

16장. IO기반 입출력 및 네트워킹

Contents

- ❖ 1절. IO 패키지 소개
- ❖ 2절. 입력 스트림과 출력 스트림
- ❖ 3절. 콘솔(Console) 입출력
- ❖ 4절. 파일(File) 입출력
- ❖ 5절. 보조 스트림
- ❖ 6절. 네트워크 기초
- ❖ 7절. TCP 네트워킹
- ❖ 8절. UDP 네트워킹

1절. IO 패키지 소개

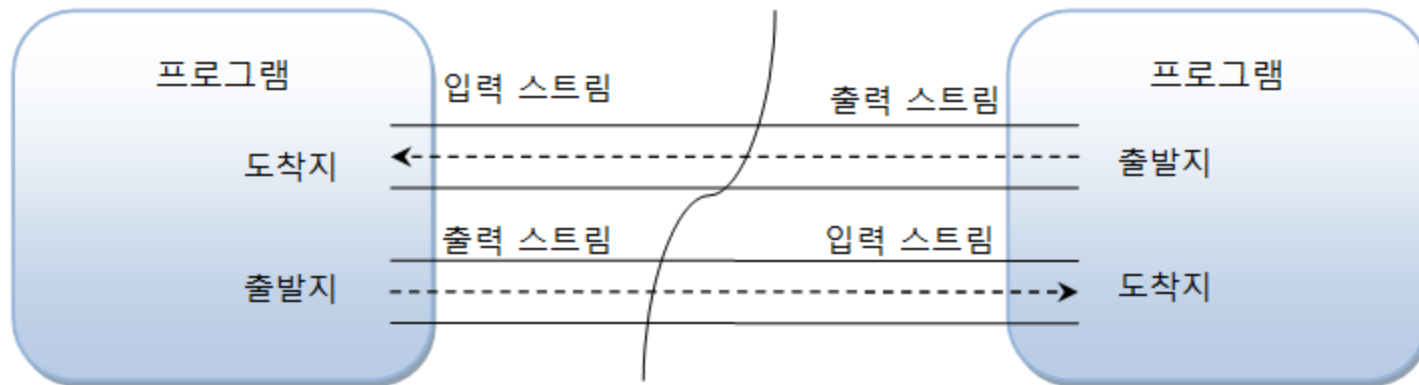
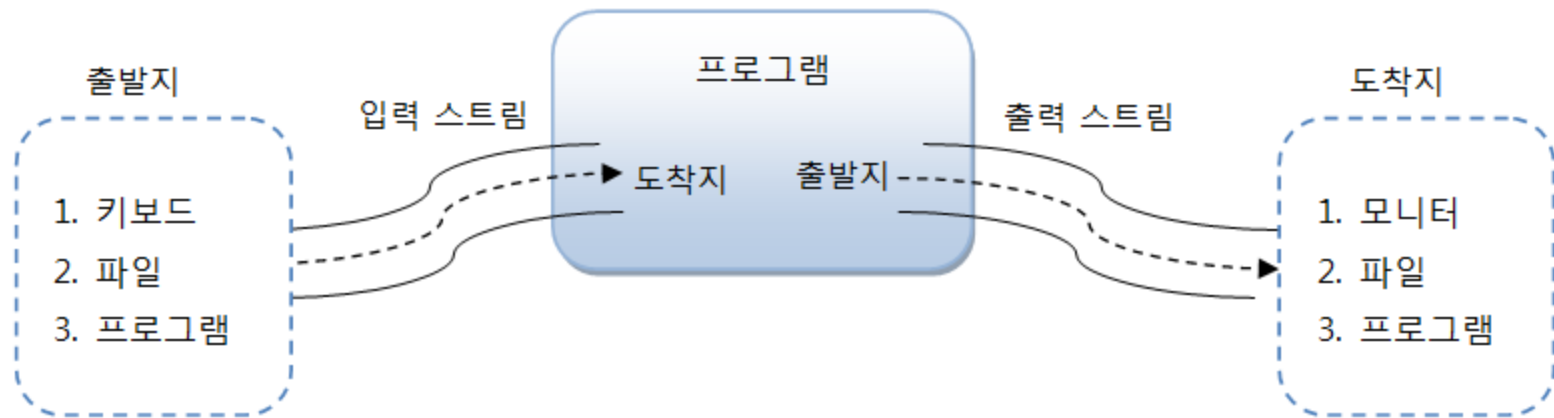
❖ java.io 패키지

■ 자바의 기본적인 데이터 입출력(IO: Input/Output) API 제공

java.io 패키지의 주요 클래스	설명
File	파일 시스템의 파일의 정보를 얻기위한 클래스
Console	콘솔로부터 문자를 입출력하기 위한 클래스
InputStream / OutputStream	바이트 단위 입출력을 위한 최상위 입출력 스트림 클래스
FileInputStream / FileOutputStream DataInputStream / DataOutputStream 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 하위 스트림 클래스

