

Java Network and Thread

Java Network and Thread

Thread

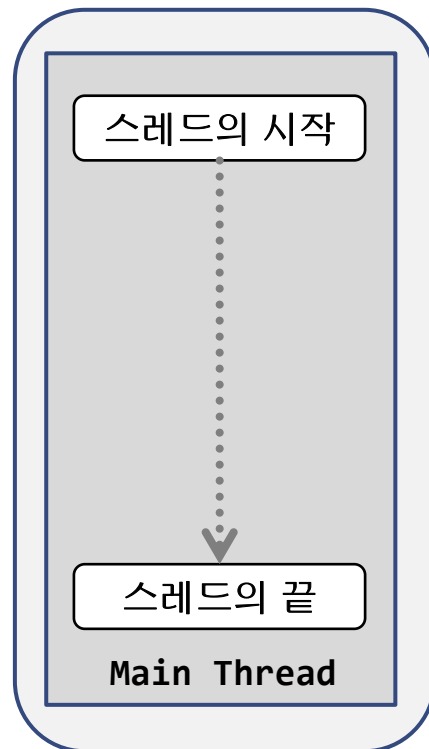
Java Thread

: Thread: 프로그램의 실행 흐름

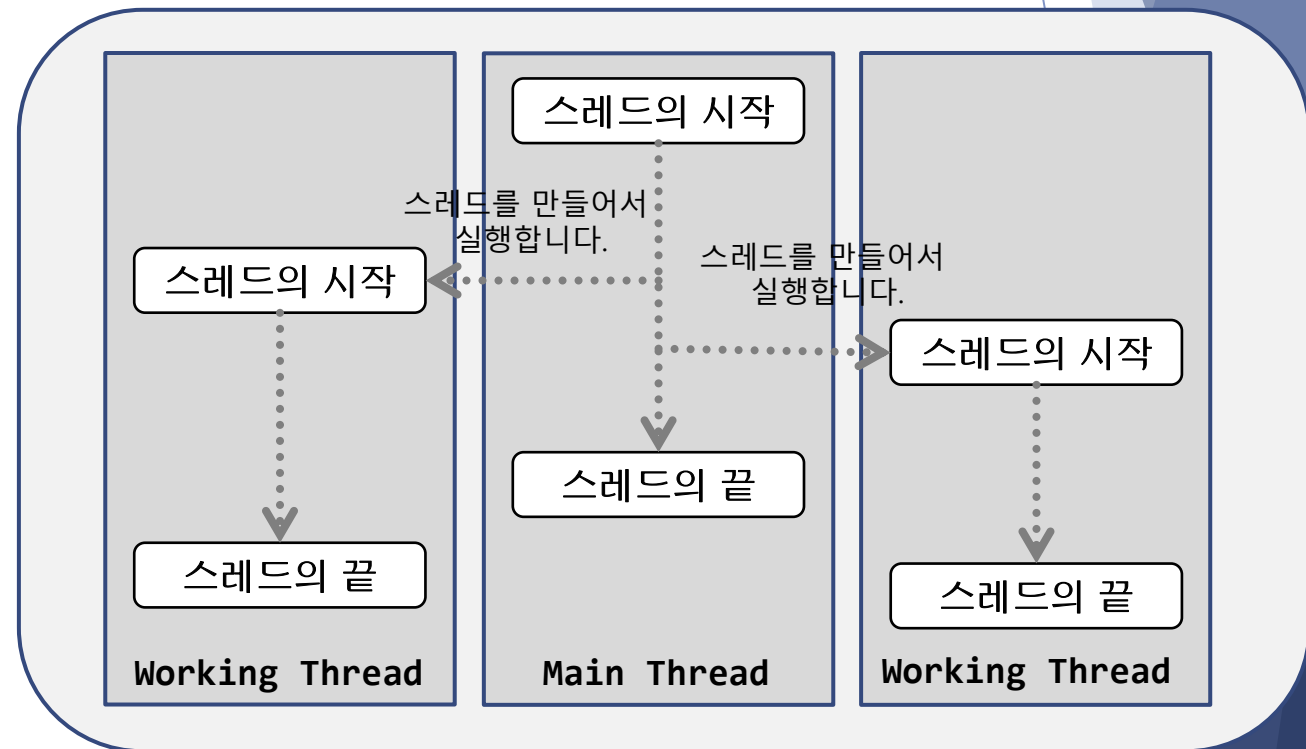
Process: 실행 중인 하나의 프로그램

Thread: 프로세스 내의 하나의 작업 흐름

- 싱글스레드(single thread) 프로그램:
스레드가 하나뿐인 프로그램



- 멀티스레드(multi thread) 프로그램:
스레드가 둘 이상인 프로그램



Java Thread

