18장 IO 기반 입출력 및 네트워킹

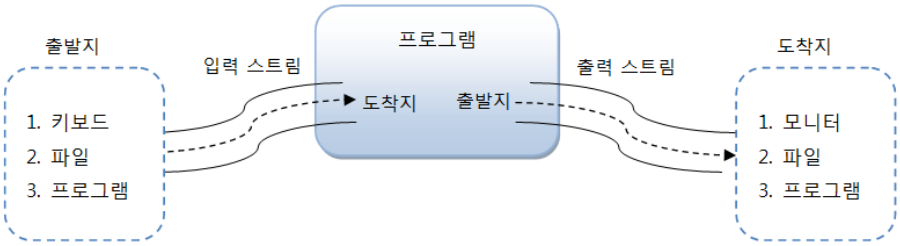
18.1 IO 패키지 소개

■ 자바에서 데이터는 스트림(Stream)을 통해 입출력된다. 스트림은 단일 방향으로 연속적으로 흘러가는 것을 말하는데, 물이 높은 곳에서 낮은 곳으로 흐르듯이 데이터는 출발지에서 나와 도착지로 들어간다는 개념이다.

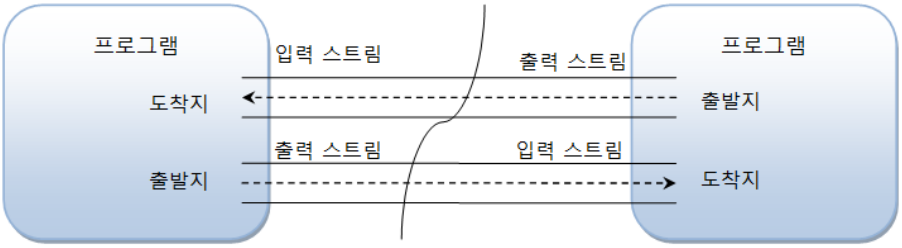
18.2 입력 스트림과 출력 스트림

■ 프로그램이 출발지냐 또는 도착지냐에 따라서 스트림의 종류가 결정된다.

■ 스트림의 특성이 단방향이므로 하나의 스트림으로 입력과 출력을 모두 할 수 없다.



■ 항상 프로그램을 기준으로 데이터가 들어오면 입력 스트림이고, 데이터가 나가면 출력 스트림이라는 것을 명심해야 한다.



■ 자바의 기본적인 데이터 입출력(IO: Input/Output) API는 java.io 패키지에서 제공하고 있다.

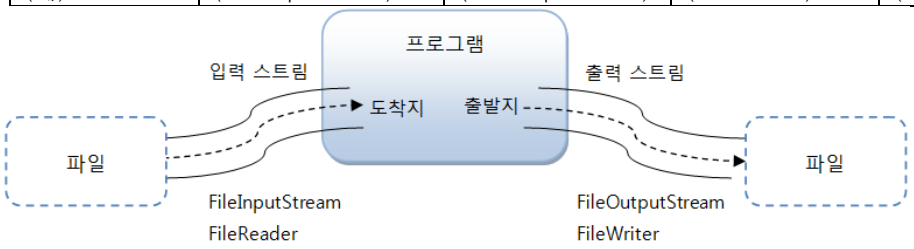
java.io 패키지의 주요 클래스	설명
File	파일 시스템의 파일 정보를 얻기 위한 클래스
Console	콘솔로부터 문자를 입출력하기 위한 클래스
InputStream / OutputStream	바이트 단위 입출력을 위한 최상위 입출력 스트림 클래스
FileInputStream / FileOutStream DataInputStream / DataOutStream ObjectInputStream / ObjectOutputStream PrintStream BufferedInputStream / BufferedOutputStream	바이트 단위 입출력을 위한 하위 스트림 클래스
Reader / Writer	문자 단위 입출력을 위한 최상위 입출력 스트림 클래스
FileReader / FileWriter InputStreamReader / OutputStreamWriter PrintWriter BufferedReader / BufferedWriter	문자 단위 입출력을 위한 스트림 클래스

■ 바이트 기반 스트림 vs 문자 기반 스트림

- 바이트 기반 스트림 : 그림, 멀티미디어, 문자 등 모든 종류의 데이터를 받고 보내는 것 가능

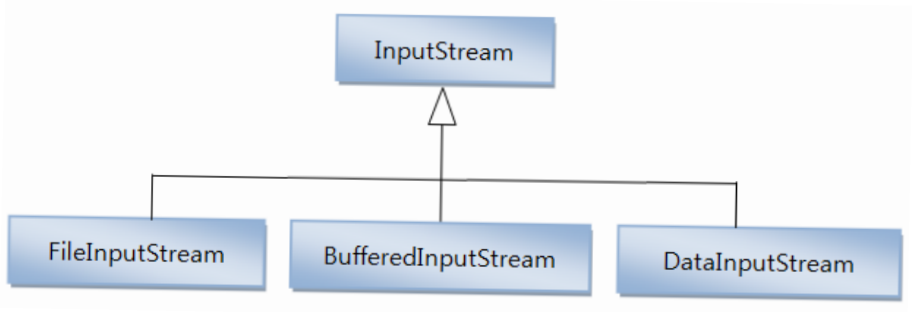
- 문자 기반 스트림 : 문자만 받고 보낼 수 있도록 특화

구분	바이트 기반 스트림		문자 기반 스트림	
	입력 스트림	출력 스트림	입력 스트림	출력 스트림
최상위 클래스	InputStream	OutputStream	Reader	Writer
하위 클래스 (예)	XXXInputStream (FileInputStream)	XXXOutputStream (FileOutputStream)	XXXReader (FileReader)	XXXWriter (FileWriter)



18.2.1 InputStream

■ 바이트 기반 입력 스트림의 최상위 클래스로 추상 클래스



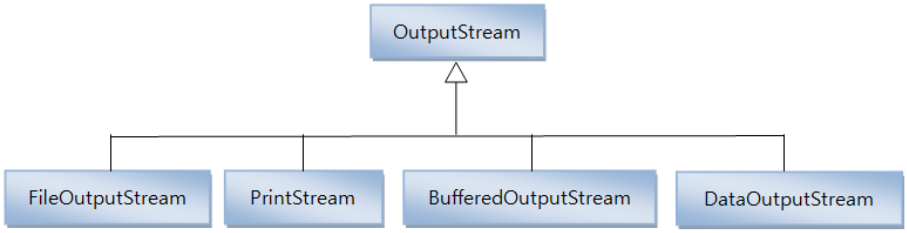
■ InputStream 클래스의 주요 메소드

리턴타입	메소드	설명
int	read()	입력 스트림으로부터 1 바이트를 읽고 읽은 바이트를 리턴한다.
int	read(byte[] b)	입력 스트림으로부터 읽은 바이트들을 매개값으로 주어진 바이트 배열 b 에 저장하고 실제로 읽은 바이트 수를 리턴한다.
int	read(byte[] b, int off, int len)	입력 스트림으로부터 len 개의 바이트 만큼 읽고 매개값으로 주어진 바이트 배열 b[off] 부터 len 개까지 저장한다. 그리고 실제로 읽은 바이트 수인 len 개를 리턴한다. 만약 len 개를 모두 읽지 못하면 실제로 읽은 바이트 수를 리턴한다.
void	close()	사용한 시스템 자원을 반납하고 입력 스트림을 닫는다.

```
InputStream is = new FileInputStream("C:/test.txt");
int readyByteNo;
while ((readyByteNo=is.read()) != -1) { ... }

byte[] readBytes = new byte[100];
while ((readyByteNo=is.read(readBytes)) != -1) { ... }

// readByteNo = is.read(readBytes);
readByteNo = is.read(readBytes,0,100);

is.close();
```

18.2.2 OutputStream

■ 바이트 기반 출력 스트림의 최상위 클래스로 추상 클래스



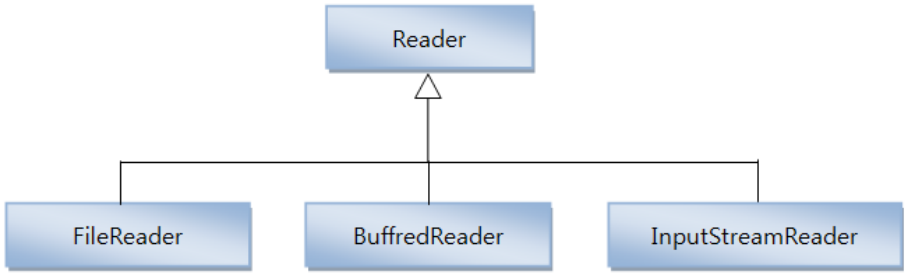
■ OutputStream의 주요 메소드

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.

```
OutputStream os = new FileOutputStream("C:/test.txt");
byte[] data = "ABC".getBytes();
for(int i=0; i<data.length; i++) {
   os.write(data[i]); // "A", "B", "C"를 하나씩 출력
}

os.write(data); // "ABC" 모두 출력

os.write(data, 1, 2); // "BC"만 출력

os.flush();
os.close();
```

18.2.3 Reader

■ Reader는 문자 기반 입력 스트림의 최상위 클래스로 추상 클래스이다. FileReader, BufferedReder, InputStreamReader 클래스는 모두 Reader 클래스를 상속하고 있다.