2절. 입력 스트림과 출력 스트림

❖ 입력 스트림과 출력 스트림의 개념



2절. 입력 스트림과 출력 스트림

❖ 바이트 기반 스트림과 문자 기반 스트림

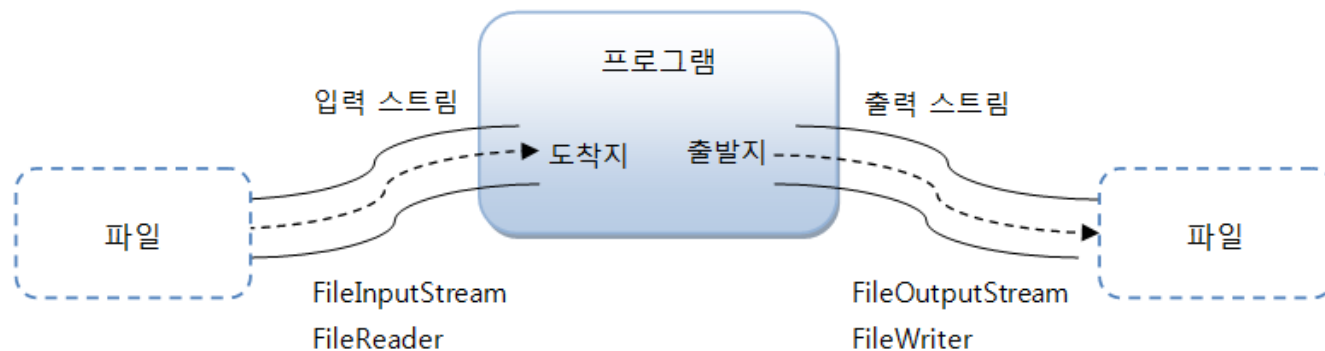
■ 바이트 기반 스트림

- 그림, 멀티미디어, 문자 등 모든 종류의 데이터를 받고 보내는 것 가능

■ 문자 기반 스트림

- 문자만 받고 보낼 수 있도록 특화

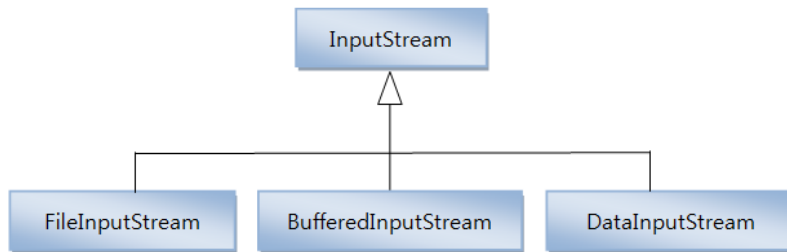
구분	바이트 기반 스트림		문자 기반 스트림	
	입력 스트림	출력 스트림	입력 스트림	출력 스트림
최상위 클래스	InputStream	OutputStream	Reader	Writer
하위 클래스 (예)	XXInputSteam (FileInputStream)	XXOutputStream (FileOutputStream)	XXReader (FileReader)	XXWriter (FileWriter)



2절. 입력 스트림과 출력 스트림

❖ InputStream

- 바이트 기반 입력 스트림의 최상위 클래스로 추상 클래스



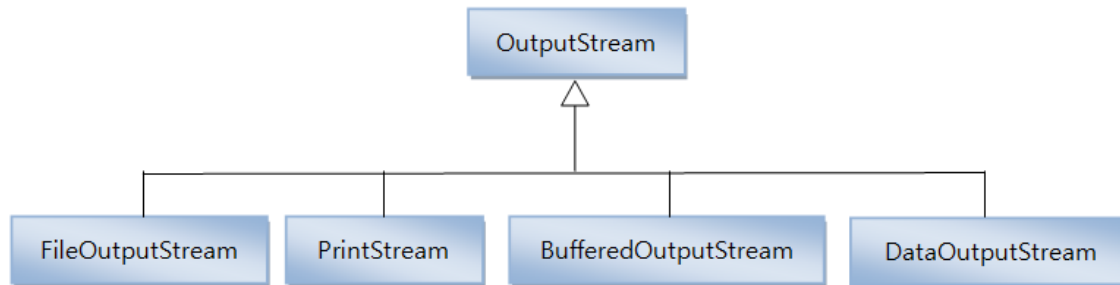
- InputStream 클래스의 주요 메소드 (p.997~999)

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.

2절. 입력 스트림과 출력 스트림

❖ OutputStream

- **바이트 기반 출력 스트림의 최상위 클래스로 추상 클래스**



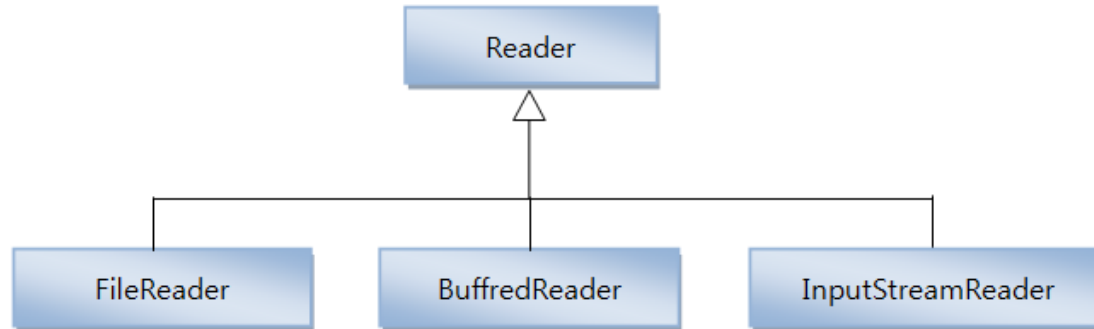
- **OutputStream의 주요 메소드 (p.1000~1002)**

리턴타입	메소드	설명
void	write(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.

2절. 입력 스트림과 출력 스트림

❖ Reader

- 문자 기반 입력 스트림의 최상위 클래스로 추상 클래스



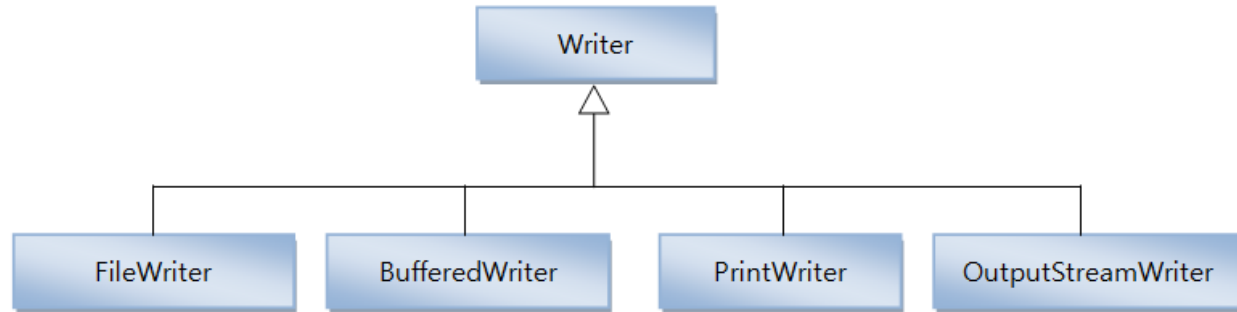
- Reader의 주요 메소드 (p.1002~1005)

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.

2절. 입력 스트림과 출력 스트림

❖ Writer

- 문자 기반 출력 스트림의 최상위 클래스로 추상 클래스



- Writer의 주요 메소드 (p.1006~1009)

리턴타입	메소드	설명
void	write(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.

3절. 콘솔 입출력

❖ 콘솔(Console)

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



3절. 콘솔 입출력

❖ 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
 - 전체 내용을 바이트 배열로 받아 String 객체 생성 후 읽기

3절. 콘솔 입출력

❖ System.out 필드

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

❖ Console 클래스

- **자바6부터 콘솔에서 입력된 문자열을 쉽게 읽을 수 있도록 제공**
 - 이클립스에서 System.console()은 null 리턴
 - 명령 프롬프트에서 반드시 실행
- **Console 클래스의 읽기 메소드**

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

3절. 콘솔 입출력

❖ 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

```
Scanner scanner = new Scanner(System.in)
```

- 제공하는 메소드

리턴타입	메소드	설명
boolean	nextBoolean()	boolean(true/false) 값을 읽는다.
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.

