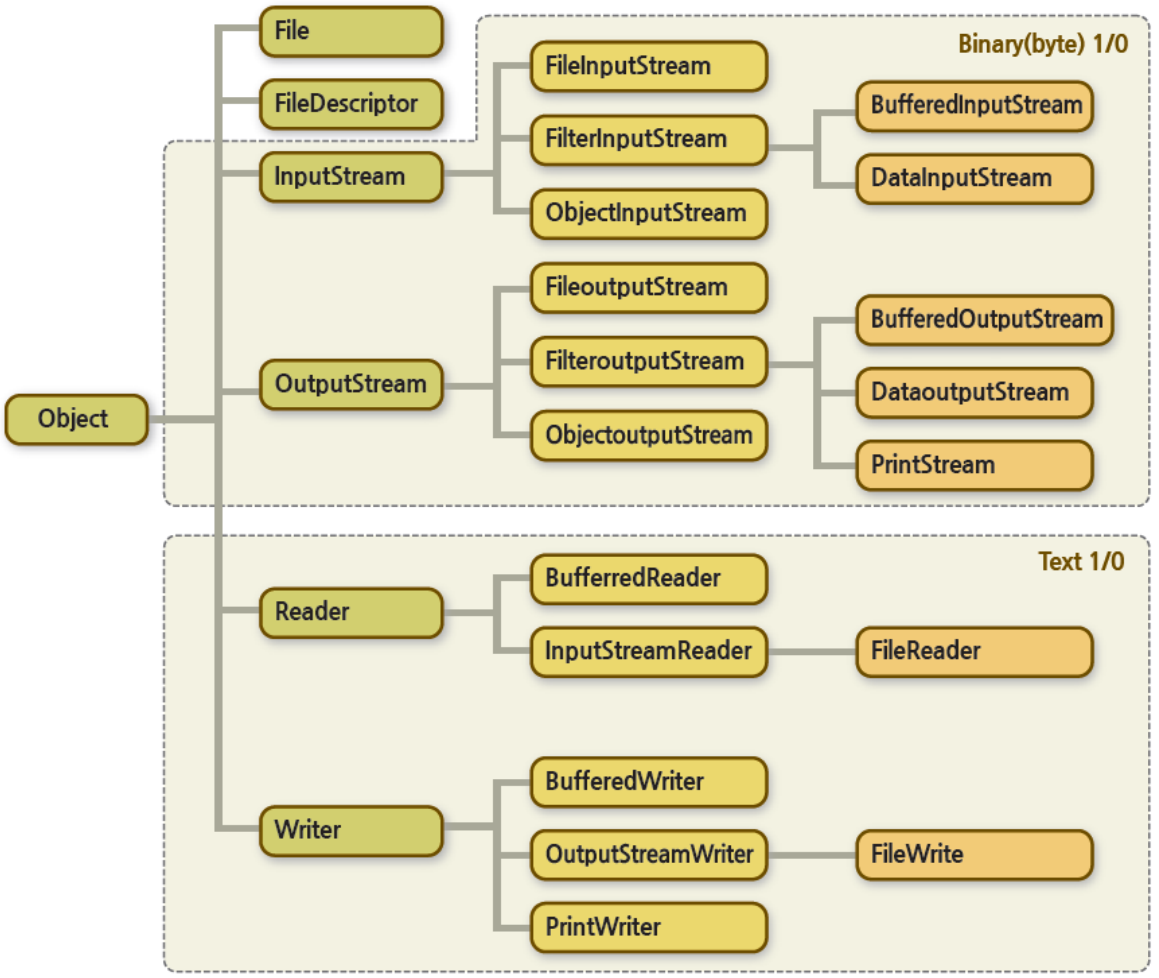


그림 13-3 java.io 패키지의 클래스 계층 구조

- 자바는 입출력을 위해 필요한 파일과 디렉터리를 다루기 위해 File 클래스를 제공

[형식] File 클래스의 생성자

```
File(String directoryPath)
File(String directoryPath, String filename)
File(File dirObj, String filename)
```

directorypath : 파일이 존재하는 디렉터리 경로 이름(파일 이름 포함)

filename : 파일의 이름

dirObj : 디렉터리 경로명을 가지고 있는 File 객체

* 경로 이름을 지정할 때에는 윈도 시스템과는 달리 역 슬래시(\)를 사용하지 않고 슬래시(/)를 사용합니다.

다음은 File 클래스의 생성자를 사용하는 예입니다.

```
01: File f1 = new File("/");
02: File f2 = new File("/", "autoexec.bat");
03: File f3 = new File(f1, "autoexec.bat");
```

파일의 경로만 가진 객체

파일의 경로와 파일 이름을 가진 객체

파일의 경로를 가진 File 객체와 파일 이름을 가진 객체

표 13-1 File 클래스의 주요 메소드

메소드 이름	설명
boolean canRead()	파일이 읽기 가능하면 true, 아니면 false를 반환
boolean canWrite()	파일이 쓰기 가능하면 true, 아니면 false를 반환
boolean createNewFile() throws IOException	File 객체를 이용하여 파일을 생성한다. 파일 생성이 생성되면 true, 아니면 flase를 반환