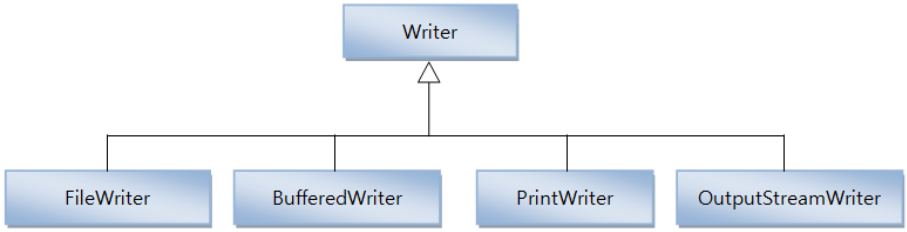
■ Reader의 주요 메소드

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.

```
Reader reader = new FileReader("C:/test.txt");
int readData;
while ((readData=reader.read()) != -1) {
   char charData = (char) readData;
}

int readCharNo;
Char[] cbuf = new Char[2];
while ((readCharNo=reader.read(cbuf)) != -1) { ... }

Char[] cbuf = new Char[100];
readCharNo=reader.read(cbuf, 0, 100);

reader.close();
```

18.2.4 Writer

■ 문자 기반 출력 스트림의 최상위 클래스로 추상 클래스



■ Writer의 주요 메소드

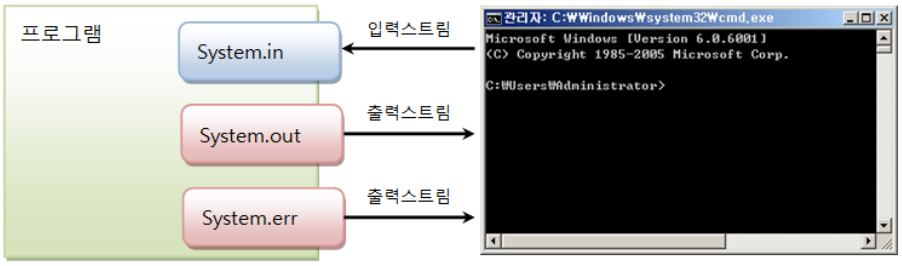
리턴타입	메소드	설명
void	wrtie(int c)	출력 스트림으로 매개값으로 주어진 한 문자를 보낸다.
void	write(char[] cbuf)	출력 스트림에 매개값으로 주어진 문자 배열 cbuf 의 모든 문자를 보낸다.
void	write(char[] cbuf, int off, int len)	출력 스트림에 매개값으로 주어진 문자 배열 cbuf[off] 부터 len 개까지의 문자를 보낸다.
void	write(String str)	출력 스트림에 매개값으로 주어진 문자열을 전부 보낸다.
void	write(String str, 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.

```
Writer writer = new FileWriter("C:/test.txt");
Char[] data = "홍길동".toCharArray();
for(int i=0; i<data.length; i++) {
   writer.write(data[i]); // "홍", "길", "동"을 하나씩 출력
```

```
writer.write(data); // "홍길동" 모두 출력

writer.write(data, 1, 2); // "길동"만 출력

String data = "안녕 자바 프로그램";
writer.write(data);
writer.flush();
writer.close();
```

18.3 콘솔 입출력

- 콘솔(Console): 시스템을 사용하기 위해 키보드로 입력을 받고 화면으로 출력하는 소프트웨어
  - Unix, Linux: 터미널
  - Windows 운영체제: 명령 프롬프트
  - 이클립스: Console 뷰



18.3.1 System.in 필드

- InputStream 타입의 입력 스트림 - InputStream 변수 대입 가능
- 읽은 byte는 키보드의 아스키 코드(ascii code)
- 아스키 코드로부터 문자 변환
- 키보드로부터 입력된 한글 읽기 예제
  - read()메소드는 1바이트씩만 읽음 → 오류 발생
  - 한글과 같이 2바이트를 필요로 하는 유니코드는 read() 메소드로 읽을 수 없다.
  - 전체 내용을 바이트 배열로 받아 String 객체 생성 후 읽기

```
InputStream is = System.in;
int asciiCode = is.read();

Char inputChar = (char) is.read();

byte[] byteData = new byte[15];
int readByteNo = is.read(byteData);
String strData = new String(byteData, 0, readByteNo-2); // 2를 빼는 이유는 Enter키에 제외하기 위함.
```

[SystemInExample2.java] 콘솔에서 입력한 한글 알아내기

```
package sec03.exam01_system_in_out;
```

```
import java.io.InputStream;

public class SystemInExample2 {
	public static void main(String[] args) throws Exception {
		InputStream is = System.in;

		byte[] datas = new byte[100];

		System.out.print("이름: ");
		int nameBytes = is.read(datas);
		String name = new String(datas, 0, nameBytes-2);

		System.out.print("하고 싶은말: ");
		int commentBytes = is.read(datas);
		String comment = new String(datas, 0, commentBytes-2);

		System.out.println("입력한 이름: " + name);
		System.out.println("입력한 하고 싶은말: " + comment);
	}
}
```

18.3.2 System.out 필드

- PrintStream 타입의 출력 스트림 - OutputStream으로 타입 변환 가능
- 아스키 코드를 출력하면 콘솔에는 문자가 출력
- 문자열을 출력하려면 바이트 배열을 얻어야

```
OutputStream os = System.out;
byte b = 97;
os.write(b);
os.flush();

String name = "홍길동";
byte[] nameBytes = name.getBytes();
os.write(nameBytes);
os.flush();

PrintStream ps = System.out;
ps.println(...);
// System.out.println(...);
```

[SystemOutExample.java] 연속된 숫자, 영어, 한글 출력

```
package sec03.exam01_system_in_out;

import java.io.OutputStream;

public class SystemOutExample {
	public static void main(String[] args) throws Exception {
		OutputStream os = System.out;

		for(byte b=48; b<58; b++) {
			os.write(b);  // 아스키 코드 48에서 57까지의 문자를 출력한다.
		}
		os.write(10);  // 라인피드(10)을 출력하면 다음 행으로 넘어간다.
```

```
                    for(byte b=97; b<123; b++) {
                              os.write(b);  // 아스키 코드 97에서 122까지의 문자를 출력한다.
                    }
                    os.write(10);

                    String hangul = "가나다라마바사아자차카타파하";
                    byte[] hangulBytes = hangul.getBytes();
                    os.write(hangulBytes);

                    os.flush();
          }
}
```

18.3.3 Console 클래스

- 자바6부터 콘솔에서 입력된 문자열을 쉽게 읽을 수 있도록 제공하고 있다.
  - 기본형: Console console = System.console();
  - 주의할 점은 이클립스에서 System.console() 메소드는 null 리턴하기 때문에 반드시 명령 프롬프트에서 실행해야 한다.

- Console 클래스의 읽기 메소드

리턴 타입	메소드	설명
String	readLine()	Enter키를 입력하기 전의 모든 문자열을 읽음
char[]	readPassword()	키보드 입력 문자를 콘솔에 보여주지 않고 문자열을 읽음

[ConsoleExample.java] 아이디와 패스워드를 콘솔로부터 읽음

```
package sec03.exam02_console;

import java.io.Console;

public class ConsoleExample {
          public static void main(String[] args) {
                    Console console = System.console();

                    System.out.print("아이디: ");
                    String id = console.readLine();

                    System.out.print("패스워드: ");
                    char[] charPass = console.readPassword();
                    String strPassword = new String(charPass);

                    System.out.println("---------------------");
                    System.out.println(id);
                    System.out.println(strPassword);
          }
}
```

18.3.4 Scanner 클래스

- Console 클래스의 단점
  - 문자열은 읽을 수 있지만 기본 타입(정수, 실수) 값을 바로 읽을 수 없음

- java.util.Scanner
  - 콘솔로부터 기본 타입의 값을 바로 읽을 수 있음

- 제공하는 메소드

리턴 타입	메소드	설명
boolean	nextBoolean()	boolean(true/false) 값을 얻는다.
byte	nextByte()	byte 값을 읽는다.
short	nextShort()	short 값을 읽는다.
int	nextInt()	int 값을 읽는다.
long	nextLong()	long 값을 읽는다.
float	nextFloat()	float 값을 읽는다.
double	nextDouble()	double 값을 읽는다.
String	nextLine()	String 값을 읽는다.

