날짜 API

날짜 관련 API

- 날짜관련 API 클래스
  - ✓ **Date**
  - ✓ **Calendar**
  - ✓ **SimpleDateFormat**

Date

- 1.0 버전 부터 지원되는 클래스
- 1.1 버전 부터는 Calendar 클래스 사용을 권장

➢ **생성자**

- Date()
- Date(long *msec*)

Date

메소드

이름	설명
long getTime()	1970년 이후로 현재까지의 시간을 밀리초로 반환
int getYear()	1900년 이후부터의 년수를 반환
int getMonth()	해당되는 월을 반환. 0:1월 - 11:12월
int getDate()	1-31 사이의 날짜를 반환
int getDay()	요일을 해당되는 숫자로 반환. 0:일요일 - 6:토요일
int getHours()	0-23 까지의 시간을 반환
int getMinutes()	0-59 사이의 분을 반환
int getSeconds()	0-59 사이의 초를 반환

Calendar

- 추상클래스
- 객체를 얻기위해 Calendar.getInstance( ) 를 활용

➢ 메 소 드

- 객체 얻기 : static Calendar getInstance()
- 정보 추출 : int get(int calendarField)

필드	의미	필드	의미
YEAR	년	HOUR HOUR_OF_DAY	시간
MONTH	월	MINUTE	분
DATE DAY_OF_MONTH	일	SECOND	초
DAY_OF_WEEK	요일		

Calendar

- 날짜 설정 :

  void set(int year, int month, int date)

  void set(int year, int month, int date, int hour, int minute)

  void set(int year, int month, int date, int hour, int minute, int second)

- Date 객체 얻기 :

  Date getTime() 현재의 객체와 같은 날짜를 가진 Date 객체를 반환

- Date 객체 시간정보를 Calendar 로 설정하기 :

  void setTime(Date d) Date 객체 d의 정보를 이용하여 현재의 객체를 설정

- 날짜 정보에서 해당 항목의 최대값 얻기 :

  int getActualMaximum (int calendarField)

SimpleDateFormat

- 날짜 객체로 부터 원하는 형태의 문자열로 변환
- 특정한 포멧 문자열을 사용하여 날짜 정보를 추출

➢ 주요 메소드

- SimpleDateFormat(String pattern)

  pattern 에 지정된 형태로 날짜를 문자열로 변환

주요패턴 문자					
y	년	M	월	d	일
H h	시간(0 – 23) 시간(0 – 11)	m	분	s	초
E	요일				

- String format (Date d)

  Date 객체를 매개변수로 받아서 지정된 패턴 형식으로 문자열 반환

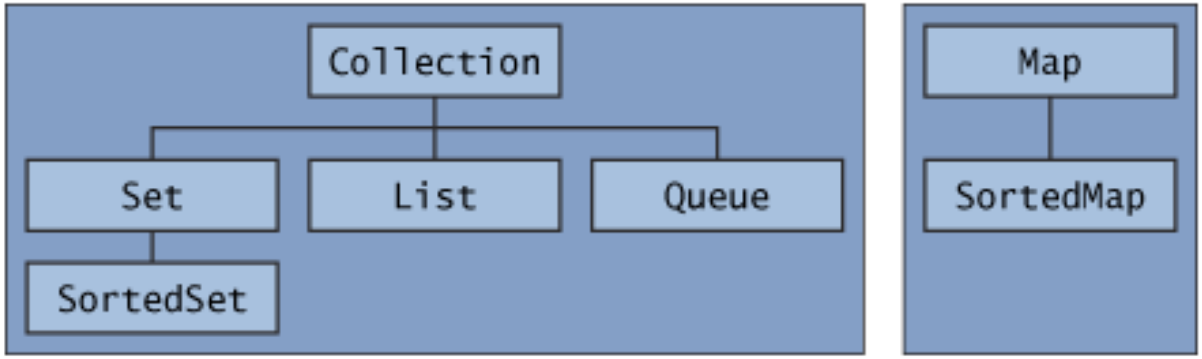
Collection API

Collection 과 자료구조

- 객체들을 한곳에 모아놓고 편리하게 사용할 수 있는 환경을 제공
- 정적 자료구조(Static structure)
  - 고정된 크기의 자료구조
  - 배열이 대표적인 정적 자료구조
  - 선언 시 크기를 명시하면 바꿀 수 없음
- 동적 자료구조(Dynamic structure)
  - 요소의 개수에 따라 자료구조의 크기가 동적으로 증가하거나 감소
  - 벡터, 리스트, 스택, 큐 등

Collection 과 자료구조

- 자료구조들의 종류는 결국은 어떤 구조에서 얼마나 빨리 원하는 데이터를 찾는가에 따라 결정된다.
  - 순서를 유지할 것인가?
  - 중복을 허용할 것인가?
  - 다른 자료구조들에 비해서 어떤 단점과 장점을 가지고 있는가?