4절. 파일 입출력

❖ 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 제공

■ 파일 객체 생성

```
File file = new File("C:\\Temp\\file.txt");  
File file = new File("C:/Temp/file.txt");
```

■ 파일 또는 디렉토리 존재 유무 확인 메소드

```
boolean isExist = file.exists();
```

■ 파일 및 디렉토리 생성 및 삭제 메소드

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

4절. 파일 입출력

■ 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

4절. 파일 입출력

❖ 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

4절. 파일 입출력

❖ 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 덮어쓰는 단점
 - 기존 파일 내용 끝에 데이터를 추가할 경우

```
FileOutputStream fis = new FileOutputStream("C:/Temp/data.txt", true);  
FileOutputStream fis = new FileOutputStream(file, true);
```

- OutputStream 하위 클래스 - 사용 방법이 OutputStream과 동일

4절. 파일 입출력

❖ FileReader

- 텍스트 파일로부터 데이터를 읽어 들일 때 사용
 - 문자 단위로 읽음
 - 텍스트가 아닌 그림, 오디오, 비디오 등의 파일은 읽을 수 없음

■ 객체 생성 방법

//방법 1

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

//방법 2

```
File file = new File("C:/Temp/file.txt");
```

```
FileReader fr = new FileReader(file);
```

- FileReader 객체가 생성될 때 파일과 직접 연결
 - 만약 파일이 존재하지 않으면 FileNotFoundException 발생
 - try-catch문으로 예외 처리
-
- Reader 하위 클래스 - 사용 방법 Reader와 동일

4절. 파일 입출력

❖ 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 됨.
 - 파일 존재여부 따라 분기
 - 기존 파일 내용 끝에 데이터를 추가할 경우

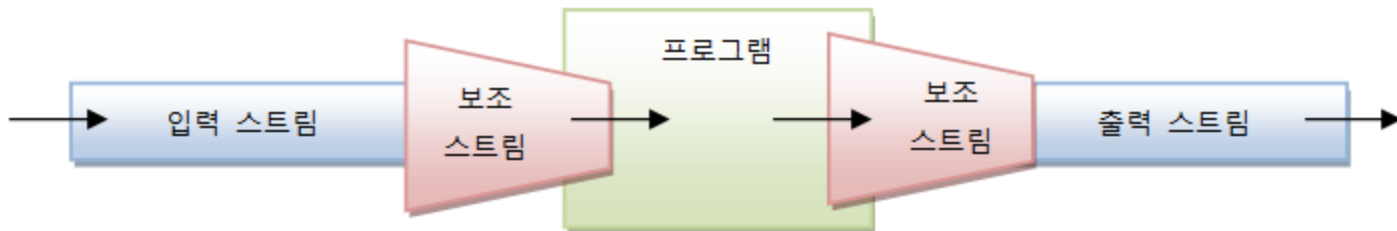
```
FileWriter fw = new FileWriter("C:/Temp/file.txt", true);  
FileWriter fw = new FileWriter(file, true);
```

- Writer 하위 클래스 - 사용 방법이 Writer와 동일

5절. 보조 스트림

❖ 보조 스트림

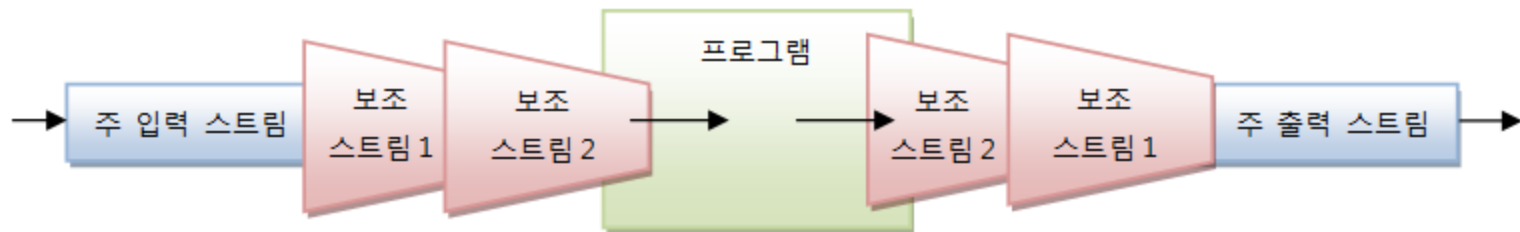
- 다른 스트림과 연결 되어 여러 가지 편리한 기능을 제공해주는 스트림
 - 문자 변환, 입출력 성능 향상, 기본 데이터 타입 입출력, 객체 입출력 등의 기능을 제공