[ScannerExample.java] 문자열, 정수, 실수를 직접 읽는 예제

```
package sec03.exam03_scanner;

import java.util.Scanner;

public class ScannerExample {
	public static void main(String[] args) {
		Scanner scanner = new Scanner(System.in);

		System.out.print("문자열 입력> ");
		String inputString = scanner.nextLine();
		System.out.println(inputString);
		System.out.println();

		System.out.print("정수 입력> ");
		int inputInt = scanner.nextInt();
		System.out.println(inputInt);
		System.out.println();

		System.out.print("실수 입력> ");
		double inputDouble = scanner.nextDouble();
		System.out.println(inputDouble);
	}
}
```

18.4 파일 입출력

18.4.1 File 클래스

■ 파일 시스템의 파일을 표현하는 클래스
- 파일 크기, 파일 속성, 파일 이름 등의 정보 제공
- 파일 생성 및 삭제 기능 제공
- 디렉토리 생성, 디렉토리에 존재하는 파일 리스트 얻어내는 기능

```
File file = new File("C:\\Temp\\file.txt");  // 디렉토리 구분자는 운영체제마다 조금씩 다르다.
File file = new File("C:/Temp/file.txt");    // 윈도우(/ or \), 유닉스(/)
boolean isExist = file.exists();             // exists() 메소드로 파일이나 디렉토리의 존재 여부를 확
인
```

■ 파일 및 디렉토리 생성 및 삭제 메소드

리턴타입	메소드	설명
boolean	createNewFile()	새로운 파일을 생성
booelan	mkdir()	새로운 디렉토리를 생성
boolean	mkdirs()	경로상에 없는 모든 디렉토리를 생성
boolean	delete()	파일 또는 디렉토리 삭제

■ 파일 및 디렉토리의 정보를 리턴하는 메소드

리턴타입	메소드	설명
boolean	canExecute()	실행할 수 있는 파일인지 여부
boolean	canRead()	읽을 수 있는 파일인지 여부
boolean	canWrite()	수정 및 저장할 수 있는 파일인지 여부
String	getName()	파일의 이름을 리턴
String	getParent()	부모 디렉토리를 리턴
File	getParentFile()	부모 디렉토리를 File 객체로 생성후 리턴
String	getPath()	전체 경로를 리턴
boolean	isDirectory()	디렉토리인지 여부
boolean	isFile()	파일인지 여부
boolean	isHidden()	숨김 파일인지 여부
long	lastModified()	마지막 수정 날짜 및 시간을 리턴
long	length()	파일의 크기 리턴
String[]	list()	디렉토리에 포함된 파일 및 서브디렉토리 목록 전부를 String 배열로 리턴
String[]	list(FilenameFilter filter)	디렉토리에 포함된 파일 및 서브디렉토리 목록 중에 FilenameFilter 에 맞는 것만 String 배열로 리턴
File[]	listFiles()	디렉토리에 포함된 파일 및 서브 디렉토리 목록 전부를 File 배열로 리턴
File[]	listFiles(FilenameFilter filter)	디렉토리에 포함된 파일 및 서브디렉토리 목록 중에 FilenameFilter 에 맞는 것만 File 배열로 리턴

[FileExample.java] File 클래스를 이용한 파일 및 디렉토리 정보 출력

```
package sec04.exam01_file;

import java.io.File;
import java.net.URI;
import java.text.SimpleDateFormat;
import java.util.Date;

public class FileExample {
	public static void main(String[] args) throws Exception {
		File dir = new File("C:/Temp/Dir");
		File file1 = new File("C:/Temp/file1.txt");
		File file2 = new File("C:/Temp/file2.txt");
		File file3 = new File(new URI("file:///C:/Temp/file3.txt"));

		if(dir.exists() == false) {  dir.mkdirs();  }
		if(file1.exists() == false) {  file1.createNewFile();  }
		if(file2.exists() == false) {  file2.createNewFile();  }
		if(file3.exists() == false) {  file3.createNewFile();  }

		File temp = new File("C:/Temp");
		SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd   a   HH:mm");
		File[] contents = temp.listFiles();
		System.out.println("날짜                  시간          형태          크기     이름");
		System.out.println("-------------------------------------------------------------------------");
		for(File file : contents) {
			System.out.print(sdf.format(new Date(file.lastModified())));
			if(file.isDirectory()) {
				System.out.print("\t<DIR>\t\t\t" + file.getName());
			} else {
```

```
                                        System.out.print("\t\t\t" + file.length() + "\t" + file.getName());
                                }
                                System.out.println();
                        }
                }
}
```

18.4.2 FileInputStream

■ 파일로부터 바이트 단위로 읽어 들일 때 사용
- 그림, 오디오, 비디오, 텍스트 파일 등 모든 종류의 파일을 읽을 수 있음

■ 객체 생성 방법
- FileInputStream 객체가 생성될 때 파일과 직접 연결
- 만약 파일이 존재하지 않으면 FileNotFoundException 발생
- try-catch문으로 예외 처리

■ InputStream 하위 클래스 - 사용 방법이 InputStream과 동일

```
// 첫번째 방법
FileInputStream fis = new FileInputStream("C:/Temp/image.gif");

// 두번째 방법
File file = new File("C:/Temp/image.gif");
FileInputStream fis = new FileInputStream(file);
```

[FileInputStreamExample.java] 텍스트 파일을 읽고 출력

```
package sec04.exam02_fileinputstream;

import java.io.FileInputStream;

public class FileInputStreamExample {
        public static void main(String[] args) {
                try {
                        FileInputStream fis = new
FileInputStream("D:/02.download/java/chap18/src/sec04/exam02_fileinputstream/FileInputStreamExample.java");
                        int data;
                        while ( (data = fis.read() ) != -1 ) {
                                System.out.write(data);
                        }
                        fis.close();
                } catch(Exception e) {
                        e.printStackTrace();
                }
        }
}
```

18.4.3 FileOutputStream

■ 파일에 바이트 단위로 데이터를 저장할 때 사용
- 그림, 오디오, 비디오, 텍스트 등 모든 종류의 데이터를 파일로 저장

■ 객체 생성 방법
- 파일이 이미 존재할 경우, 데이터를 출력하게 되면 파일을 덮어쓰게 되므로 기존의 파일 내용은 사라지게 된다.
- 기존 파일 내용 끝에 데이터를 추가할 경우에는 FileOutputStream 생성자의 두 번째 매개값을 true로 주면 된다.
  예) FileOutputStream fos = new FileOutputStream("C:/Temp/data.txt", true);

[FileOutputStreamExample.java] 파일 복사

```
package sec04.exam03_fileoutputstream;

import java.io.FileInputStream;
import java.io.FileOutputStream;

public class FileOutputStreamExample {
	public static void main(String[] args) throws Exception {
		String originalFileName =
"D:/02.download/java/chap18/src/sec04/exam03_fileoutputstream/house.jpg";
		String targetFileName = "C:/Temp/house.jpg";

		FileInputStream fis = new FileInputStream(originalFileName);
		FileOutputStream fos = new FileOutputStream(targetFileName);

		int readByteNo;
		byte[] readBytes = new byte[100];
		while( (readByteNo = fis.read(readBytes)) != -1 ) {
			fos.write(readBytes, 0, readByteNo);
		}

		fos.flush();
		fos.close();
		fis.close();

		System.out.println("복사가 잘 되었습니다.");
	}
}
```

18.4.4 FileReader

■ 텍스트 파일로부터 데이터를 읽어 들일 때 사용
- 문자 단위로 읽기 때문에 텍스트가 아닌 그림, 오디오, 비디오 등의 파일은 읽을 수 없다.

```
//객체 생성 방법1
FileReader fr = new FileReader("C:/Temp/file.txt");

//객체 생성 방법2
File file = new File("C:/Temp/file.txt");
FileReader fr = new FileReader(file);
```

[FileReaderExample.java] 텍스트 파일 읽기

```
package sec04.exam04_file_reader;

import java.io.FileReader;
```

```
public class FileReaderExample {
	public static void main(String[] args) throws Exception {
		FileReader fr = new FileReader("C:/temp/FileReaderExample.java");

		int readCharNo;
		char[] cbuf = new char[100];
		while ((readCharNo = fr.read(cbuf)) != -1) {
			String data = new String(cbuf, 0, readCharNo); //cbuf 배열을 문자열로 생성
			System.out.print(data);
		}
		fr.close();
	}
}
```

18.4.5 FileWriter

■ 텍스트 파일에 문자 데이터를 저장할 때 사용
- 텍스트가 아닌 그림, 오디오, 비디오 등의 데이터를 파일로 저장 불가

■ 객체 생성 방법
- 파일이 이미 존재할 경우, 데이터를 출력하게 되면 파일을 덮어쓰게 되므로 기존의 파일 내용은 사리지게 된다.
- 기존 파일 내용 끝에 데이터를 추가할 경우에는 FileWriter 생성자에 두 번째 매개값으로 true를 주면 된다.
  예) FileWriter fw = new FileWriter("C:/Temp/file.txt", true);

[FileWriterExample.java] 문자열을 파일에 저장

```
package sec04.exam04_file_writer;

import java.io.File;
import java.io.FileWriter;

public class FileWriterExample {
	public static void main(String[] args) throws Exception {
		File file = new File("C:/Temp/file.txt");
		FileWriter fw = new FileWriter(file, true);
		fw.write("FileWriter는 한글로된 " + "\r\n");
		fw.write("문자열을 바로 출력할 수 있다." + "\r\n");
		fw.flush();
		fw.close();
		System.out.println("파일에 저장되었습니다.");
	}
}
```

18.5 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 스트림은 또 다른 보조 스트림에도 연결되어 스트림 체인을 구성할 수 있다.