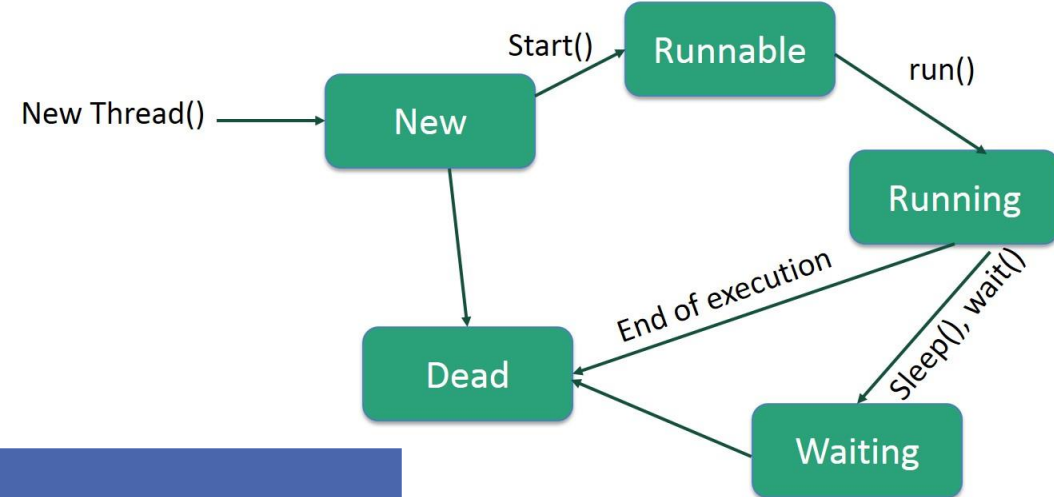
: Thread의 상태와 Multi Thread 프로그램의 작성 방법

▶ Thread의 상태

상태	열거 상수	설명
객체 생성	NEW	쓰레드 객체가 생성. 아직 start() 메서드가 호출되지 않은 상태
실행 대기	RUNNABLE	실행 상태로 언제든지 갈 수 있는 상태
일시 정지	WAITING	다른 쓰레드가 통지할 때까지 기다리는 상태
	TIMED_WAITING	주어진 시간 동안 기다리는 상태
	BLOCKED	사용하고자 하는 객체의 락이 풀릴 때까지 기다리는 상태
종료	TERMINATED	실행을 마친 상태