5 파일과 디렉터리

메소드 이름	설명
boolean equals(Object obj)	현재의 객체와 obj로 지정된 객체가 같은 파일을 가지고 있으면 true, 아니면 false를 반환
String getAbsolutePath()	파일에 대한 절대 경로를 반환
String getCanonicalPath()	파일에 대한 정규 경로를 반환
String getParent()	부모 디렉터리 이름을 반환
String getName()	파일의 이름을 반환
String getPath()	파일의 경로를 반환
boolean isAbsolute()	경로가 절대 경로이면 true, 아니면 false를 반환
boolean isDirectory()	현재의 객체가 디렉터리이면 true, 아니면 false를 반환
boolean isFile()	현재의 객체가 파일이면 true, 아니면 false를 반환
long lastModified()	1970년 1월 1일(GMT)부터 파일이 마지막으로 수정된 날짜까지의 시간을 밀리초로 반환
long length()	파일의 바이트 수를 반환
String[] list()	지정된 디렉터리에 있는 파일과 디렉터리를 문자열 배열로 반환
boolean mkdir()	디렉터리를 생성. 경로로 지정된 모든 부모 디렉터리가 존재하여야 한다. 지정한 디렉터리가 생성되면 true를 반환하고,아니면 false를 반환
boolean mkdirs()	디렉터리를 생성. 경로로 지정된 디렉터리가 존재하지 않으면 생성한 다음 지정한 디렉터리를 생성. 디렉터리가 생성되면 true를, 아니면 false를 반환
boolean renameTo(File newName)	파일이나 디렉터리의 이름을 newName으로 변경한 다음 true를 반환. 이름을 변경하지 못하면 false를 반환

● 예제 13.2

```
01: import java.io.File;
02: public class FileDirTest1 {
03:     public static void main(String args[]) {
04:         String directory = "C:/Windows";
05:         File f1 = new File(directory);
06:         if (f1.isDirectory()) {
07:             System.out.println("검색 디렉터리 " + directory);
08:             System.out.println("═══════════════════════════");
09:             String s[] = f1.list();
10:             for (int i=0; i < s.length; i++) {
11:                 File f = new File(directory + "/" + s[i]);
12:                 if (f.isDirectory())
13:                     System.out.println(s[i] + " : 디렉터리");
14:                 else
15:                     System.out.println(s[i] + " : 파일");
16:             }
17:         }
18:         else
19:             System.out.println("지정한 " + directory + " 는 디렉터리가 아님");
20:     }
21: }
```

C:₩Windows 디렉터리로 File 객체 생성

디렉터리의 모든 요소를 문자열의 배열로 생성

문자열의 항목을 가지고 File 객체 생성

디렉터리인지 파일인지를 판별하여 출력

실행 결과 Windows 디렉터리 많은 항목이 있어 출력 결과를 일부만 표시하였습니다.

```
검색 디렉터리 C:/Windows
==========================
addins : 디렉터리
AppCompat : 디렉터리
Application Data : 디렉터리
AppPatch : 디렉터리
assembly : 디렉터리
AxInstSV : 디렉터리
bfsvc.exe : 파일
BitLockerDiscoveryVolumeContents : 디렉터리
```

예제 13.3 FileDirTest2.java

```
01: import java.io.File;
02: public class FileDirTest2 {
03:     public static void main(String args[]) {
04:         File f1 = new File("C:/a.txt");
05:         System.out.println("파일 이름 : " + f1.getName());
06:         System.out.println("파일 경로 : " + f1.getPath());
07:         System.out.println("절대 경로 : " + f1.getAbsolutePath());
08:         System.out.println(f1.exists() ? "파일 존재" : "파일 없음");
09:         System.out.println(f1.canWrite() ? "수정 가능" : "수정 불가능");
10:         System.out.println(f1.canRead() ? "읽기 가능" : "읽기 불가능");
11:         System.out.println(f1.isDirectory() ? "디렉토리" : "디렉토리 아님");
12:         System.out.println(f1.isFile() ? "파일" : "파일 아님");
13:         System.out.println(f1.isAbsolute() ? "절대 경로" : "상대 경로");
14:         System.out.println("1970년 1월 1일부터 파일이 마지막 수정된 날짜까지의
    밀리초 : " + f1.lastModified());
15:         System.out.println("파일의 크기 : " + f1.length() + " Bytes");
16:     }
17: }
```

File 객체를 생성

실행 결과

```
파일 이름 : a.txt
파일 경로 : C:₩a.txt
절대 경로 : C:₩a.txt
파일 존재
수정 가능
읽기 가능
디렉토리 아님
파일
절대 경로
1970년 1월 1일부터 파일이 마지막 수정된 날짜까지의 밀리초 : 1384655700845
파일의 크기 : 31 Bytes
```

- **스트림은 입출력 데이터의 추상적인 표현**
- **스트림에는 문자 스트림과 바이트 스트림 두 가지 형태가 있다**
  - 문자 스트림은 16비트 문자나 문자열들을 읽고 쓰기 위한 스트림
  - 바이트 스트림(또는 바이너리 스트림)은 8비트의 바이트를 읽고 쓰기 위한 스트림