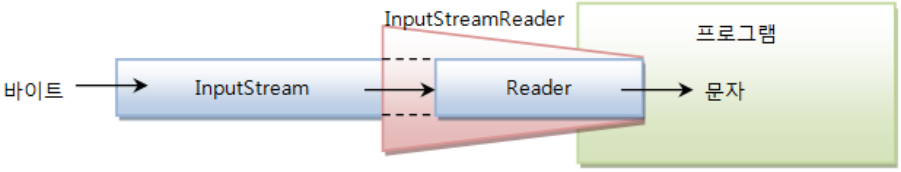
- 예를 들어 문자 변환 보조 스트림인 InputStreamReader를 다시 성능 향상 보조 스트림인 BufferedReader에 연결하는 코드는 다음과 같다.

```
InputStream is = System.in;
InputStreamReader reader = new InputStreamReader(is);
BufferedReader br = new BufferedReader(reader);
```

18.5.1 문자 변환 보조 스트림

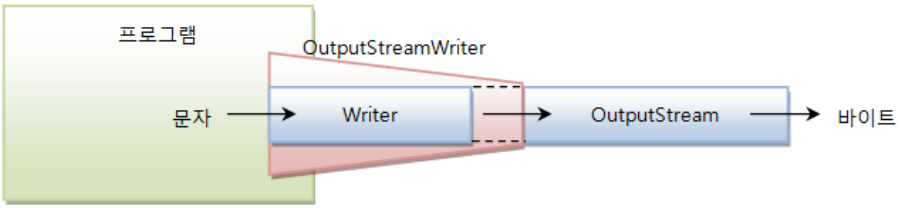
■ 소스 스트림이 바이트 기반 스트림이지만 데이터가 문자일 경우 사용
- Reader와 Writer는 문자 단위로 입출력 -> 바이트 기반 스트림보다 편리
- 문자셋의 종류를 지정할 수 있기 때문에 다양한 문자 입출력 가능

(1) InputStreamReader




```
///콘솔 입력
InputStream is = System.in;
Reader reader = new InputStreamReader(is);

//파일 입력
FileInputStream fis = FileInputStream("C:/Temp/file.txt");
Reader reader = new InputStreamReader(fis);
```

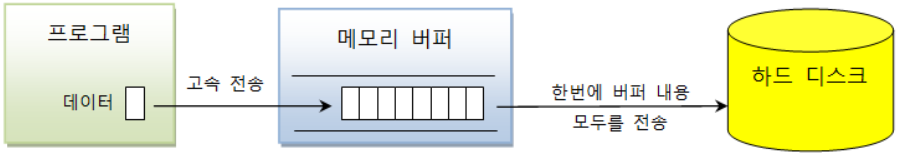
(2) OutputStreamWriter



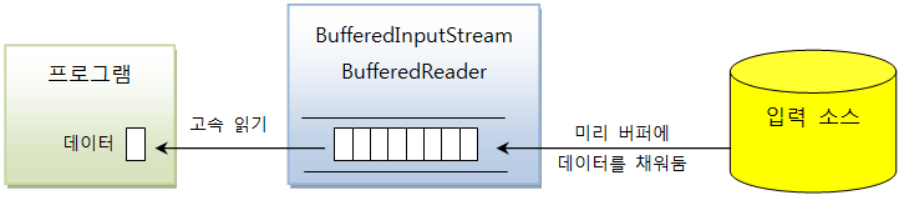

```
//파일 출력
FileOutputStream fos = FileOutputStream("C:/Temp/file.txt");
Writer writer = new OutputStreamWriter(fos);
```

18.5.2 성능 향상 보조 스트림

- 입출력 성능에 영향을 미치는 입출력 소스: 하드 디스크, 느린 네트워크
- 버퍼를 이용한 해결: 입출력 소스와 직접 작업하지 않고 버퍼(buffer)와 작업 -> 실행 성능 향상



(1) BufferedInputStream과 BufferedReader




```
BufferedInputStream bis = new BufferedInputStream(바이트입력스트림);
BufferedReader br = new BufferedReader(문자입력스트림);
```

[BufferedInputStreamExample.java] 버퍼 사용여부에 따른 성능비교

```
package sec05.exam03_bufferedinputstream;

import java.io.BufferedInputStream;
import java.io.FileInputStream;

public class BufferedInputStreamExample {
	public static void main(String[] args) throws Exception {
		long start = 0;
		long end = 0;

		FileInputStream fis1 = new FileInputStream("c:/temp/forest.jpg");
		start = System.currentTimeMillis();
		while (fis1.read() != -1) {
		}
```

```
                    end = System.currentTimeMillis();
                    System.out.println("사용하지 않았을 때: " + (end - start) + "ms");
                    fis1.close();

                    FileInputStream fis2 = new FileInputStream("c:/temp/forest.jpg");
                    BufferedInputStream bis = new BufferedInputStream(fis2);
                    start = System.currentTimeMillis();
                    while (bis.read() != -1) {
                    }
                    end = System.currentTimeMillis();
                    System.out.println("사용했을 때: " + (end - start) + "ms");
                    bis.close();
                    fis2.close();
          }
}
```

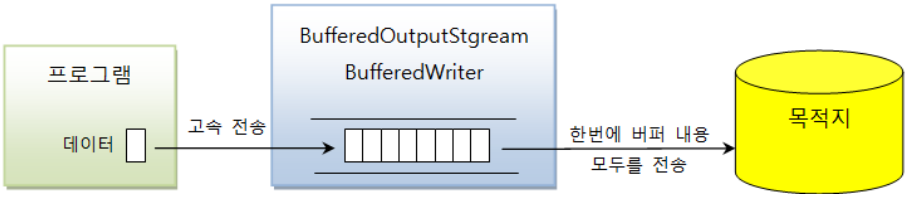
[과제] 라인 번호를 출력

■ 소스 파일을 읽고 라인 번호를 추가시켜 출력하는 프로그램을 작성하라. (힌트: `while ((rowData = br.readLine()) != null) { System.out.println(++rowNumber + ": " + rowData); })`

[AddLineNumberExample.java] 라인 번호를 출력

```
01    package verify.exam07;
02
03    import java.io.BufferedReader;
04    import java.io.FileReader;
05
06    public class AddLineNumberExample {
07              public static void main(String[] args) throws Exception {
08                        String filePath =
09    "C:/JavaProgramming/source/chap18/src/sec05/exam04_bufferedreader/BufferedReaderExample.java";
10
11                        // 작성 위치
12
13              }
14    }
```

(2) BufferedOutputStream과 BufferedWriter




```
BufferedOutputStream bos = new BufferedOutputStream(바이트출력스트림);
BufferedWriter bw = new BufferedWriter(문자출력스트림);
```

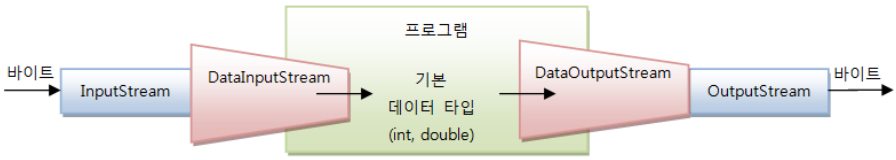
[BufferedOutputStreamExample.java] 버퍼를 사용했을 때의 성능 테스트

```
package sec05.exam05_bufferedoutputstream;

import java.io.BufferedInputStream;
```

```java
import java.io.BufferedOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;

public class BufferedOutputStreamExample {
	public static void main(String[] args) throws Exception {
		FileInputStream fis = null;
		FileOutputStream fos = null;
		BufferedInputStream bis = null;
		BufferedOutputStream bos = null;

		int data = -1;
		long start = 0;
		long end = 0;

		fis = new FileInputStream("C:/temp/forest.jpg");
		bis = new BufferedInputStream(fis);
		fos = new FileOutputStream("C:/temp/forest1.jpg");
		start = System.currentTimeMillis();
		while ((data = bis.read()) != -1) {
			fos.write(data);
		}
		fos.flush();
		end = System.currentTimeMillis();
		fos.close();
		bis.close();
		fis.close();
		System.out.println("사용하지 않았을 때: " + (end - start) + "ms");

		fis = new FileInputStream("C:/temp/forest.jpg");
		bis = new BufferedInputStream(fis);
		fos = new FileOutputStream("C:/temp/forest1.jpg");
		bos = new BufferedOutputStream(fos);
		start = System.currentTimeMillis();
		while ((data = bis.read()) != -1) {
			bos.write(data);
		}
		bos.flush();
		end = System.currentTimeMillis();
		bos.close();
		fos.close();
		bis.close();
		fis.close();
		System.out.println("사용했을 때: " + (end - start) + "ms");
	}
}
```

18.5.3 기본 타입 입출력 보조 스트림

■ 바이트 스트림은 바이트 단위로 입출력하기 때문에 자바의 기본 데이터 타입인 boolean, char, short, int, long, float, double 단위로 입출력할 수 없다. 그러나 DataInputStream과 DataOutputStream 보조 스트림을 연결하면 기본 데이터 타입으로 입출력이 가능하다.

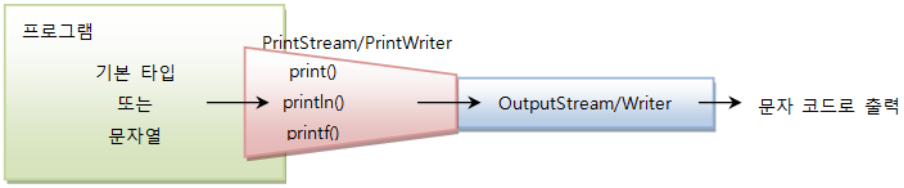


■ DataInputStream과 DataOutputStream이 제공하는 메소드들

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)

18.5.4 프린터 보조 스트림

■ PrintStream과 PrintWriter는 프린터와 유사하게 출력하는 print(), println() 메소드를 가지고 있는 보조 스트림이다.