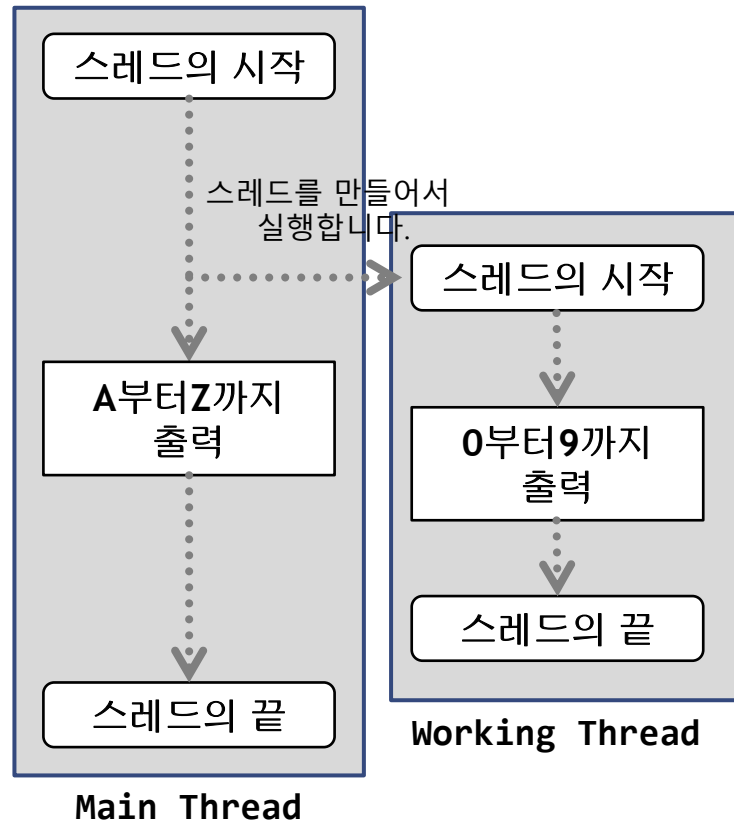
▶ Multi Thread 프로그램 작성 방법

- ▶ java.lang.Thread **클래스**를 이용하는 방법
- ▶ java.lang.Runnable **인터페이스**를 이용하는 방법



Java Thread

: java.lang.Thread 클래스를 이용하는 방법



main 메소드를 포함하는 클래스

```
1 public class Multithread {
2     public static void main(String args[]) {
3         Thread thread = new DigitThread(); // 스레드를 생성
4         thread.start(); // 스레드를 시작
5         for (char ch = 'A'; ch <= 'Z'; ch++) {
6             System.out.print(ch);
7         }
8     }
9 }
```

숫자를 출력하는 스레드 클래스

```
1 public class DigitThread extends Thread {
2     public void run() {
3         for (int cnt = 0; cnt < 10; cnt++) {
4             System.out.print(cnt);
5         }
6     }
7 }
```

연습문제

: java.lang.Thread 클래스 연습

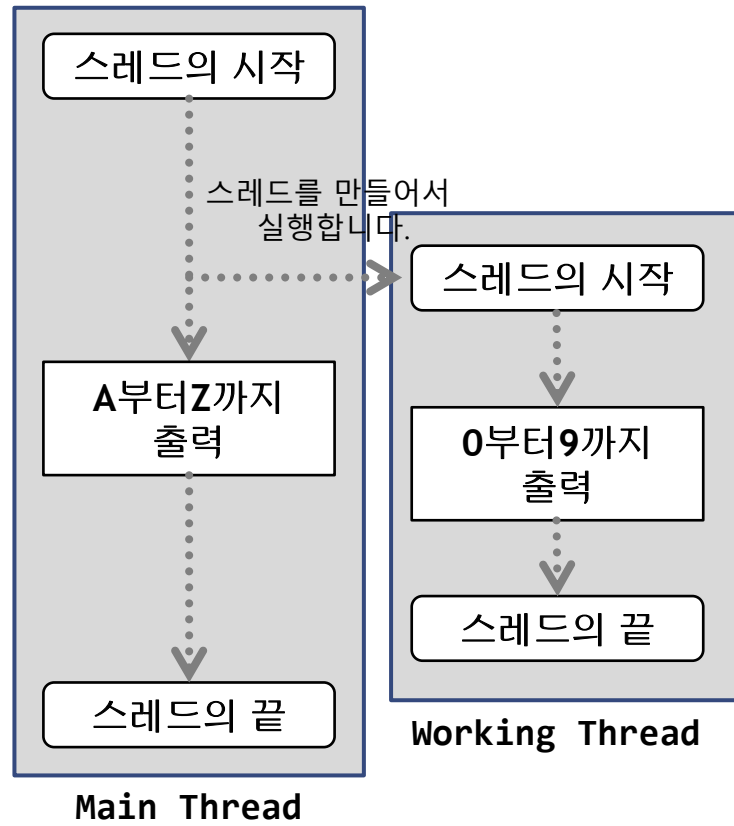
- ▶ 다음 코드를 참고하여 멀티스레드 프로그램을 작성하고 테스트 한다.

```
1 public class Multithread {  
2     public static void main(String args[]) {  
3  
4         Thread thread1 = new DigitThread();  
5         Thread thread2 = new DigitThread();  
6         Thread thread3 = new AlphabetThread();  
7  
8         thread1.start();  
9         thread2.start();  
10        thread3.start();  
11    }
```

3개의 스레드를
생성해서 시작

Java Thread

: java.lang.Runnable 인터페이스를 이용하는 방법



main 메소드를 포함하는 클래스

```
1 public class Multithread {
2     public static void main(String args[]) {
3         Thread thread = new Thread(new DigitRunnableImpl())
4                                     // 스레드를 생성
5         thread.start();           // 스레드를 시작
6         for (char ch = 'A'; ch <= 'Z'; ch++) {
7             System.out.print(ch);
8         }
9     }
10 }
```