Generic

- Collections Framework이 기존에는 모든 객체자료형들을 처리하기 위해서 java.lang.Object 타입을 사용
- JDK1.5이후에는 컴파일 시점에 자료구조에서 사용되는 Type을 체크하는 Generic 문법을 사용하는 방식으로 변화

```
형식 : 클래스<타입>
예>  List<String> list = new ArrayList<String>( );
```

List

- 특징: 순서가 있고, 중복을 허용 (배열과 유사)
- 장점: 가변적인 배열
- 단점: 원하는 데이터가 뒤쪽에 위치하는 경우 속도의 문제
- 구현 클래스
  - ArrayList
  - LinkedList

ArrayList - 메소드

- 내부적으로 배열을 이용하여 데이터를 관리
- 배열과 다르게 크기가 유동적으로 변함(동적 자료구조)
- 배열을 다루는 것과 유사하게 사용 할 수 있음

ArrayList와 Generic

- ArrayList list = new ArrayList( );
- ArrayList<String> list2 = new ArrayList<String>( );

위의 두 코드의 차이점??

list 는 모든 객체를 받을 수 있음

list2 는 String 만을 받을 수 있음

ArrayList – 메소드

- add(E e) : 데이터 입력

봄	여름			

list.add (“봄”);

list.add (“여름”);

ArrayList – 메소드

- get(int index) : 데이터 추출

봄	여름			
0	1	2	3	4

String val = list.get(0); → 봄이 반환

ArrayList – 메소드

- size( ) : 크기 반환

봄	여름
0	1

int size = list.size( ); → 2가 반환

ArrayList – 메소드

- remove(int i) : 인덱스 위치의 데이터를 삭제


```
list.remove( 0 );
```

ArrayList – 메소드

- remove(Object o) : 동일한 데이터를 삭제

봄	여름			
0	1	2	3	4

list.remove(“봄”);

ArrayList – 메소드

- clear ( ) : 모든 데이터를 삭제

봄	여름			
0	1	2	3	4

```
list.clear( );
```

ArrayList – 메소드

- contains(Object o) : 특정 데이터가 있는지 체크

봄	여름			
0	1	2	3	4

boolean b = list.contains(“봄”);