- println()과 print() 메소드의 오버로딩

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()

- printf() 메소드는 형식화된 문자열(format string)을 출력한다.

형식화된 문자		설명	출력 형태
정수	%d	정수	123
	%6d	6자리 정수. 왼쪽 빈자리 공백	___123
	%-6d	6자리 정수. 오른쪽 빈자리 공백	123___
	%06d	6자리 정수. 왼쪽 빈자리 0 채움	000123
실수	%10.2f	소수점 이상 7자리, 소수점 이하 2자리. 왼쪽 빈자리 공백	___123.45
	%-10.2f	소수점 이상 7자리, 소수점 이하 2자리. 오른쪽 빈자리 공백	123.45___
	%010.2f	소수점 이상 7자리, 소수점 이하 2자리. 왼쪽 빈자리 0 채움	0000123.45
문자열	%s	문자열	abc
	%6s	6자리 문자열. 왼쪽 빈자리 공백	___abc
	%-6s	6자리 문자열. 오른쪽 빈자리 공백	abc___

날짜	%tF	%tY-%tm-%td	2010-01-06
	%tY	4자리 년	2010
	%ty	2자리 년	10
	%tm	2자리 월	01
	%td	2자리 일	06
	%tH	2자리 시(0~23)	08
	%tl	시(0~12)	8
	%tM	2자리 분	06
	%tS	2자리 초	24
특수 문자	/t	탭(tab)	
	/n	줄바꿈	
	%%	%	%

[PrintfExample.java] printf() 메소드 사용 방법

```
01    package sec05.exam07_printstream;
02
03    import java.util.Date;
04
05    public class PrintfExample {
06              public static void main(String[] args) {
07                        System.out.printf("상품의 가격:%d원\n", 123);
08                        System.out.printf("상품의 가격:%6d원\n", 123);
09                        System.out.printf("상품의 가격:%-6d원\n", 123);
10                        System.out.printf("상품의 가격:%06d원\n", 123);
11
12                        System.out.printf("반지름이 %d인 원의 넓이:%10.2f\n", 10, Math.PI*10*10);
13
14                        System.out.printf("%6d | %-10s | %10s\n", 1, "홍길동", "도적");
15
16                        Date now = new Date();
17                        System.out.printf("오늘은 %tY년 %tm월 %td일 입니다\n", now, now, now);
18                        System.out.printf("오늘은 %1$tY년 %1$tm월 %1$td일 입니다\n", now);
19                        System.out.printf("현재 %1$tH시 %1$tM분 %1$tS초 입니다\n", now);
20              }
21    }
```

18.5.5 객체 입출력 보조 스트림

■ 객체를 파일 또는 네트워크로 입출력할 수 있는 기능 제공

■ 객체 직렬화(serialization)
- 객체는 문자가 아니므로 바이트 기반 스트림으로 출력해야 한다.
- 객체를 출력하기 위해서는 객체의 데이터(필드값)를 일렬로 늘어선 연속적인 바이트로 변경해야 한다.

(1) ObjectInputStream, ObjectOutputStream

■ ObjectOutputStream은 바이트 출력 스트림과 연결되어 객체를 직렬화하는 역할을 하고, ObjectInputStream은 바이트 입력 스트림과 연결되어 객체로 역직렬화하는 역할을 한다.

[ObjectInputOutputStreamExample.java] 다양한 객체를 쓰고 읽기

```
package sec05.exam08_objectinputstream_objectoutputstream;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

public class ObjectInputOutputStreamExample {
	public static void main(String[] args) throws Exception {
		FileOutputStream fos = new FileOutputStream("C:/Temp/Object.dat");
		ObjectOutputStream oos = new ObjectOutputStream(fos);

		oos.writeObject(new Integer(10));
		oos.writeObject(new Double(3.14));
		oos.writeObject(new int[] { 1, 2, 3 });
		oos.writeObject(new String("홍길동"));

		oos.flush();
		oos.close();
		fos.close();

		FileInputStream fis = new FileInputStream("C:/Temp/Object.dat");
		ObjectInputStream ois = new ObjectInputStream(fis);

		Integer obj1 = (Integer) ois.readObject();
		Double obj2 = (Double) ois.readObject();
		int[] obj3 = (int[]) ois.readObject();
		String obj4 = (String) ois.readObject();

		ois.close();
		fis.close();

		System.out.println(obj1);
		System.out.println(obj2);
		System.out.println(obj3[0] + "," + obj3[1] + "," + obj3[2]);
		System.out.println(obj4);
	}
}
```

(2) 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

```
public class XXX implements Serializable {
   public int field1;
   protected int field2;
   int field3;
   private int field4;
   public static int field5;
   transcient int field6;   // static 또는 transcient 키워드가 붙은 필드는 직렬화에서 제외
}
```

(3) serialVersionUID 필드

- 직렬화된 객체를 역직렬화 할 때는 직렬화 했을 때와 같은 클래스 사용해야 한다.
- 클래스의 이름이 같더라도 클래스의 내용이 변경되면, 역직렬화는 실패한다.
- serialVersionUID
  - 같은 클래스임을 알려주는 식별자 역할
  - Serializable 인터페이스 구현 : 컴파일 시 자동적으로 serialVersionUID 정적 필드 추가
  - 재컴파일하면 serialVersionUID의 값 변경
- 불가피한 수정 있을 경우 명시적으로 serialVersionUID 선언하면 된다.

```
// Serializable 인터페이스를 구현할 XXX 클래스에 명시적으로 선언
public class XXX implements Serializable {
   static final long serialVersionUID = 정수값;
   ...
}

// 자동으로 생성시켜주는 명령어
c:\>"%JAVA_HOME%\bin\serialver.exe" XXX
XXX:    private static final long serialVersionUID = -1609400435821254064L;
```

(4) writeObject()와 readObject() 매소드

- 부모 클래스가 Serializable 구현하지 않고 자식 클래스가 Serializable 구현한 경우 -> 부모 필드는 직렬화에서 제외
- 위 경우에 부모 클래스의 필드를 직렬화하려면 ...
  - 방법1: 부모 클래스가 Serializable 인터페이스를 구현하도록 한다.
  - 방법2: 자식 클래스에서 writeObject() 와 readObject() 메소드를 선언해서 부모 객체의 필드를 직접 출력시킨다.
- writeObject() 메소드는 직렬화될 때 자동으로 호출되고, readObject() 메소드는 역직렬화될 때 자동적으로 호출된다. 주의할 점은 접근 제한자가 private가 아니면 자동 호출되지 않기 때문에 반드시 private를 붙여주어야 한다.

```
private void writeObject(ObjectOutputStream out) throws IOException {
   out.writeXXX(부모필드);    // 부모 객체의 필드값을 출력함
   ...
   out.defaultWriteObject();  // 자식 객체의 필드값을 직렬화
}

private void readObject(ObjectInputStream in) throws IOException, ClassNotFoundException {
   부모필드 = in.readXXX();   // 부모 객체의 필드값을 읽어옴
   ...
   in.defaultReadObject();    // 자식 객체의 필드값을 역직렬화
}
```

[Parent.java] Serializable를 구현하지 않은 부모 클래스

```
package sec05.exam08_objectinputstream_objectoutputstream;

public class Parent {
          public String field1;
}
```

[Child.java] 직렬화되지 않은 부모 클래스의 필드 처리

```
package sec05.exam08_objectinputstream_objectoutputstream;

import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;

public class Child extends Parent implements Serializable {
          public String field2;

          private void writeObject(ObjectOutputStream out) throws IOException {
             out.writeUTF(field1);
             out.defaultWriteObject();
          }

          private void readObject(ObjectInputStream in) throws IOException, ClassNotFoundException {
             field1 = in.readUTF();
             in.defaultReadObject();
          }
}
```

[NonSerializableParentExample.java] 직렬화 및 역직렬화

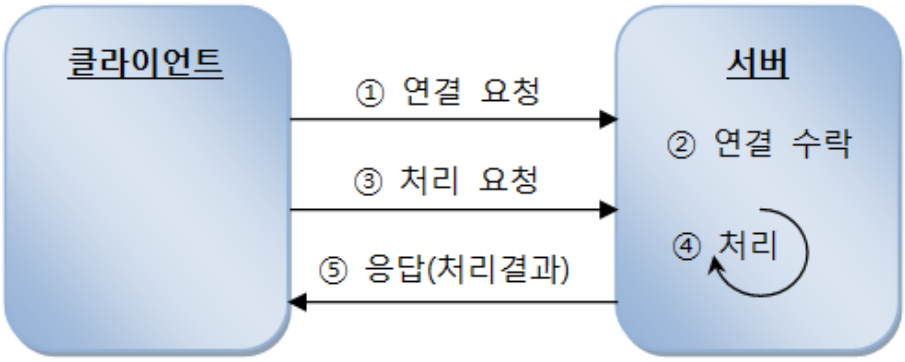
```
package sec05.exam08_objectinputstream_objectoutputstream;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

public class NonSerializableParentExample {
          public static void main(String[] args) throws Exception {
                    FileOutputStream fos = new FileOutputStream("C:/Temp/Object.dat");
                    ObjectOutputStream oos = new ObjectOutputStream(fos);
                    Child child = new Child();
                    child.field1 = "홍길동";
                    child.field2 = "홍삼원";
                    oos.writeObject(child);
                    oos.flush(); oos.close(); fos.close();

                    FileInputStream fis = new FileInputStream("C:/Temp/Object.dat");
                    ObjectInputStream ois = new ObjectInputStream(fis);
                    Child v = (Child) ois.readObject();
                    System.out.println("field1: " + v.field1);
                    System.out.println("field2: " + v.field2);
                    ois.close(); fis.close();
          }
}
```