● 문자 스트림 관련 클래스

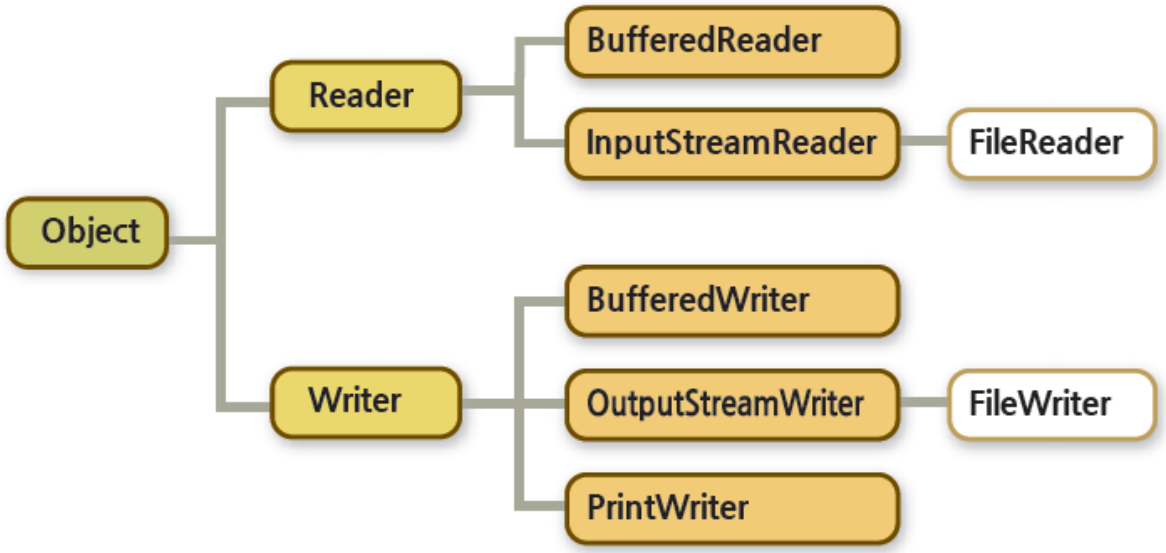


그림 13-4 문자 스트림과 연관된 클래스

7.1 Writer 클래스와 Reader 클래스

Writer와 Reader 클래스는 추상 클래스

- 이 클래스들은 문자 스트림의 입출력에 필요한 많은 메소드들을 제공

표 13-2 Writer 클래스의 주요 메소드

메소드 이름	설명
void close( )	출력 스트림을 닫는다.
void flush( )	출력 버퍼에 저장된 모든 데이터를 출력 장치로 전송
void write(int c)	c의 하위 16비트를 스트림으로 출력
void write(char buffer[ ])	buffer 배열에 있는 문자들을 스트림으로 출력
void write(char buffer[ ], int index, int size)	buffer 배열의 index 위치부터 size 크기만큼의 문자들을 스트림으로 출력
void write(String s)	문자열 s를 스트림으로 출력
void write(String s, int index, int size)	문자열의 index 위치부터 size 크기만큼의 문자들을 스트림으로 출력

7.1 Writer 클래스와 Reader 클래스

표 13-3 Reader 클래스의 주요 메소드

메소드 이름	설명
void close( )	입력 스트림을 닫는다.
int read( )	다음 문자를 읽어 반환한다. 입력 스트림에 읽을 문자가 없으면 대기한다. 읽은 문자가 파일의 끝이면 –1을 반환
int read(char buffer[ ])	입력 스트림으로부터 buffer 배열 크기만큼의 문자를 읽어 buffer에 저장
int read(char buffer[ ], int offset, int numChars)	입력 스트림으로부터 numChars에 지정한 만큼의 문자를 읽어 buffer의 offset 위치에 저장하고 읽은 문자의 개수를 반환
void mark(int numChars)	입력 스트림의 현재의 위치에 mark 한다.
boolean markSupported( )	현재의 입력 스트림이 mark()와 reset()을 지원하면 true를 반환
boolean ready( )	다음 read()문을 수행할 수 있으면 true, 입력 스트림이 없어 기다려야 되는 경우에는 false를 반환
void reset( )	입력 스트림의 입력 시작 부분을 현재의 위치에서 가장 가까운 이전의 mark 위치로 설정
int skip(long numChars)	numChars로 지정된 문자 수만큼을 스킵하고 스킵된 문자의 수를 반환

7.2 FileWriter 클래스와 FileReader 클래스

- **FileWriter 클래스는 OutputStreamWriter 클래스로부터 상속된 클래스**
  - 이 클래스는 파일에 문자를 출력하는 기능을 제공

[형식] FileWriter 클래스의 생성자

```
FileWriter(String filepath) throws IOException
FileWriter(String filepath, boolean append) throws IOException

FileWriter(File fileObj) throws IOException
```

filepath : 파일의 이름(경로명 포함)

fileObj : 특정 파일을 지정하고 있는 File 객체

append : append가 true이면 파일의 끝에 문자를 첨가하고, false이면 기존 문자 위에 겹쳐 출력한다.

7.2 FileWriter 클래스와 FileReader 클래스

- **FileReader 클래스는 InputStreamReader 클래스로부터 상속된 클래스**
  - 파일로부터 문자를 입력받기 위해 사용

[형식] FileReader 클래스의 생성자

```
FileReader(String filepath)
FileReader(File fileObj)
```

filepath : 파일의 이름(경로명 포함)