■ 보조 스트림 생성

```
보조스트림 변수 = new 보조스트림(연결스트림)
```

- 보조 스트림 체인 – 다른 보조 스트림과 연결되어 역할 수행

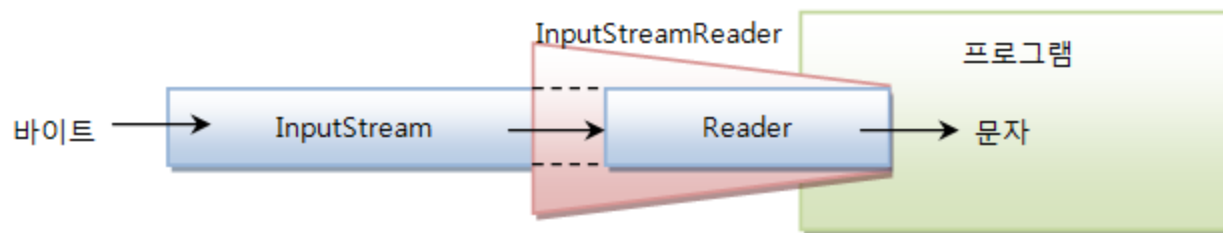


5절. 보조 스트림

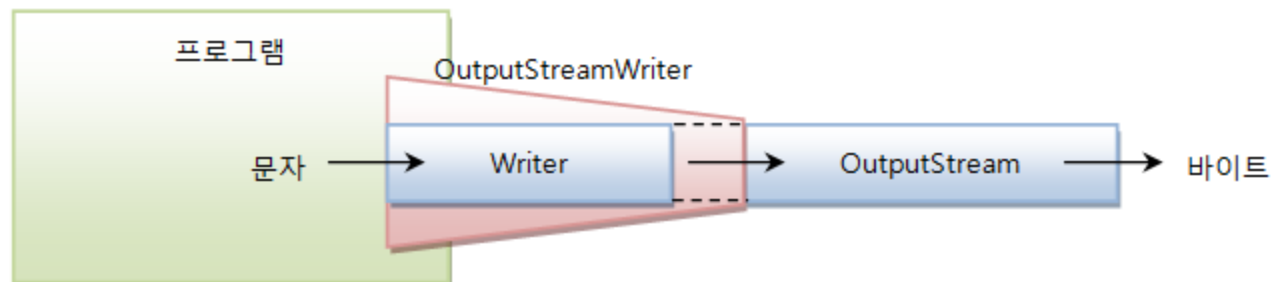
❖ 문자 변환 보조 스트림

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

■ InputStreamReader



■ OutputStreamWriter



5절. 보조 스트림

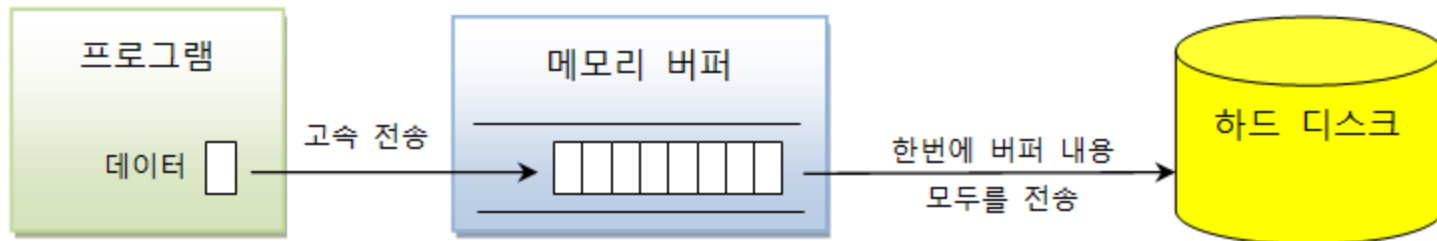
❖ 성능 향상 보조 스트림

■ 입출력 성능에 영향을 미치는 입출력 소스

- 하드 디스크
- 느린 네트워크

■ 버퍼를 이용한 해결 (p.1032~1037)

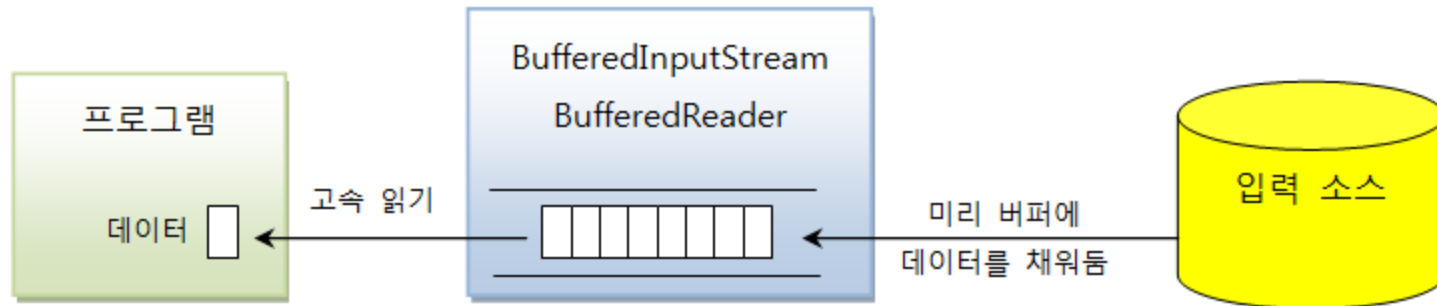
- 입출력 소스와 직접 작업하지 않고 버퍼(buffer)와 작업 - 실행 성능 향상



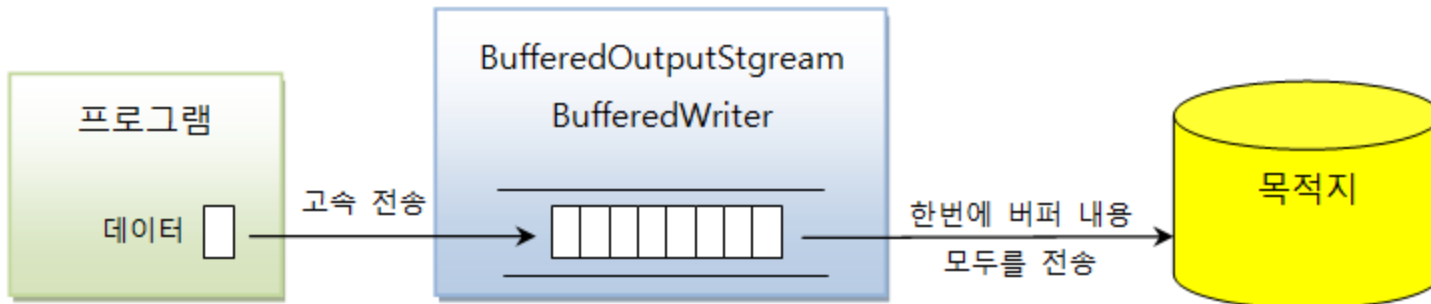
- 프로그램은 쓰기 속도 향상
- 버퍼 차게 되면 데이터를 한꺼번에 하드 디스크로 보내 출력 횟수를 줄여줌