숫자를 출력하는 스레드 클래스

```
1 public class DigitRunnableImpl implements Runnable {
2     public void run() {
3         for (int cnt = 0; cnt < 10; cnt++) {
4             System.out.print(cnt);
5         }
6     }
7 }
```

연습문제

: java.lang Runnable 인터페이스 연습

- ▶ 다음 코드를 참고하여 멀티스레드 프로그램을 작성하고 테스트한다 (Runnable 이용)

```
1 public class Multithread {  
2     public static void main(String args[]) {  
3  
4         Thread thread1 = new Thread(new DigitRunnableImpl());  
5         Thread thread2 = new Thread(new DigitRunnableImpl());  
6         Thread thread3 = new Thread(new AlphabetRunnableImpl());  
7  
8         thread1.start();  
9         thread2.start();  
10        thread3.start();  
11    }
```

3개의 스레드를
생성해서 시작

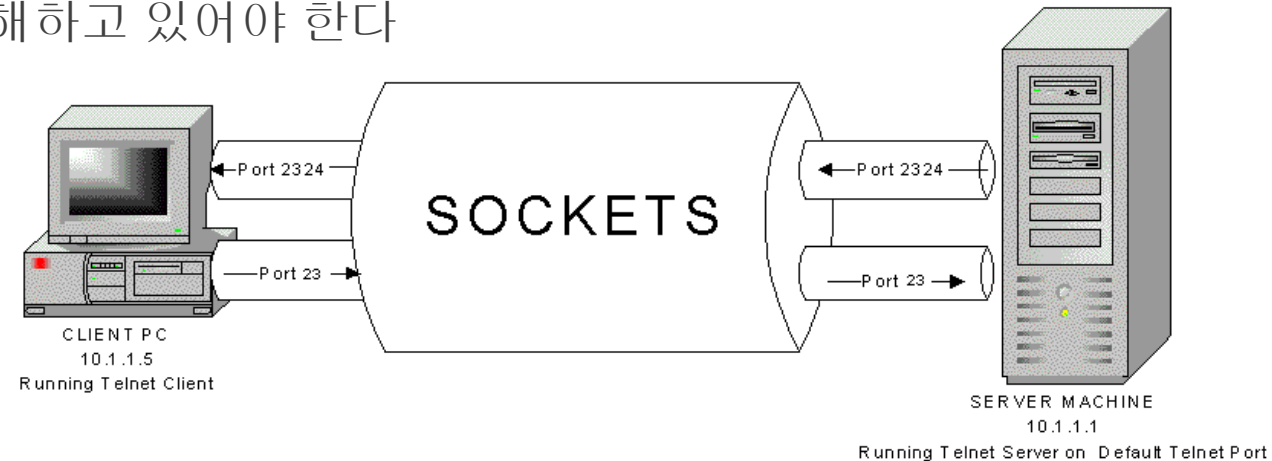
Java Network and Thread

TCP Socket Programming

TCP Socket Programming

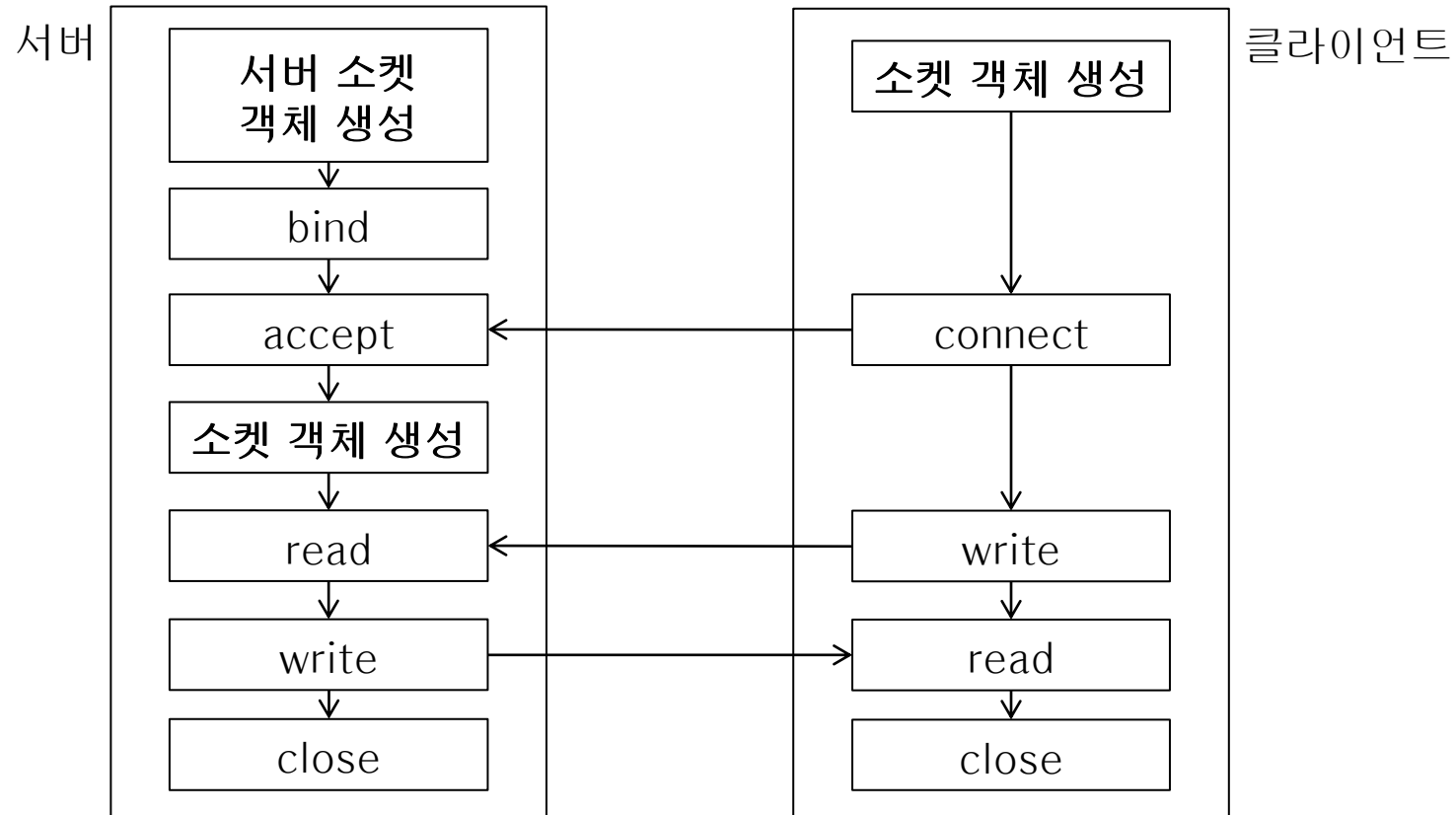
: TCP 소켓의 특징

- ▶ 스트림(stream) 통신 프로토콜
- ▶ 양쪽의 소켓이 연결된 상태에서 통신이 가능하다(연결지향 프로토콜)
- ▶ 신뢰성 있는 데이터 통신
- ▶ 한번 연결되면 끊어질 때까지 송신한 데이터는 차례대로 목적지의 소켓에 전달
- ▶ 자바는 **java.net** 패키지에 관련 클래스를 제공
 - ▶ TCP 소켓 프로그래밍을 쉽게 할 수 있도록 돕는다
- ▶ 라이브러리의 사용법과 작동 순서를 명확하게 이해하고 있어야 한다
- ▶ **ServerSocket**과 **Socket** 클래스를 사용한다