fileObj : 특정 파일을 지정하고 있는 File 객체

7.2 FileWriter 클래스와 FileReader 클래스

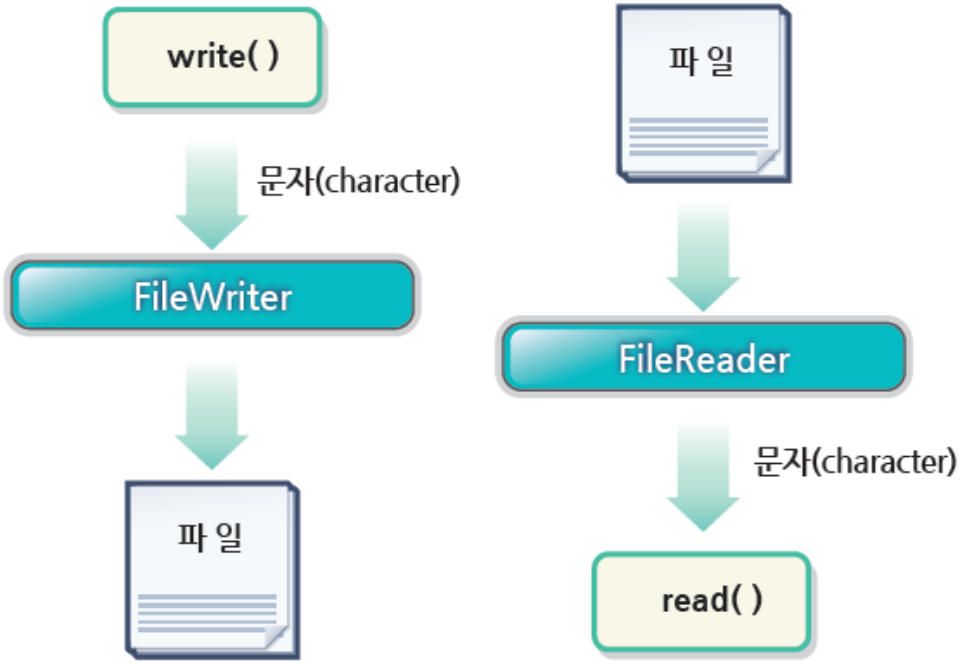


그림 13-5 FileWriter 클래스와 FileReader 클래스의 역할

7.2 FileWriter 클래스와 FileReader 클래스

예제 13.4 FileWriterTest1.java

```
01: import java.io.*;
02: import java.util.Scanner;
03: public class FileWriterTest1 {
04:     public static void main(String args[]) throws Exception {
05:         Scanner stdin = new Scanner(System.in);
06:         String source = "비어 있어야 비로소 가득해지는 사랑\n" + "영원히
    사랑한다는 것은\n" + "평온한 마음으로 아침을 맞는다는 것입니다.\n";
07:         System.out.print("파일명을 입력하세요 : ");
08:         String s = stdin.next();
09:         FileWriter fw = new FileWriter(s);
10:         fw.write(source);
11:         fw.close();
12:         System.out.print(s + "파일이 생성되었습니다");
13:     }
14: }
```

throws 절을 이용하여 예외 처리

파일에 저장할 문자열을 생성

파일 명으로 FileWriter 객체 생성

문자 배열을 파일에 출력

출력 스트림을 닫는다.

7.2 FileWriter 클래스와 FileReader 클래스

예제 13.5 FileReaderTest1.java

```
01: import java.io.*;
02: import java.util.Scanner;
03: class FileReaderTest1 {
04:     public static void main(String args[]) throws Exception {
05:         Scanner stdin = new Scanner(System.in);
06:         System.out.print("읽어 들일 파일명을 입력하세요 : ");
07:         String s = stdin.next();
08:         FileReader fr = new FileReader(s);
09:         int i;
10:         while((i = fr.read()) != -1 ) {
11:             System.out.print((char)i);
12:         }
13:         fr.close();
14:     }
15: }
```

throws 절을 이용하여 예외 처리

읽어 들일 파일명으로 객체를 생성

한 문자씩 읽는다.

문자를 출력

입력 스트림을 닫는다.

실행 결과

```
읽어 들일 파일명을 입력하세요 : aa.txt
비어 있어야 비로소 가득해지는 사랑
 영원히 사랑한다는 것은
 평온한 마음으로 아침을 맞는다는 것입니다.
```

- 바이트 스트림은 8비트인 바이트(byte) 단위로 입출력을 제어하므로 파일을 2진 데이터로 취급
- 바이트 스트림 관련 클래스

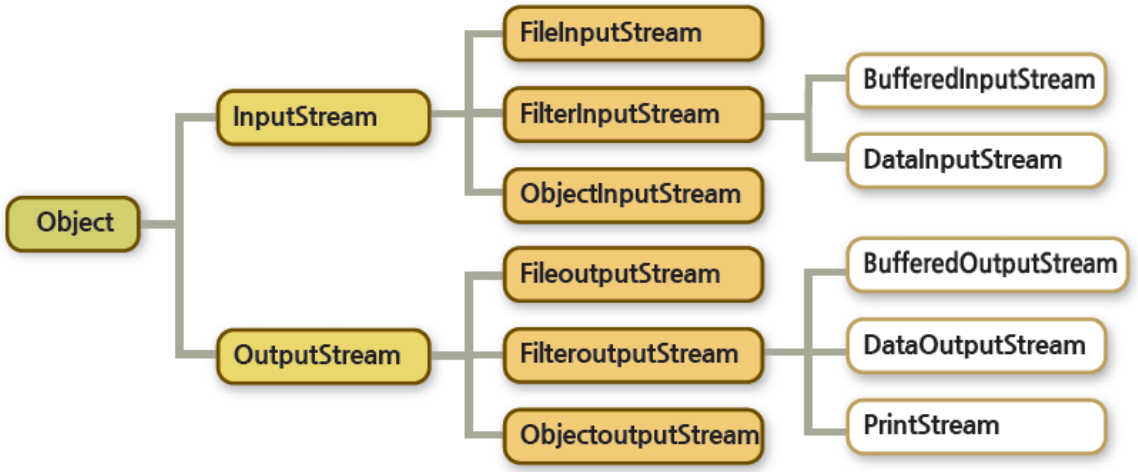


그림 13-6 바이트 스트림과 연관된 클래스

8-1 OutputStream 클래스와 InputStream 클래스

- **OutputStream과 InputStream 클래스는 문자 스트림의 Writer와 Reader 클래스와 같이 추상 클래스**
  - 추상 클래스로서 바이트 스트림의 입출력을 위한 메소드들이 선언되어 있다

표 13-4 OutputStream 클래스의 주요 메소드

메소드 이름	설명
void close() throws IOException	출력 스트림을 닫는다.
void flush() throws IOException	버퍼에 남아 있는 출력 스트림을 모두 출력
void write(int i) throws IOException	정수 i의 하위 8비트를 출력
void write(byte buffer[]) throws IOException	buffer의 내용을 출력
void write(byte buffer[], int index, int size) throws IOException	buffer의 index위치부터 size만큼의 바이트를 출력