5절. 보조 스트림

■ BufferedInputStream, BufferedReader



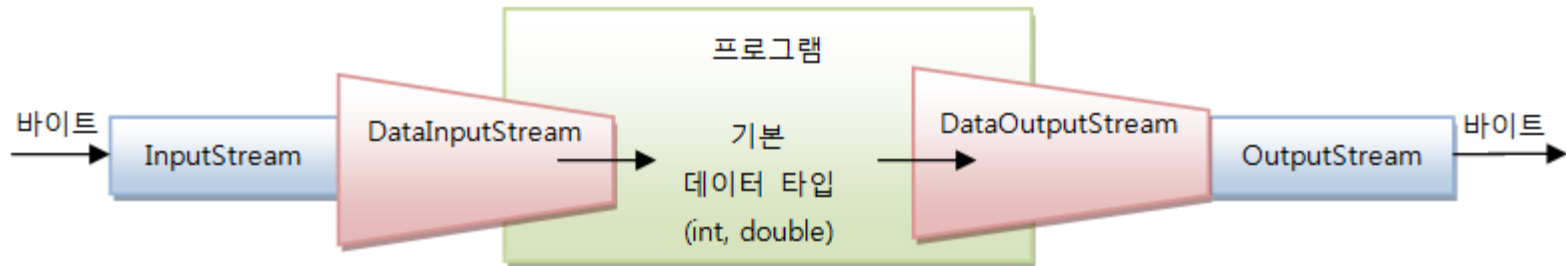
■ BufferedOutputStream과 BufferedWriter



5절. 보조 스트림

❖ 기본 타입 입출력 보조 스트림

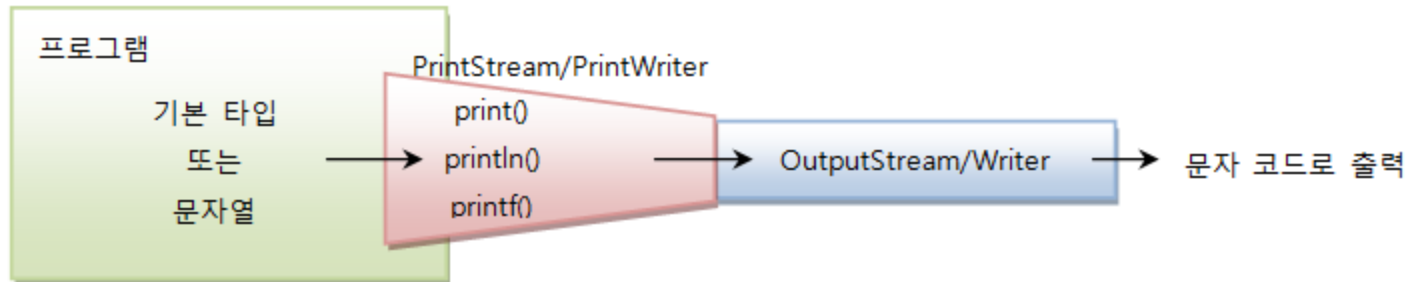
- 입출력 순서를 맞추어 사용



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)

5절. 보조 스트림

❖ 프린터 보조 스트림



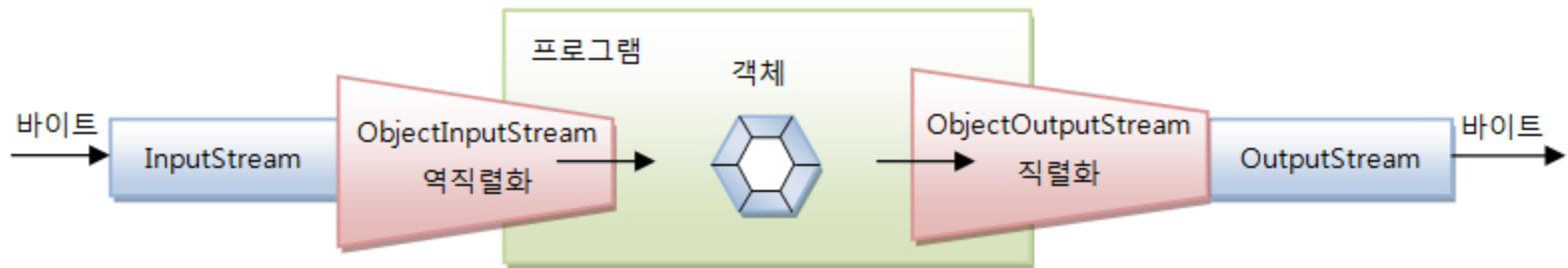
PrintStream / PrintWriter			
void	print(boolean b)	void	println(boolean b)
void	print(char c)	void	println(char c)
void	print(double d)	void	println(double d)
void	print(float f)	void	println(float f)
void	print(int i)	void	println(int i)
void	print(long l)	void	println(long l)
void	print(Object obj)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

- **println()**은 데이터 끝에 개행문자 추가, **printf**는 **format string** 출력

5절. 보조 스트림

❖ 객체 입출력 보조 스트림

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



■ 직렬화가 가능한 클래스(Serializable)

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

5절. 보조 스트림

- serialVersionUID 필드 (p.1047~1049)
 - 직렬화된 객체를 역직렬화 할 때는 직렬화 했을 때와 같은 클래스 사용
 - 클래스의 이름이 같더라도 클래스의 내용이 변경된 경우 역직렬화 실패
- 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 선언

```
static final long serialVersionUID = 정수값;
```

5절. 보조 스트림

- **writeObject()와 readObject() 메소드**
 - **writeObject(ObjectOutputStream out)**
 - 직렬화 직전 자동 호출
 - 추가 직렬화할 내용 작성 가능
 - **readObject(ObjectInputStream in)**
 - 역직렬화 직전 자동 호출
 - 추가 역직렬화 내용 작성 가능
 - **추가 직렬화 및 역직렬화 필요한 경우**
 - 부모 클래스가 Serializable 구현하지 않고, 자식 클래스가 Serializable 구현한 경우
 - 부모 필드는 직렬화에서 제외
 - writeObject() 에서 부모 필드 직렬화 필요
 - readObject()에서 부모 필드 역직렬화 필요
 - 부모 클래스가 Serializable 구현하도록 하는 게 제일 쉬움

6절. 네트워크 기초

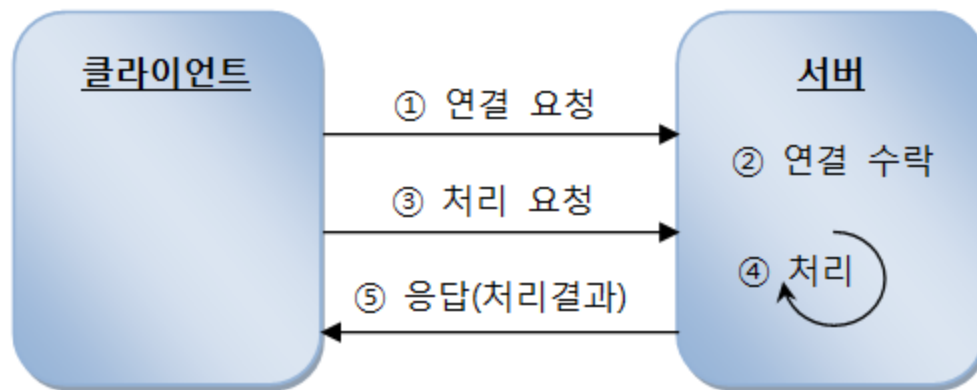
❖ 네트워크

- 여러 대의 컴퓨터를 통신 회선으로 연결한 것
 - 홈 네트워크: 컴퓨터가 방마다 있고, 이들 컴퓨터를 유·무선 등의 통신 회선으로 연결
 - 지역 네트워크: 회사, 건물, 특정 영역에 존재하는 컴퓨터를 통신 회선으로 연결한 것
 - 인터넷: 지역 네트워크를 통신 회선으로 연결한 것