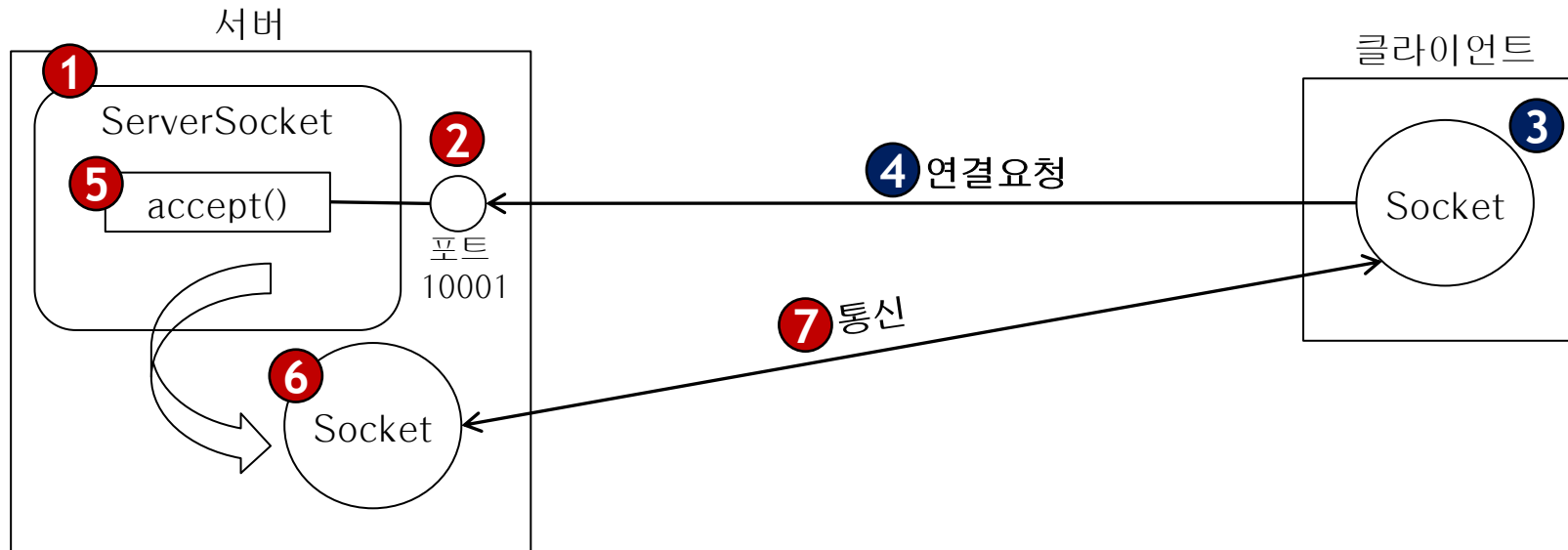
TCP Socket Programming

: TCP 소켓 프로그래밍 절차



TCP Socket Programming

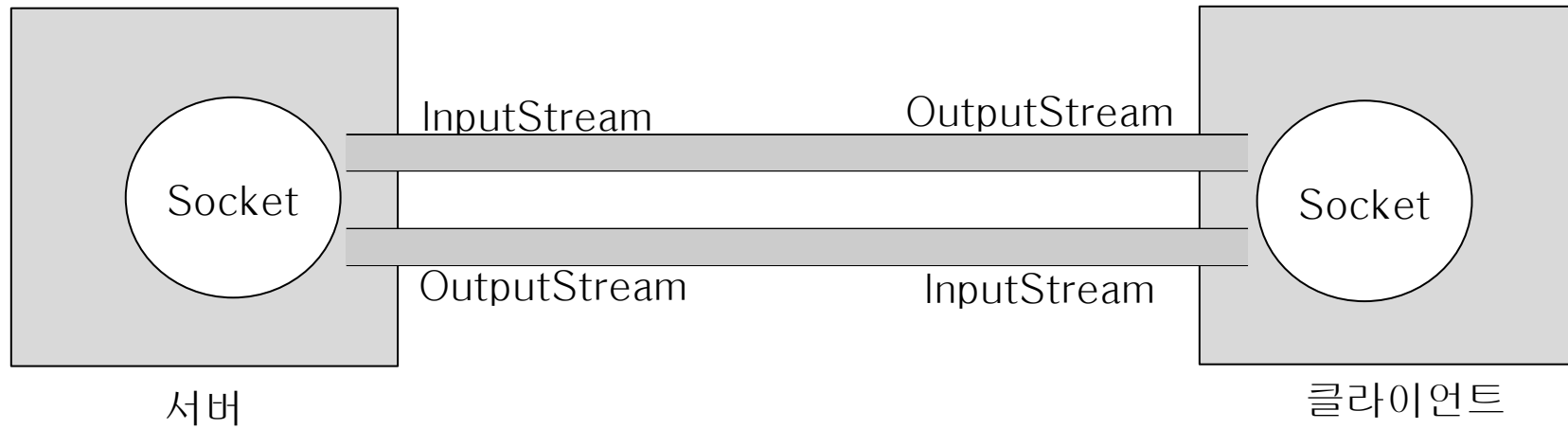
: ServerSocket과 Socket



- ▶ **ServerSocket** : 클라이언트의 연결 요청을 기다리면서 연결 요청에 대한 수락을 담당
- ▶ **Socket** : 클라이언트와 통신을 직접 담당한다

TCP Socket Programming

: Socket 객체의 데이터 통신



- ▶ 양쪽의 **Socket** 객체로부터 **InputStream**, **OutputStream**을 얻어와 데이터 통신에 사용