8-1 OutputStream 클래스와 InputStream 클래스

표 13-5 InputStream 클래스의 주요 메소드

메소드 이름	설명
int available()	현재 읽기 가능한 바이트의 수를 반환
void close()	입력 스트림을 닫는다.
int read()	입력 스트림으로부터 한 바이트를 읽어 int 값으로 반환한다. 더 이상 읽을 값이 없을 경우 -1을 반환
int read(byte buffer[])	입력 스트림으로부터 buffer[] 크기만큼을 읽어 buffer 배열에 저장하고 읽은 바이트 수를 반환
int read(byte buffer[], int offset, int numBytes)	입력 스트림으로부터 numBytes만큼을 읽어 buffer[]의 offset 위치에 저장하고 읽은 바이트 수를 반환
int skip(long numBytes)	numBytes로 지정된 바이트를 스킵(skip)하고 스킵된 바이트 수를 반환
void mark(int numBytes)	입력 스트림의 현재의 위치에 mark 한다.
boolean markSupported()	현재의 입력 스트림이 mark()와 reset()을 지원하면 true를 반환
void reset()	입력 스트림의 입력 시작 부분을 현재의 위치에서 가장 가까운 이전의 mark 위치로 설정

8-2 FileOutputStream 클래스와 FileInputStream 클래스

- 이 클래스들은 파일에 바이트 스트림을 출력하고, 반대로 파일로부터 바이트 스트림을 읽는 기능을 제공

[형식] FileOutputStream 클래스의 생성자

```
FileOutputStream(String filepath) throws IOException
FileOutputStream(String filepath, boolean append) throws IOException
FileOutputStream(File fileObj) throws IOException
```

filepath : 파일의 이름(경로명 포함)

fileObj : 특정 파일을 지정하고 있는 File 객체

append : append가 true이면 파일의 끝에 문자를 첨가하고, false이면 기존 문자 위에 겹쳐 출력한다.

8-2 FileOutputStream 클래스와 FileInputStream 클래스

[형식] FileInputStream 클래스의 생성자

```
FileInputStream(String filepath) throws FileNotFoundException
FileInputStream(File fileObj) throws FileNotFoundException
```

filepath : 파일의 이름(경로명 포함)

fileObj : 특정 파일을 지정하고 있는 File 객체

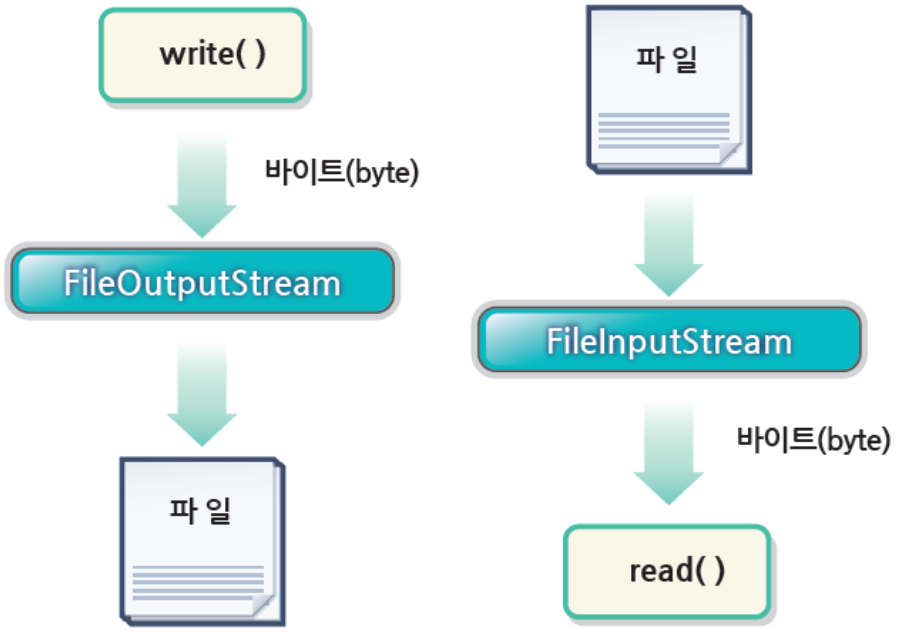


그림 13-7 FileOutputStream 클래스와 FileInputStream 클래스의 역할

8-2 FileOutputStream 클래스와 FileInputStream 클래스

예제 13.6 FileOutputStreamTest1.java

```
01: import java.io.*;
02: import java.util.Scanner;
03: public class FileOutputStreamTest1 {
04:     public static void main(String args[]) throws IOException {
05:         Scanner stdin = new Scanner(System.in);
06:         System.out.print("출력 파일명을 입력하세요 : ");
07:         String s = stdin.next();
08:         FileOutputStream fos = new FileOutputStream(s);
09:         for(int i = 1; i <= 260 ; i++) {
10:             fos.write(i);
11:         }
12:         fos.close();
13:         System.out.println(s+ " 파일명으로 바이트 파일을 생성하였습니다.");
14:     }
15: }
```

실행 결과

```
출력 파일명을 입력하세요 : byte.txt
byte.txt 파일명으로 바이트 파일을 생성하였습니다.
```

8-2 FileOutputStream 클래스와 FileInputStream 클래스

예제 13.7 FileInputStreamTest.java

```
01: import java.io.*;
02: import java.util.Scanner;
03: class FileInputStreamTest {
04:     public static void main(String args[]) throws IOException {
05:         Scanner stdin = new Scanner(System.in);
06:         System.out.print("입력 파일명을 입력하세요 : ");
07:         String s = stdin.next();
08:         FileInputStream fis = new FileInputStream(s);
09:         int i;
10:         while((i = fis.read()) != -1) {
11:             System.out.println(i);
12:         }
13:         fis.close();
14:         System.out.println(s+" 파일로부터 바이트를 읽어 화면에
    출력하였습니다.") ;
15:     }
16: }
```

throws 절을 이용하여 예외 처리

입력할 파일명으로 객체를 생성

바이트 단위로 읽어 출력

입력 스트림을 닫는다.