6절. 네트워크 기초

❖ 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)



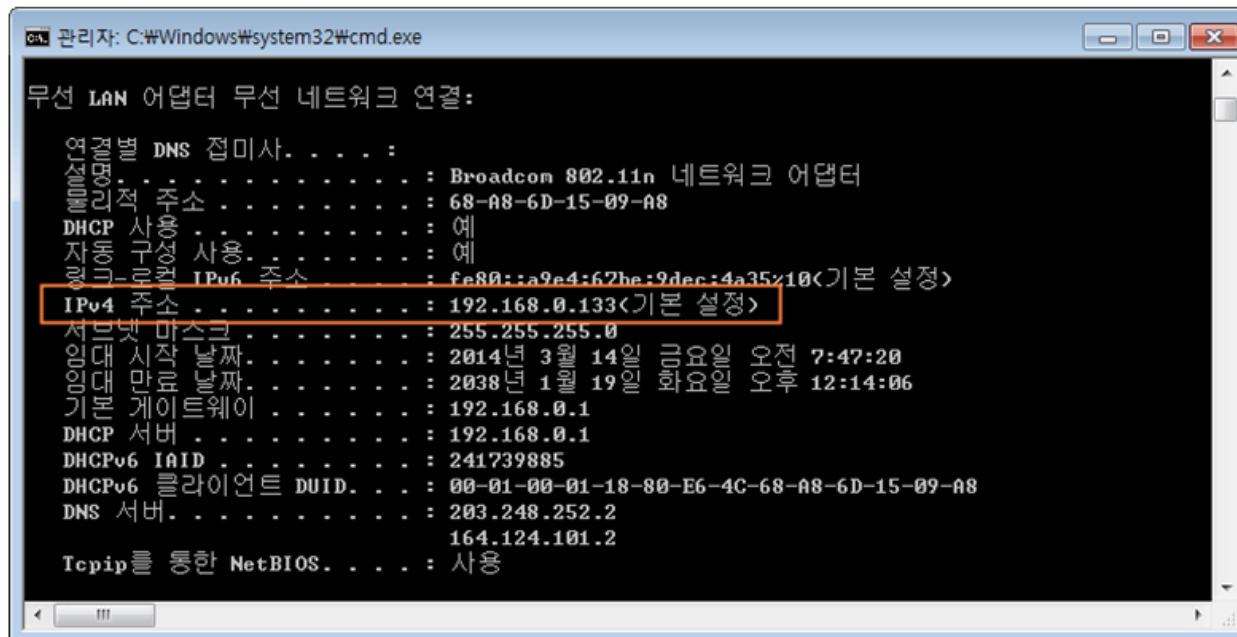
6절. 네트워크 기초

❖ 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)

```
C:\W>ipconfig /all
```



```
관리자: C:\Windows\system32\cmd.exe

무선 LAN 어댑터 무선 네트워크 연결:

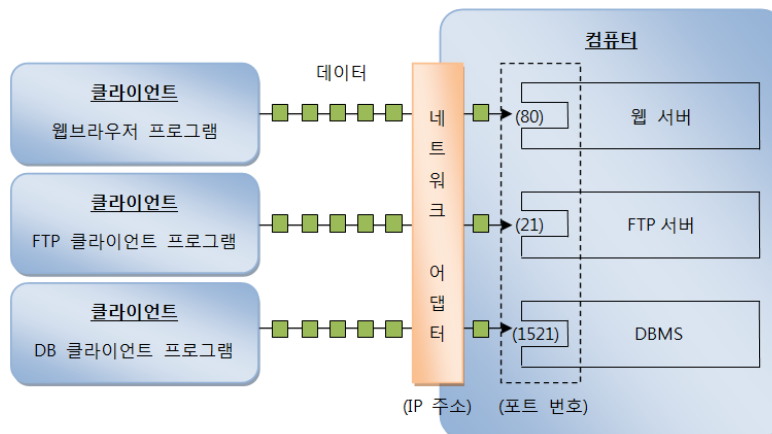
연결별 DNS 접미사. . . . . :
연결별 . . . . . : Broadcom 802.11n 네트워크 어댑터
물리적 주소 . . . . . : 68-A8-6D-15-09-A8
DHCP 사용 . . . . . : 예
자동 구성 사용 . . . . . : 예
링크-로컬 IPv6 주소 . . . . . : fe80::a9e4:67be:9dec:4a35%10<기본 설정>
IPv4 주소 . . . . . : 192.168.0.133<기본 설정>
서브넷 마스크 . . . . . : 255.255.255.0
임대 시작 날짜 . . . . . : 2014년 3월 14일 금요일 오전 7:47:20
임대 만료 날짜 . . . . . : 2038년 1월 19일 화요일 오후 12:14:06
기본 게이트웨이 . . . . . : 192.168.0.1
DHCP 서버 . . . . . : 192.168.0.1
DHCPv6 IAD . . . . . : 241739885
DHCPv6 클라이언트 DUID. . . : 00-01-00-01-18-80-E6-4C-68-A8-6D-15-09-A8
DNS 서버 . . . . . : 203.248.252.2
                  164.124.101.2
Tcpip를 통한 NetBIOS. . . . : 사용
```

6절. 네트워크 기초

■ 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

구분명	범위	설명
Well Know Port Numbers	0~1023	국제인터넷주소관리기구(ICANN)가 특정 애플리케이션용으로 미리 예약한 포트
Registered Port Numbers	1024~49151	회사에서 등록해서 사용할 수 있는 포트
Dynamic Or Private Port Numbers	49152~65535	운영체제가 부여하는 동적 포트 또는 개인적인 목적으로 사용할 수 있는 포트



6절. 네트워크 기초

❖ InetAddress로 IP 주소 얻기 (p.1055~1056)

■ java.net.InetAddress

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

7절. TCP 네트워킹

❖ TCP(Transmission Control Protocol)