18.6 네트워크 기초

18.6.1 서버와 클라이언트

■ 서버: 서비스를 제공하는 프로그램
- 웹 서버, FTP서버, DBMS, 메신저 서버
- 클라이언트의 연결을 수락하고, 요청 내용 처리한 후 응답 보내는 역할

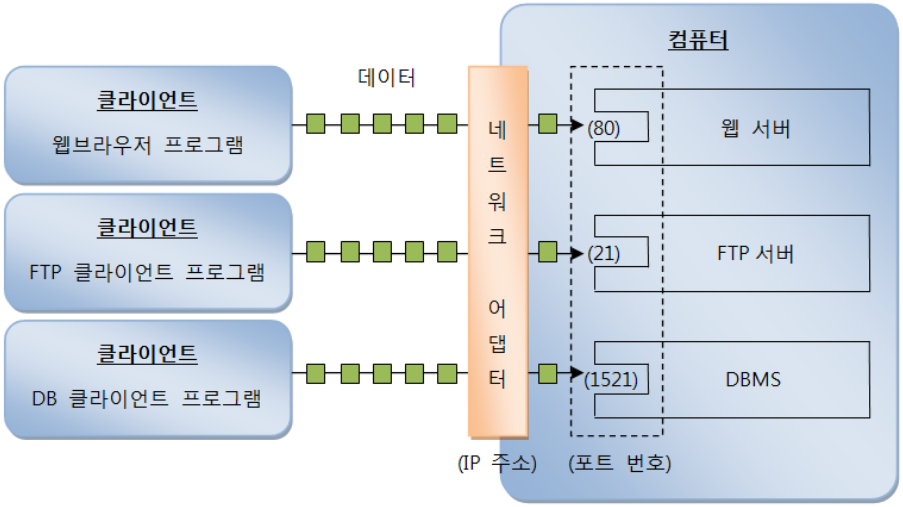
■ 클라이언트: 서비스를 받는 프로그램
- 웹 브라우저, FTP 클라이언트, 메신저
- 네트워크 데이터를 필요로 하는 모든 애플리케이션이 해당(모바일 앱 포함)



18.6.2 IP 주소와 포트(Port)

■ IP(Internet Protocol) 주소
- 네트워크상에서 컴퓨터를 식별하는 번호
- 네트워크 어댑터(랜 (Lan) 카드) 마다 할당
- IP 주소 확인 법 - 명령 프롬프트 (cmd.exe) 사용
- xxx.xxx.xxx.xxx 형식으로 표현 (xxx는 0~255 사이의 정수)

■ 포트(Port)
- 같은 컴퓨터 내에서 프로그램을 식별하는 번호
- 클라이언트는 서버 연결 요청 시 IP 주소와 Port 같이 제공
- 0~65535 범위의 값을 가짐
- 포트 범위는 세 가지로 구분



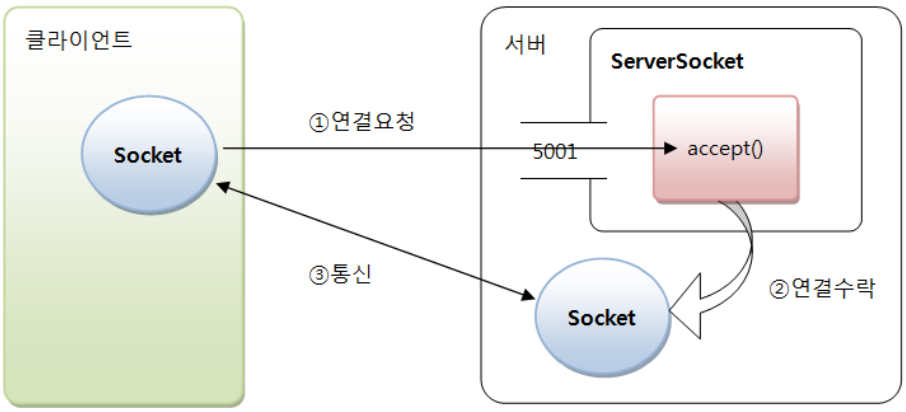
18.6.3 InetAddress로 IP 주소 얻기

- java.net.InetAddress
  - IP 주소 표현한 클래스
  - 로컬 컴퓨터의 IP 주소
  - 도메인 이름을 DNS에서 검색한 후 IP 주소를 가져오는 기능 제공

18.7 TCP 네트워킹

- 연결 지향적 프로토콜 -> 시간 소요
- 통신 선로 고정 -> 전송 속도 느려질 수 있음
- 데이터를 정확하고 안정적으로 전달
- TCP 네트워킹을 위해 java.net.ServerSocket, java.net.Socket 클래스를 제공하고 있다.

18.7.1 ServerSocket과 Socket의 용도



18.7.2 ServerSocket 생성과 연결 수락

- ServerSocket 생성과 포트 바인딩
  - 생성자에 바인딩 포트 대입하고 객체 생성
- 연결 수락
  - accept() 메소드는 클라이언트가 연결 요청 전까지 블로킹 → 대기
  - 연결된 클라이언트 IP 주소 얻기
- ServerSocket 포트 언바인딩
  - 더 이상 클라이언트 연결 수락 필요 없는 경우

```
//ServerSocket 생성과 포트 바인딩
```

```
ServerSocket serverSocket = new ServerSocket(5001);
//ServerSocket serverSocket = new ServerSocket();
//serverSocket.bind(new InetSocketAddress("localhost", 5001));

//연결 수락
Socket socket = serverSocket.accept();

//ServerSocket 포트 언바인딩
serverSocket.close();
```

[ServerExample.java] 연결 수락

```
package sec07.exam01_serversocket_socket;

import java.io.IOException;
import java.net.InetSocketAddress;
import java.net.ServerSocket;
import java.net.Socket;

public class ServerExample {
	public static void main(String[] args) {
		ServerSocket serverSocket = null;
		try {
			serverSocket = new ServerSocket(); // ServerSocket 생성
			serverSocket.bind(new InetSocketAddress("localhost", 5001));
			while (true) {
				System.out.println("[연결 기다림]");
				Socket socket = serverSocket.accept(); // 클라이언트 연결 수락
				InetSocketAddress isa = (InetSocketAddress)
socket.getRemoteSocketAddress();
				System.out.println("[연결 수락함] " + isa.getHostName());
			}
		} catch (Exception e) {
		}

		if (!serverSocket.isClosed()) { // ServerSocket이 닫혀있지 않을 경우
			try {
				serverSocket.close(); // ServerSocket 닫기
			} catch (IOException e1) {
			}
		}
	}
}
```

18.7.3 Socket 생성과 연결 요청

■ Socket 생성 및 연결 요청
- java.net.Socket 이용
- 서버의 IP 주소와 바인딩 포트 번호를 제공하면 생성과 동시에 사용가능

■ 연결 끊기
- Exception 처리 필요

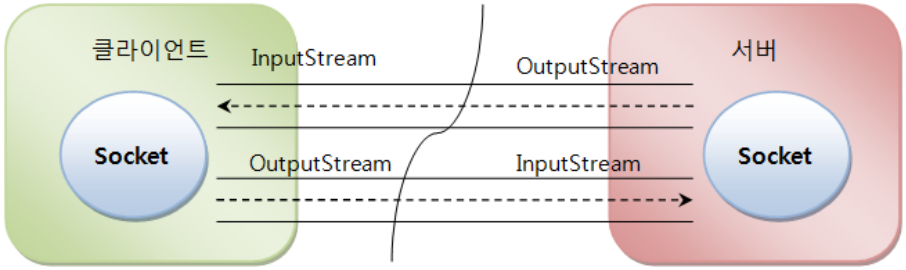
[ClientExample.java] 연결 요청

```
package sec07.exam01_serversocket_socket;

import java.io.IOException;
```

```
import java.net.InetSocketAddress;
import java.net.Socket;

public class ClientExample {
	public static void main(String[] args) {
		Socket socket = null;
		try {
			socket = new Socket();
			System.out.println("[연결 요청]");
			socket.connect(new InetSocketAddress("localhost", 5001));
			System.out.println("[연결 성공]");
		} catch (Exception e) {
		}

		if (!socket.isClosed()) {
			try {
				socket.close();
			} catch (IOException e1) {
			}
		}
	}
}
```

18.7.4 Socket 데이터 통신

■ Socket 객체로 부터 입출력 스트림 얻기




```
//데이터 보내기
String data = "보낼 데이터";
byte[] byteArr = data.getBytes("UTF-8");
OutputStream outputStream = socket.getOutputStream();
outputStream.write(byteArr);
outputStream.flush();

//데이터 받기
byte[] byteArr = new byte[100];
InputStream inputStream = socket.getInputStream();
int readByteCount = inputStream.read(byteArr);
String data = new String(byteArr, 0, readByteCount, "UTF-8");
```