실행 결과

```
중간 생략
입력 파일명을 입력하세요 : byte.txt
1
2
.....생략
253
254
255
0
1
2
3
4
byte.txt 파일로부터 바이트를 읽어 화면에 출력하였습니다.
```

8-3 DataOutputStream 클래스와 DataInputStream 클래스

- **DataOutputStream 클래스는 FilterOutputStream 클래스로부터 상속되었고 DataOutput 인터페이스를 사용하여 구현된 클래스**
  - 이 클래스는 자바의 기본 자료형 데이터를 바이트 스트림으로 출력하는 기능을 제공

[형식] DataOutputStream 클래스 생성자

```
DataOutputStream(OutputStream outputStream)
```

* 이 클래스는 OutputStream 클래스의 객체를 생성자 매개 변수로 받습니다.

8-3 DataOutputStream 클래스와 DataInputStream 클래스

표 13-6 DataOutputStream 클래스에 구현된 DataOutput 인터페이스의 메소드

메소드 이름	설명
void write(int i) throws IOException	i의 하위 8비트를 출력
void write(byte buffer[]) throws IOException	buffer를 스트림으로 출력
void write(byte buffer[], int index, int size) throws IOException	buffer의 index 위치부터 size만큼의 바이트를 스트림으로 출력
void writeByte(int i) throws IOException	i의 하위 8비트를 스트림으로 출력
void writeBytes(String s) throws IOException	문자열 s를 스트림으로 출력
void writeChar(int i) throws IOException	i의 하위 16비트를 스트림으로 출력
void writeChars(String s) throws IOException	문자열 s를 스트림으로 출력
void writeDouble(double d) throws IOException	d를 스트림으로 출력
void writeFloat(float f) throws IOException	f를 스트림으로 출력
void writeInt(int i) throws IOException	i를 스트림으로 출력
void writeLong(long l) throws IOException	l을 스트림으로 출력
void writeShort(short s) throws IOException	s를 스트림으로 출력
void writeUTF(String s) throws IOException	s를 스트림으로 출력. 유니코드를 UTF-8 엔코딩을 이용하여 변환

8-3 DataOutputStream 클래스와 DataInputStream 클래스

- **DataInputStream 클래스는 FilterInputStream 클래스로부터 상속되고 DataInput 인터페이스를 사용하여 구현된 클래스**
  - 이 클래스는 DataOutputStream 클래스와 반대로 바이트 스트림으로부터 자바의 기본 자료형 데이터를 읽는 기능을 제공

[형식] DataInputStream 클래스 생성자

```
DataInputStream(InputStream inputStream)
```

* 이 클래스는 InputStream 클래스의 객체를 생성자 매개 변수로 받습니다.

8-3 DataOutputStream 클래스와 DataInputStream 클래스

그림 13-8 DataOutputStream 클래스와 DataInputStream 클래스의 역할

8-3 DataOutputStream 클래스와 DataInputStream 클래스

표 13-7 DataInputStream 클래스에 구현된 DataInput 인터페이스의 메소드

메소드 이름	설명
boolean readBoolean(boolean b) throws IOException	스트림으로부터 읽은 boolean을 반환
byte readByte( ) throws IOException	스트림으로부터 읽은 byte를 반환
char readChar( ) throws IOException	스트림으로부터 읽은 char를 반환
double readDouble( ) throws IOException	스트림으로부터 읽은 double을 반환
float readFloat( ) throws IOException	스트림으로부터 읽은 float를 반환
long readLong( ) throws IOException	스트림으로부터 읽은 long을 반환
short readShort( ) throws IOException	스트림으로부터 읽은 short를 반환
int readInt( ) throws IOException	스트림으로부터 읽은 int를 반환
void readFully(byte buffer[ ]) throws IOException	스트림으로부터 buffer 크기만큼의 바이트를 읽어 buffer 배열에 저장
void readfully(byte buffer[ ] int index, int size) throws IOException	스트림으로부터 size만큼의 바이트를 읽어 buffer의 index 위치에 저장
int readUnsignedByte( ) throws IOException	스트림으로부터 읽은 unsigned byte를 반환
int readUnsignedShort( ) throws IOException	스트림으로부터 읽은 unsigned short를 반환
int skipBytes(int n)	입력 스트림으로부터 n 바이트를 스킵

8-3 DataOutputStream 클래스와 DataInputStream 클래스

예제 13.8 DataOutputStreamTest1.java

```
01: import java.io.*;
02: import java.util.Scanner;
03: public class DataOutputStreamTest1 {
04:     public static void main(String args[]) throws IOException {
05:         Scanner stdin = new Scanner(System.in);
06:         System.out.print("출력할 파일명을 입력하세요 : ");
07:         String s = stdin.next();
08:         FileOutputStream fos = new FileOutputStream(s);
09:         DataOutputStream dos = new DataOutputStream(fos);
10:         dos.writeBoolean(false);
11:         dos.writeByte(Byte.MAX_VALUE);
12:         dos.writeChar('김');
13:         dos.writeDouble(Double.MAX_VALUE);
14:         dos.writeFloat(Float.MIN_VALUE);
15:         dos.writeInt(Integer.MAX_VALUE);
16:         dos.writeLong(Long.MAX_VALUE);
17:         dos.writeShort(Short.MAX_VALUE);
18:         fos.close();
19:         System.out.print(s + "파일이 생성되었습니다");
20:     }
21: }
```