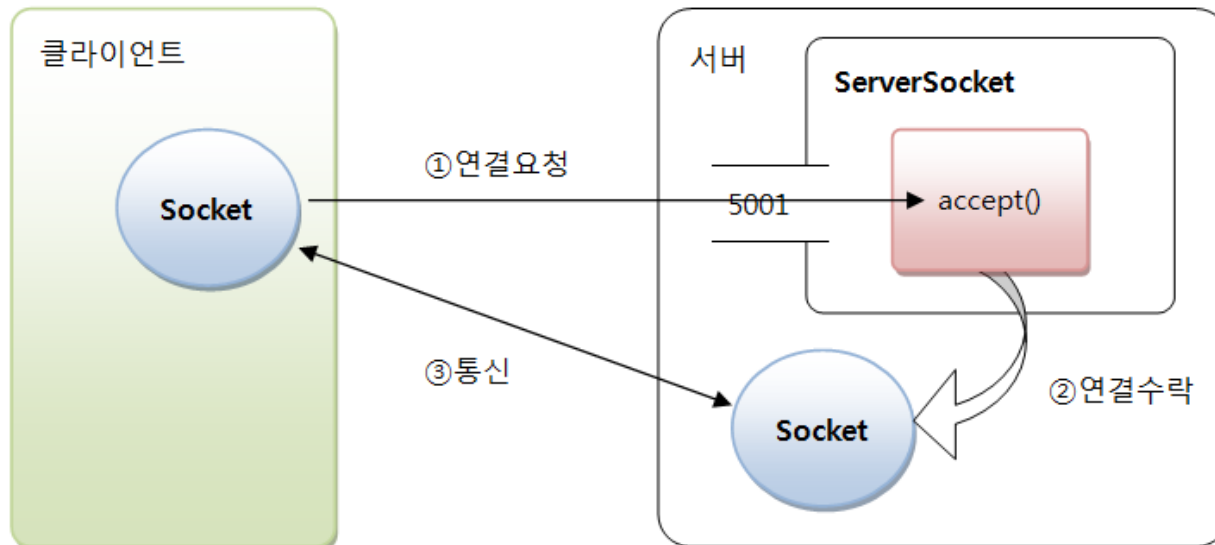
■ 특징

- 연결 지향적 프로토콜 -> 시간 소요
- 통신 선로 고정 -> 전송 속도 느려질 수 있음
- 데이터를 정확하고 안정적으로 전달

■ java.net API

- ServerSocket, Socket

■ ServerSocket과 Socket 용도



7절. TCP 네트워킹

❖ 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

```
InetSocketAddress socketAddress = (InetSocketAddress) socket.getRemoteSocketAddress();
```

리터타입	메소드명(매개변수)	설명
String	<code>getHostName()</code>	클라이언트 IP 리턴
int	<code>getPort()</code>	클라이언트 포트 번호 리턴
String	<code>toString()</code>	"IP:포트번호" 형태의 문자열 리턴

■ ServerSocket 포트 언바인딩

- 더 이상 클라이언트 연결 수락 필요 없는 경우

7절. TCP 네트워킹

❖ 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

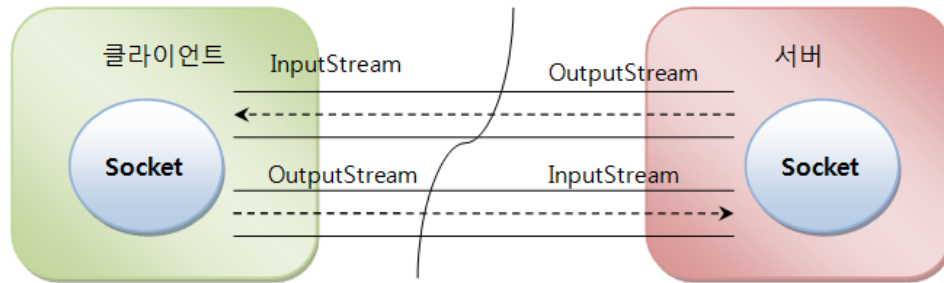
- Exception 처리 필요

■ 1061 페이지 예제를 통해 생성 → 연결 → 끊기 이해

7절. TCP 네트워킹

❖ Socket 데이터 통신

■ Socket 객체로 부터 입출력 스트림 얻기



- 입출력 스트림 구현 예제 (p.1063~1066)
 - 연결 성공 후 클라이언트가 서버에 “Hello Server”
 - 서버가 데이터 받음
 - 서버가 클라이언트에 “Hello Client” 보냄
 - 클라이언트가 데이터 받음
- read()의 블로킹 해제

블로킹이 해제되는 경우	리턴값
상대방이 데이터를 보냄	읽은 바이트 수
상대방이 정상적으로 Socket 의 close()를 호출	-1
상대방이 비정상적으로 종료	IOException 발생

7절. TCP 네트워킹

❖ 스레드 병렬 처리

■ 블로킹(대기 상태)가 되는 메소드

- ServerSocket의 accept()
- Socket 생성자 또는 connect()
- Socket의 read(), write()

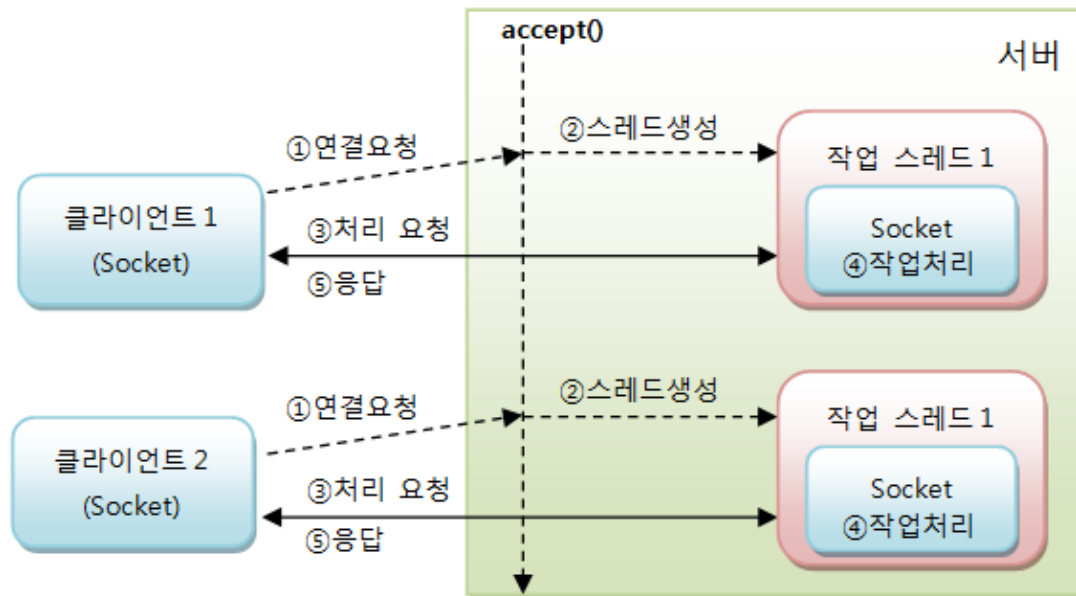
■ 병렬 처리의 필요성

- 스레드가 블로킹되면 다른 작업을 수행하지 못한다.
 - 입출력 할 동안 다른 클라이언트의 연결 요청 수락 불가
 - 입출력 할 동안 다른 클라이언트의 입출력 불가
- UI 생성/변경 스레드에서 블로킹 메소드를 호출하지 않도록
 - UI 생성 및 변경이 안되고 이벤트 처리 불가