[ClientExample.java] 데이터 보내고 받기

```
package sec07.exam02_data_read_write;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.net.Socket;
```

```
public class ClientExample {
	public static void main(String[] args) {
		Socket socket = null;
		try {
			socket = new Socket();
			System.out.println("[연결 요청]");
			socket.connect(new InetSocketAddress("localhost", 5001));
			System.out.println("[연결 성공]");

			byte[] bytes = null;
			String message = null;

			OutputStream os = socket.getOutputStream();
			message = "Hello Server";
			bytes = message.getBytes("UTF-8");
			os.write(bytes);
			os.flush();
			System.out.println("[데이터 보내기 성공]");

			InputStream is = socket.getInputStream();
			bytes = new byte[100];
			int readByteCount = is.read(bytes);
			message = new String(bytes, 0, readByteCount, "UTF-8");
			System.out.println("[데이터 받기 성공]: " + message);

			os.close();
			is.close();
		} catch (Exception e) {
		}

		if (!socket.isClosed()) {
			try {
				socket.close();
			} catch (IOException e1) {
			}
		}
	}
}
```

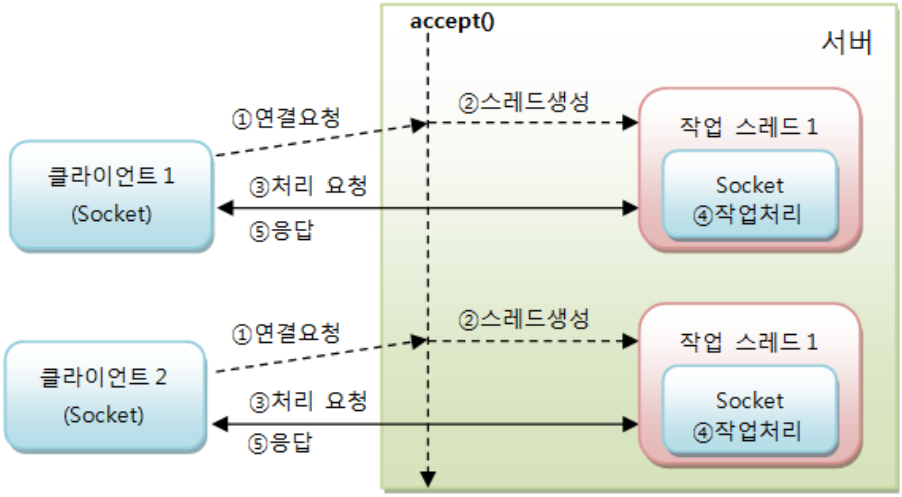
[ServerExample.java] 데이터 받고 보내기

```
package sec07.exam02_data_read_write;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.net.ServerSocket;
import java.net.Socket;

public class ServerExample {
	public static void main(String[] args) {
		ServerSocket serverSocket = null;
		try {
			serverSocket = new ServerSocket();
			serverSocket.bind(new InetSocketAddress("localhost", 5001));
			while (true) {
				System.out.println("[연결 기다림]");
				Socket socket = serverSocket.accept();
				InetSocketAddress isa = (InetSocketAddress)
socket.getRemoteSocketAddress();
```

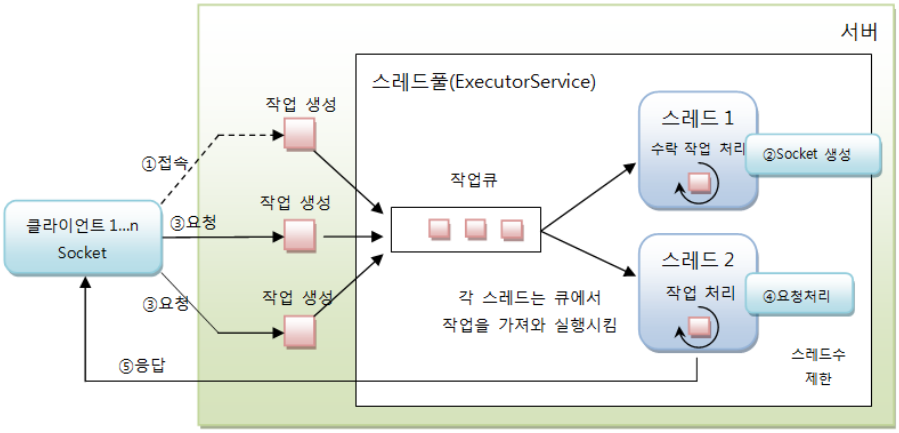
```
                                System.out.println("[연결 수락함] " + isa.getHostName());

                                byte[] bytes = null;
                                String message = null;

                                InputStream is = socket.getInputStream();
                                bytes = new byte[100];
                                int readByteCount = is.read(bytes);
                                message = new String(bytes, 0, readByteCount, "UTF-8");
                                System.out.println("[데이터 받기 성공]: " + message);

                                OutputStream os = socket.getOutputStream();
                                message = "Hello Client";
                                bytes = message.getBytes("UTF-8");
                                os.write(bytes);
                                os.flush();
                                System.out.println("[데이터 보내기 성공]");

                                is.close();
                                os.close();
                                socket.close();
                        }
                } catch (Exception e) {
                }

                if (!serverSocket.isClosed()) {
                        try {
                                serverSocket.close();
                        } catch (IOException e1) {
                        }
                }
        }
}
```

18.7.5 스레드 병렬 처리

■ Accept(), connect(), read(), write()는 별도 작업 스레드 생성



■ 스레드풀 사용해 스레드 수 관리
- 스레드풀은 스레드 수 제한해 사용
- 갑작스런 클라이언트의 폭증은 작업 큐의 작업량만 증가



18.7.6 채팅 서버 구현

(1) 서버 클래스 구조

■ 다음은 서버 클래스의 구조를 보여준다.

```
01  // 채팅 서버
02  public class ServerExample extends Application {  //JavaFX 메인 클래스로 만들기 위해 상속받음.
03     ExecutorService executorService;  //스레드풀인 ExecutorService 필드가 선언.
04     ServerSocket serverSocket;  //클라이언트의 연결을 수락하는 ServerSocket 필드가 선언
05     List<Client> connections = new Vector<Client>();  //연결된 클라이언트를 저장
06
07     void startServer() { //서버 시작 코드 }
08     void stopServer() { //서버 종료 코드 }
09
10     class Client { //데이터 통신 코드 }
11
12     //UI 생성 코드
13  }
```

(2) startServer() 메소드

■ Executor Service 생성, ServerSocket 생성 및 포트 바인딩, 연결 수락 코드가 있다.

```
01  void startServer() {
02     // Executor Service 생성
03     executorService = Executors.newFixedThreadPool(
04        Runtime.getRuntime().availableProcessors()
05     );
06
07     // ServerSocket 생성 및 포트 바인딩
08     try {
09        serverSocket = new ServerSocket();
10        serverSocket.bind(new InetSocketAddress("localhost", 5001));
11     } catch(Exception e) {
12        if(!serverSocket.isClosed()) { stopServer(); }
13        return;
14     }
15
16     // 연결 수락
17     Runnable runnable = new Runnable() {
```

```
18         @Override
19         public void run() {
20            Platform.runLater(()->{
21               displayText("[서버 시작]");
22               btnStartStop.setText("stop");
23            });
24            while(true) {
25               try {
26                  Socket socket = serverSocket.accept();
27                  String message = "[연락 수락: " + socket.getRemoteSocketAddress() +
28                                   ": " + Thread.currentThread().getName() + "]";
29                  Platform.runLater(()->displayText(message));
30
31                  Client client = new Client(socket);
32                  connections.add(client);
33                  Platform.runLater(()->displayText("[연결 개수: " + connections.size() + "]"));
34               } catch (Exception e) {
35                  if(!serverSocket.isClosed()) { stopServer(); }
36                  break;
37               }
38            }
39         }
40      }
41      executorService.submit(runnable);
42   }
```

(3) stopServer() 메소드

■ 연결된 모든 Socket 닫기, ServerSocket 닫기, ExecutorService 종료 코드가 있다.

```
01   void stopServer() {
02      try {
03         // 모든 Socket 닫기
04         Iterator<Client> iterator = connections.iterator();
05         while(iterator.hasNext()) {
06            Client client = iterator.next();
07            client.socket.close();
08            iterator.remove();
09         }
10         // ServerSocket 닫기
11         if(serverSocket!=null && !serverSocket.isClosed()) {
12            serverSocket.close();
13         }
14         // ExecutorService 종료
15         if(executorService!=null) && !executorService.isShutdown() {
16            executorService.shutdown();
17         }
18         Platform.runLater(()->{
19            displayText("[서버 멈춤]");
20            btnStartStop.setText("start");
21         });
22      } catch (Exception e) { }
23   }
```

(4) Client 클래스

■ 다수 클라이언트 관리 → 연결 수락 시 마다 Client 인스턴스를 생성해서 관리하는 것이 좋다.