throws 절을 이용하여 예외 처리

출력할 파일명으로 객체를 생성

생성된 객체를 이용하여 DataOutputStream 객체를 생성

다양한 형태의 기본 자료형 데이터 출력

실행 결과

```
출력할 파일명을 입력하세요 : data.txt
data.txt 파일이 생성되었습니다
```

8-3 DataOutputStream 클래스와 DataInputStream 클래스

예제 13.9 DataInputStreamTest.java

```
01: import java.io.*;
02: import java.util.Scanner;
03: class DataInputStreamTest1 {
04:     public static void main(String args[]) throws IOException {
05:         Scanner stdin = new Scanner(System.in);
06:         System.out.print("입력 파일명을 입력하세요 : ");
07:         String s = stdin.next();
08:        FileInputStream fis = new FileInputStream(s);
09:         DataInputStream dis = new DataInputStream(fis);
10:         System.out.println(dis.readBoolean());
11:         System.out.println(dis.readByte());
12:         System.out.println(dis.readChar());
13:         System.out.println(dis.readDouble());
14:         System.out.println(dis.readFloat());
15:         System.out.println(dis.readInt());
16:         System.out.println(dis.readLong());
17:         System.out.println(dis.readShort());
18:         fis.close();
19:     }
20: }
```

throws 절을 이용하여 예외 처리

입력할 파일명으로 객체를 생성

생성된 객체를 이용하여 DataInputStream 객체를 생성

파일로부터 다양한 형태의 기본 자료형 데이터 입력하여 화면에 출력

실행 결과

```
입력 파일명을 입력하세요 : data.txt
false
127
김
1.7976931348623157E308
1.4E-45
2147483647
9223372036854775807
32767
```

8-4 ObjectOutputStream 클래스와 ObjectInputStream 클래스

- **자바에서 객체를 입출력하기 위해서 ObjectOutputStream 클래스와 ObjectInputStream 클래스를 제공**
  - 이 클래스들은 객체 단위로 입출력을 수행
- **이 클래스들이 객체를 입출력할 때는 직렬화Serialization된 데이터를 사용**
  - 직렬화된 데이터란 객체를 순차적인 바이트로 표현한 데이터를 의미
  - 객체를 직렬화된 데이터로 표현함으로써 객체의 영속성Object Persistence을 보장
    - 영속성이란 객체가 자신의 상태를 기록해 두어 다음 기회에 또는 다른 환경에서 재생될 수 있는 능력을 의미

- **객체가 직렬화된 데이터로 사용되기 위해서는 객체를 생성하는 클래스가 Serializable 인터페이스를 포함하고 있어야 한다**
  - 자바에서 제공되는 대부분의 클래스들은 Serializable 인터페이스를 포함하고 있기 때문에 직렬화된 데이터로 취급 가능
  - 그러나 사용자가 작성하는 클래스의 객체를 직렬화된 데이터로 사용하기 위해서는 반드시 Serializable 인터페이스를 포함시켜 클래스를 작성하여야 한다.

```
01: class Box implements Serializable {
02:     int w;
03:     int h;
04:     int d;
05: }
```

8-4 ObjectOutputStream 클래스와 ObjectInputStream 클래스

- **ObjectOutputStream 클래스와 ObjectInputStream 클래스는 객체뿐만 아니라 기본 자료형의 입출력에도 사용될 수 있다.**
  - ObjectOutputStream 클래스는 자바의 기본 자료형과 객체를 직렬화된 데이터로 저장하기 위해 사용되는 클래스로서 다음과 같은 생성자를 가진다

[형식] ObjectOutputStream 클래스 생성자

```
ObjectOutputStream(OutputStream outputStream)
```

* 이 클래스는 OutputStream 클래스의 객체를 생성자 매개 변수로 받습니다

8-4 ObjectOutputStream 클래스와 ObjectInputStream 클래스

표 13-8 ObjectOutputStream 클래스의 주요 메소드

메소드 이름	설명
void close( )	스트림을 닫는다.
void flush( )	버퍼의 모든 내용을 출력한다.
void write(byte[ ] b, int index, int size)	바이트 배열 b의 index 위치부터 size만큼을 출력한다.
void write(byte[ ] b)	바이트 배열 b를 출력한다.
void write(int ontbyte)	int값의 하위 한 바이트를 출력한다.
void writeBoolean(boolean data)	boolean값을 출력한다.
void writeByte(int data)	한 바이트를 출력한다.
void writeBytes(String str)	str 문자열을 연속된 바이트로 출력한다.
void writeChar(int data)	한 문자를 출력한다.
void writeChars(String str)	str 문자열을 연속된 문자로 출력한다.
void writeDouble(double data)	double값을 출력한다.
void writeFloat(float data)	float값을 출력한다.
void writeInt(int data)	int값을 출력한다.
void writeLong(long data)	long값을 출력한다.
void writeObject(Object obj)	객체를 출력한다.
void writeShort(int data)	short값을 출력한다.
void writeUTF(String data)	문자열을 UTF 인코딩으로 출력한다.

8-4 ObjectOutputStream 클래스와 ObjectInputStream 클래스

- **ObjectInputStream 클래스는 직렬화된 데이터로부터 역직렬화 Deserialization를 수행하여 원래의 데이터로 복구하는 기능을 제공하는 클래스**

[형식] ObjectInputStream 클래스 생성자

```
ObjectInputStream(InputStream inputStream)
```

* 이 클래스는 InputStream 클래스의 객체를 생성자 매개 변수로 받습니다.

8-4 ObjectOutputStream 클래스와 ObjectInputStream 클래스

표 13-9 ObjectOutputStream 클래스의 주요 메소드

메소드 이름	설명
void close()	스트림을 닫는다.
int read()	한 바이트를 읽는다.
int read(byte[] b, int off, int len)	바이트 배열로 읽어 들인다.
boolean readBoolean()	boolean 값을 읽는다.
byte readByte()	한 바이트를 읽는다.
char readChar()	문자를 읽는다.
double readDouble()	double값을 읽는다.
float readFloat()	float값을 읽는다.
int readInt()	int값을 읽는다.
long readLong()	long값을 읽는다.
Object readObject()	객체를 읽는다.
short readShort()	short값을 읽는다.
int readUnsignedByte()	부호 없는 바이트 값을 읽는다.
int readUnsignedShort()	부호 없는 short값을 읽는다.
String readUTF()	UTF 인코딩을 읽어서 문자열 타입으로 반환