7절. TCP 네트워킹

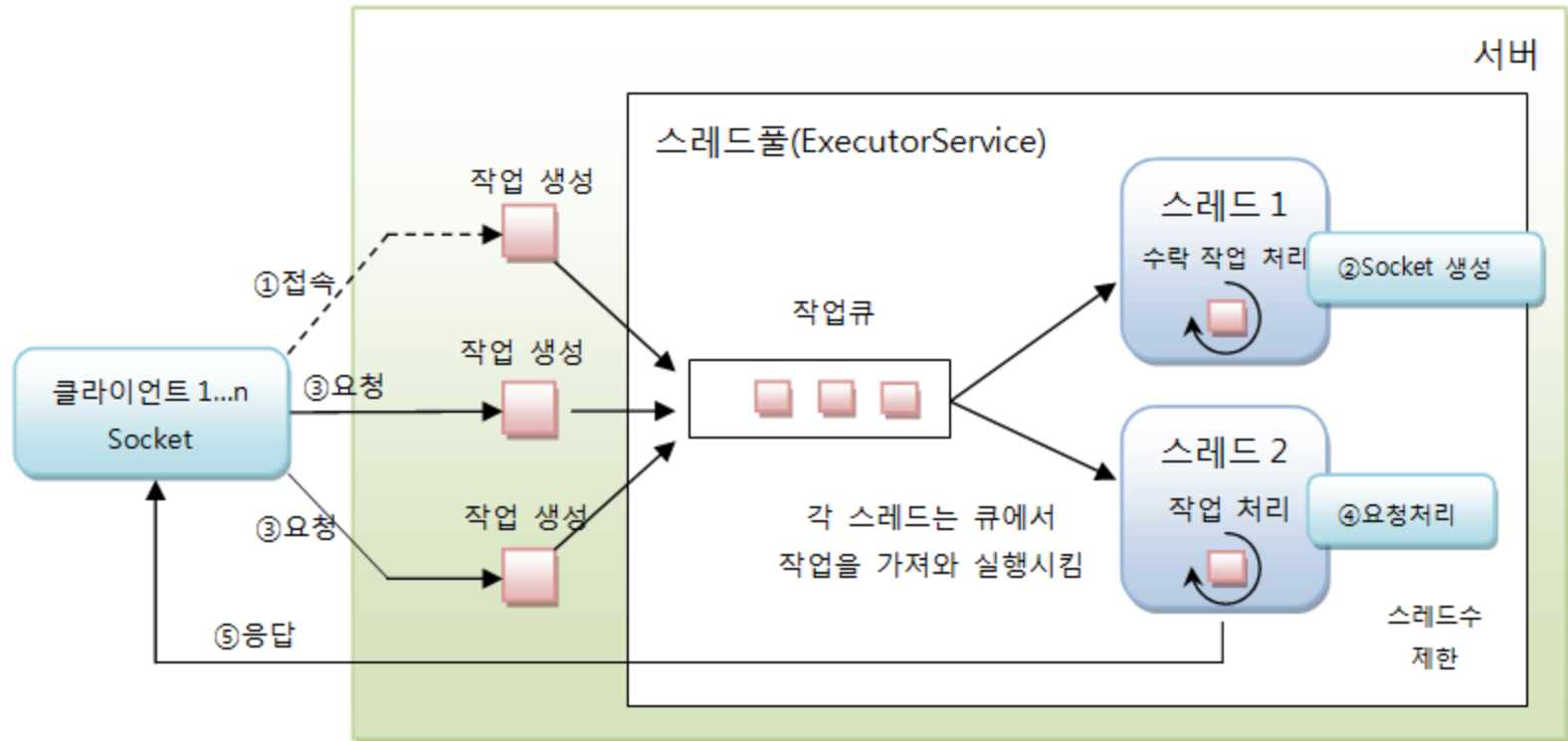
■ 스레드 병렬 처리

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



7절. TCP 네트워킹

■ 스레드풀 사용해 스레드 수 관리



- 스레드풀은 스레드 수 제한해 사용
- 갑작스런 클라이언트의 폭증은 작업 큐의 작업량만 증가
 - 서버 성능은 완만히 저하
 - 대기하는 작업량 많아 개별 클라이언트에서 응답을 늦게 받기도

7절. TCP 네트워킹

❖ 채팅 서버 및 클라이언트 구현 (p.1068~1086)

■ 서버

- `startServer ()`
 - Executor Service , 서버소켓 생성, 포트 바인딩, 연결수락 코드
- `stopServer()`
 - 연결된 모든 소켓, 서버소켓 닫기, Executor Service 종료
- 클라이언트 클래스
 - 다수 클라이언트 관리 → 각자 클라이언트 인스턴스 생성해 관리
- UI 생성 코드 (javaFX 이용한 UI 생성 코드)

■ 클라이언트

- `Startclient()` – 소켓 생성 및 연결요청 코드
- `Stopclient()` – 소켓 통신 닫는 기능도 포함
- `Receive ()` – 서버에서 보낸 데이터 받음
- `Send(String data)` – 사용자가 보낸 메시지 서버로 보냄
- UI 생성 코드

8절. UDP 네트워킹

❖ UDP(User Datagram Protocol)

■ 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

8절. UDP 네트워킹

■ java.net API

- DatagramSocket, DatagramPacket



■ UDP 네트워킹 구현 예제 (p.1087~1091)

- 발신자 구현 코드 – 소켓 통해 데이터 패킷 전송
- 수신자 구현 코드 – 바인딩한 특정 포트로 데이터 받아 저장
- DatagramSocket 닫기