■ 다음은 Client 클래스의 구조를 보여준다.

```
01  class Client {
02     Socket socket;
03
04     Client(Socket socket) {
05        this.socket = socket;
06        receive();
07     }
08
09     void receive() { //데이터 받기 코드 }
10     void send(String data) { //데이터 전송 코드 }
11  }
```

■ 다음은 클라이언트의 데이터를 받는 receive() 메소드 코드이다.

```
01  void receive() {
02     Runnable runnable = new Runnable() {  //받기 작업 생성
03        @Override
04        public void run() {
05           try {
06              while(true) {
07                 byte[] byteArr = new byte[100];
08                 InputStream inputStream = socket.getInputStream();
09                 //클라이언트가 비정상 종료를 했을 경우 IOException 발생
10                 int readByteCount = inputStream.read(byteArr);  //데이터 받기
11
12                 //클라이언트가 정상적으로 Socket의 close()를 호출했을 경우
13                 if(readByteCount == -1) { throw new IOException(); }
14
15                 String message = "[요청 처리: " + socket.getRemoteSocketAddress() +
16                                  ": " + Thread.currentThread().getName() + "]";
17                 Platform.runLater(()->displayText(message());
18
19                 String data = new String(byteArr, 0, readyByteCount, "UTF-8");  //문자열로 변환
20                 for(Client client : connections) {
21                    client.send(data);  //모든 클라이언트에게 보냄
22                 }
23              }
24           } catch(Exception e) {
25              try {
26                 String message = "[클라이언트 통신 안됨: " +
27                                  socket.getRemoteSocketAddress() + ": " +
28                                  Thread.currentThread().getName() + "]";
29                 Platform.runLater(()->displayText(message));
30                 connections.remove(Client.this);
31                 socket.close();
32              } catch (IOException e2) {}
33           }
34        }
35     };
36     executorService.submit(runnable);  //스레드풀에서 처리
37  }
```

■ 클라이언트로 메시지를 보내는 send(String data) 메소드 코드를 보자.

```
01  void send(String data) {
02     Runnable runnable = new Runnable() {  //보내기 작업 생성
03        @Override
04        public void run() {
05           try {  //클라이언트로 데이터 보내기
```

```
06              byte[] byteArr = data.getBytes("UTF-8");
07              OutputStream outputStream = socket.getOutputStream();
08              outputStream.write(byteArr);
09              outputStream.flush();
10           } catch(Exception e) {
11              try {
12                 String message = "[클라이언트 통신 안됨: " +
13                                  socket.getRemoteSocketAddress() + ": " +
14                                  Thread.currentThread().getName() + "]";
15                 Platform.runLater(()->displayText(message));
16                 connections.remove(Client.this);
17                 socket.close();
18              } catch (IOException e2) {}
19           }
20        }
21     };
22     executorService.submit(runnable);  //스레드풀에서 처리
23  }
```

(5) UI 생성 코드

■ 다음은 javaFX 이용한 UI 생성 코드를 보여준다.

```
01            TextArea txtDisplay;
02            TextField txtInput;
03            Button btnConn, btnSend;
04
05            @Override
06            public void start(Stage primaryStage) throws Exception {
07                      BorderPane root = new BorderPane();
08                      root.setPrefSize(500, 300);
09
10                      txtDisplay = new TextArea();
11                      txtDisplay.setEditable(false);
12                      BorderPane.setMargin(txtDisplay, new Insets(0, 0, 2, 0));
13                      root.setCenter(txtDisplay);
14
15                      BorderPane bottom = new BorderPane();
16                      txtInput = new TextField();
17                      txtInput.setPrefSize(60, 30);
18                      BorderPane.setMargin(txtInput, new Insets(0, 1, 1, 1));
19
20                      btnConn = new Button("start");
21                      btnConn.setPrefSize(60, 30);
22                      btnConn.setOnAction(e -> {
23                                if (btnConn.getText().equals("start")) {
24                                          startClient();
25                                } else if (btnConn.getText().equals("stop")) {
26                                          stopClient();
27                                }
28                      });
29
30                      btnSend = new Button("send");
31                      btnSend.setPrefSize(60, 30);
32                      btnSend.setDisable(true);
33                      btnSend.setOnAction(e -> send(txtInput.getText()));
34
35                      bottom.setCenter(txtInput);
36                      bottom.setLeft(btnConn);
37                      bottom.setRight(btnSend);
38                      root.setBottom(bottom);
```

```
39
40                      Scene scene = new Scene(root);
41                      scene.getStylesheets().add(getClass().getResource("app.css").toString());
42                      primaryStage.setScene(scene);
43                      primaryStage.setTitle("Client");
44                      primaryStage.setOnCloseRequest(event -> stopClient());
45                      primaryStage.show();
46              }
47
48              void displayText(String text) {  //작업 스레드의 작업처리 내용을 출력할 때 호출하는 메소드
49                      txtDisplay.appendText(text + "\n");
50              }
51
52              public static void main(String[] args) {
53                      launch(args);
54              }
```

■ 다음과 같이 외부 CSS 클래스 선택자를 이용해서 컨테이너의 배경색을 변경한다.

[app.css] 외부 CSS 파일

```
01  /*text-area 배경색*/
02  .text-area {
03              -fx-background-color: gold;
04  }
05
06  /*scroll-pane 배경색*/
07  .text-area .scroll-pane {
08              -fx-background-color: transparent;
09  }
10
11  /*viewport 배경색*/
12  .text-area .scroll-pane .viewport{
13              -fx-background-color: transparent;
14  }
15
16  /*content 배경색*/
17  .text-area .scroll-pane .content{
18              -fx-background-color: transparent;
19  }
```

~~18.7.7 채팅 클라이언트 구현~~

[과제] 채팅 클라이언트 구현

1. 프로그램적 레이아웃 -> FXML 레이아웃으로 변형
2. 로그인 화면을 포함한 화면 이동 UI 구현

18.8 UDP 네트워킹

■ UDP(User Datagram Protocol)의 특징
  ■ 비연결 지향적 프로토콜
    - 연결 절차 거치지 않고 발신자가 일방적으로 데이터 발신하는 방식
    - TCP 보다는 빠른 전송
  ■ 통신 선로가 고정적이지 않음

- 데이터 패킷들이 서로 다른 통신 선로 통해 전달될 수 있음
- 먼저 보낸 패킷이 느린 선로 통해 전송될 경우, 나중에 보낸 패킷보다 늦게 도착 가능

■ 데이터 손실 발생 가능성
- 일부 패킷은 잘못된 선로로 전송되어 유실 가능
- 데이터 전달 신뢰성 떨어짐

■ java.net API
■ DatagramSocket, DatagramPacket

18.8.1 발신자 구현

■ 소켓 통해 데이터 패킷 전송

```
// 1. DatagramScoket 생성
DatagramSocket datagramSocket = new DatagramSocket();

// 2. DatagramPacket 생성
byte[] byteArr = data.getBytes("UTF-8");
DatagramPacket packet = new DatagramPacket(
   byteArr, byteArr.length,
   new InetSocketAddress("localhost", 5001)
);

// 3. DatagramPacket 발송
datagramSocket.send(packet);

// 4. DatagramSockt 닫기
datagramSocket.close();
```

[UdpSendExample.java] 발신자

```
01    package sec08.exam01_udp;
02
03    import java.net.DatagramPacket;
04    import java.net.DatagramSocket;
05    import java.net.InetSocketAddress;
06
07    public class UdpSendExample {
08              public static void main(String[] args) throws Exception {
09                        DatagramSocket datagramSocket = new DatagramSocket();
10
11                        System.out.println("[발신 시작]");
12
13                        for(int i=1; i<3; i++) {
14                                  String data = "메시지" + i;
15                                  byte[] byteArr = data.getBytes("UTF-8");
16                                  DatagramPacket packet = new DatagramPacket(
17                                                      byteArr, byteArr.length,
18                                                      new InetSocketAddress("localhost", 5001)
```

```
19                                  );
20
21                                  datagramSocket.send(packet);
22                                  System.out.println("[보낸 바이트 수]: " + byteArr.length + " bytes");
23                          }
24
25                          System.out.println("[발신 종료]");
26
27                          datagramSocket.close();
28                  }
29          }
```

18.8.2 수신자 구현

■ 바인딩한 특정 포트로 데이터 받아 저장

```
// 1. DatagramScoket 생성
DatagramSocket datagramSocket = new DatagramSocket(5001);

// 2. DatagramPacket 생성
DatagramPacket datagramPacket = new DatagramPacket(new byte[100], 100);

// 3. DatagramPacket 수신
datagramSocket.receive(datagramPacket);
String data = new String(datagramPacket.getData(), 0, datagramPacket.getLength(), "UTF-8");

// 4. DatagramScoket 닫기
datagramSocket.close();
```

[UdpReceiveExample.java] 수신자

```
01  package sec08.exam01_udp;
02
03  import java.net.DatagramPacket;
04  import java.net.DatagramSocket;
05
06  public class UdpReceiveExample extends Thread {
07          public static void main(String[] args) throws Exception {
08                  DatagramSocket datagramSocket = new DatagramSocket(5001);
09
10                  Thread thread = new Thread() {
11                          @Override
12                          public void run() {
13                                  System.out.println("[수신 시작]");
14                                  try {
15                                          while(true) {
16                                                  DatagramPacket packet = new
17  DatagramPacket(new byte[100], 100);
18                                                  datagramSocket.receive(packet);
19
20                                                  String data = new String(packet.getData(), 0,
21  packet.getLength(), "UTF-8");
22                                                  System.out.println("[받은 내용: " +
23  packet.getSocketAddress() + "] " + data);
24                                          }
25                                  } catch (Exception e) {
26                                          System.out.println("[수신 종료]");
27                                  }
28                          }
```

```
29                      };
30                      thread.start();
31
32                      Thread.sleep(10000);
33                      datagramSocket.close();
34              }
35      }
```