● 예제 13.10

```
01: import java.io.*;
02: class Box implements Serializable {  <-- 객체를 저장하기 위해 Serializable 인터페이스를 포함
03:     public int width;
04:     public int height;
05:     public int depth;
06:     public Box(int w, int h, int d) {
07:         width = w;
08:         height = h;
09:         depth = d;
10:     }
11: }
12: public class ObjectWriteTest1 {
13:     public static void main(String args[]) throws Exception {
14:         String s1 = "***박스의 가로,세로,높이***";
15:         Box mybox1 = new Box(10,20,30);
16:         ObjectOutputStream oos = new ObjectOutputStream(new
    FileOutputStream("tmp.txt"));  <-- 파일에 출력하기 위해 객체를 생성
17:         oos.writeObject(s1);
18:         oos.writeObject(mybox1);   <-- 메소드를 통하여 객체 출력
19:         oos.writeDouble(123.456);
20:         oos.close();
21:         System.out.println("tmp.txt 파일명으로 객체 파일을 생성하였습니다.");
22:     }
23: }
```

실행 결과

```
tmp.txt 파일명으로 객체 파일을 생성하였습니다.
```

8-4 ObjectOutputStream 클래스와 ObjectInputStream 클래스

예제 13.11 ObjectReadTest1.java

```
01: import java.io.*;
02: public class ObjectReadTest1 {
03:     public static void main(String args[]) throws Exception {
04:         ObjectInputStream ois = new ObjectInputStream(new
    FileInputStream("tmp.txt"));
05:         Object s2 = ois.readObject();
06:         Box mybox2 = (Box)ois.readObject();
07:         System.out.println(s2);
08:         System.out.println("박스의 가로는 : "+mybox2.width);
09:         System.out.println("박스의 세로는 : "+mybox2.height);
10:         System.out.println("박스의 높이는 : "+mybox2.depth);
11:         System.out.println("Double 값은 : " + ois.readDouble());
12:     }
13: }
```

ObjectInputStream 객체 생성

readObject() 메소드를 이용하여 객체를 읽는다.

객체를 읽어 Box 형으로 형 변환

객체의 속성을 출력

Double 객체를 출력

실행 결과

```
***박스의 가로,세로,높이***
박스의 가로는 : 10
박스의 세로는 : 20
박스의 높이는 : 30
Double 값은 : 123.456
```

학습 정리

- **예외의 개요**

  ① 예외란 프로그램 실행 중에 발생하는 예기치 않은 사건을 의미합니다.

  ② JVM은 프로그램 실행 중 예외가 발생하면, 관련된 예외 클래스로부터 객체를 생성하여예외를 발생시킨 메소드에 전달합니다.

- **예외 관련 클래스**

  ① 자바는 약 50여 개의 예외 관련 클래스를 제공하고 있습니다.

  ② RuntimeException 클래스와 Error 클래스와 연관된(하위 클래스 포함) 예외는 프로그램에서 처리하지 않습니다. 얻어지는 효과보다 예외 처리에 드는 시간과 비용이 크기 때문입니다.

  ③ 자바는 RuntimeException 클래스와 Error 클래스가 아닌 예외들은 명시적인 예외 처리를 요구합니다..

학습 정리

- **예외 처리**

  ① 자바는 2가지 예외 처리 방법을 제공합니다.

  ② 한 가지 방법은 try-catch 절을 이용하여 예외가 발생하면, 발생한 메소드 내에서 예외를 처리하는 방법이며, 다른 방법은 발생된 예외를 호출한 메소드에 넘겨주는 방법입니다.

- **java.io 패키지 개요**

  ① 자바의 입출력은 하드웨어와 독립적으로 설계되어 어떠한 컴퓨터에서나 일관된 입출력을 수행합니다.

  ② 자바의 입출력은 스트림(stream)을 사용합니다. 스트림은 순서가 있는 일련의 데이터를 의미합니다.

- **파일과 디렉터리**

  ① 자바는 파일과 디렉터리를 다루기 위해 File 클래스를 제공합니다. 다양한 메소드를 이용하여 디렉터리와 파일에 관한 정보를 얻을 수 있습니다.

학습 정리

- **문자 스트림과 바이트 스트림**

  ① 스트림에는 문자 스트림과 바이트 스트림 두 가지 형태가 있습니다.

  ② 문자 스트림은 16비트 문자나 문자열들을 읽고 쓰기 위한 스트림이고, 바이트 스트림(또는 바이너리 스트림)은 8비트의 바이트를 읽고 쓰기 위한 스트림입니다.

- **문자 스트림**

  ① Writer와 Reader 클래스는 문자 스트림 입출력을 대표하는 추상 클래스로서 다양한 메소드를 가지고 있습니다. 이러한 추상 클래스는 하위 클래스에서 오버라이딩되어 사용 됩니다.

  ② FileWriter 클래스와 FileReader 클래스는 파일에 문자 스트림을 입출력하기 위해 사용하는 클래스입니다.

학습 정리

- **바이트 스트림**

  ① OutputStream 클래스와 InputStream 클래스는 바이트 스트림 입출력을 대표하는 추상클래스로서 다양한 메소드를 가지고 있습니다. 이러한 추상 클래스는 하위 클래스에서 오버라이딩되어 사용됩니다.

  ② FileOutputStream 클래스와 FileInputStream 클래스는 파일에 바이트 스트림을 입출력하기 위해 사용하는 클래스입니다.

  ③ DataOutputStream 클래스와 DataInputStream 클래스는 자바의 기본 자료형 데이터를 바이트로 입출력하기 위해 사용하는 클래스입니다.

  ④ OjbectOutputStream 클래스와 OjbectInputStream 클래스는 객체를 입출력하기 위한 클래스입니다. 객체를 입출력하기 위해 자바는 직렬화된 데이터를 사용합니다.