ArrayList – 메소드

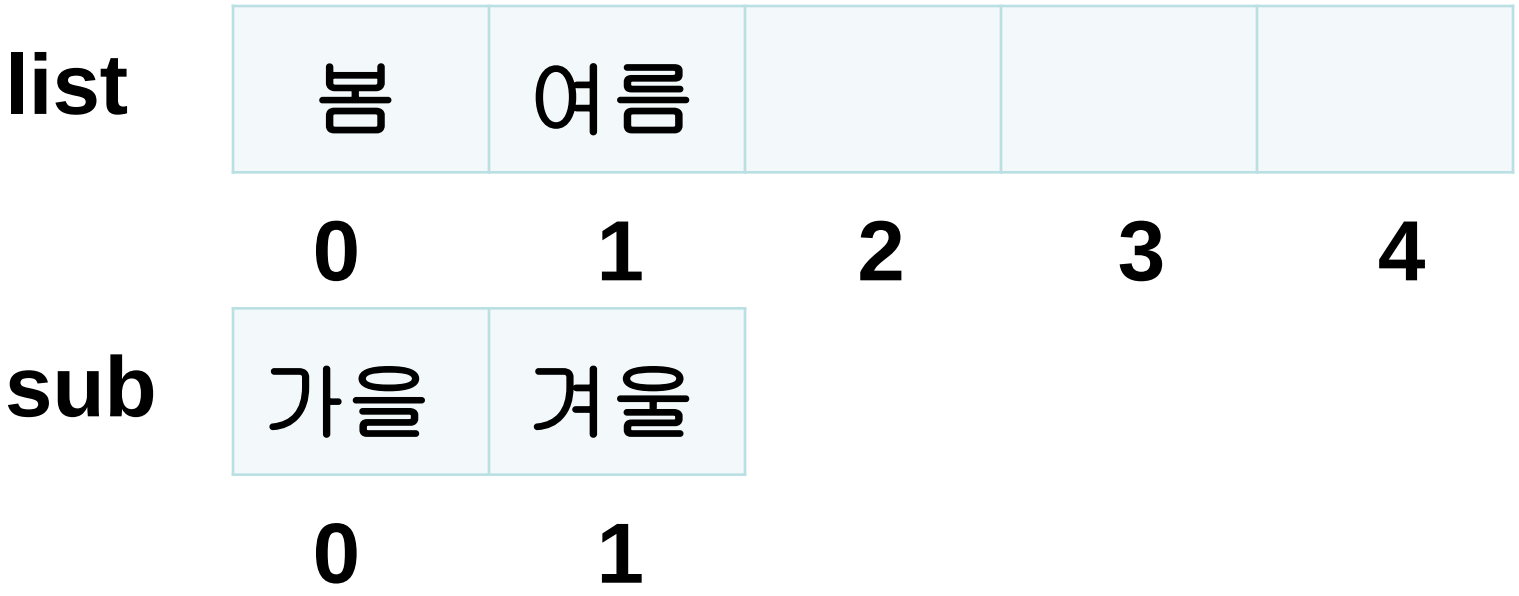
- isEmpty( ) : 데이터가 존재하는지 체크

봄	여름			
0	1	2	3	4

boolean b = list.isEmpty( );

ArrayList – 메소드

- addAll(Collection c) : 기존 등록된 콜렉션 데이터 추가




```
list.addAll(  sub );
```

ArrayList - 메소드

- add(E e) : 데이터 입력
- get(int index) : 데이터 추출
- size() : 입력된 데이터의 크기 반환
- remove(int i) : 특정한 데이터를 삭제
- remove(Object o) : 특정한 데이터를 삭제
- clear() : 모든 데이터 삭제
- contains(Object o) : 특정 객체가 포함되어 있는지 체크
- isEmpty() : 비어있는지 체크(true, false)
- addAll(Collection c) : 기존 등록된 콜렉션 데이터 입력
- iterator() : Iterator 인터페이스 객체 반환

Map

- 특징 : Key(키)와 Value(값)으로 나누어 데이터 관리, 순서는 없으며, 키에 대한 중복은 없음
- 장점 : 빠른 속도
- 구현 클래스
  - HashMap
  - TreeMap

Map – 메소드

- **V put (K key, V value) : 데이터 입력**

  동일한 값이 있을 경우 새로운 값으로 대체하고 기존 값 반환

value	길동	인천			
key	**name**	**addr**			

map.put(“name”, “길동”);

map.put(“addr”, “인천”);

Map – 메소드

- **V get (Object Key) : 데이터 추출**

  Key 에 해당하는 값이 없을 경우 null 반환

value	길동	인천			
key	**name**	**addr**			

String val = map.get(“name”); → “길동”이 반환

Map – 메소드

- **V remove (Object Key) : 데이터 삭제**

  삭제된 값을 리턴, Key 에 해당하는 값이 없을 경우 null 반환

value	길동	인천			
key	**name**	**addr**			

String val = map.remove(“addr”); → “인천”이 반환

Map – 메소드

- **boolean containsKey(Object Key) : 특정 키 확인**

  Key 가 존재할 경우 true 반환

value	길동	인천			
key	**name**	**addr**			

boolean flag = map.containsKey(“addr”); → true 반환

Map – 메소드

- **void putAll(Map<K Key, V value> m) : 콜렉션 추가**

  리스트의 addAll과 같은 역할, 기존 콜렉션에 구성된 데이터를 추가할 경우

value	길동	인천			
key	**name**	**addr**			

value	**16**				
key	**age**				

map.putAll(sub);

Map - 메소드

- V put(K key, V value) : 데이터 입력
- V get(Object key) : 데이터 추출
- V remove(K key) : 입력된 데이터의 크기 반환
- boolean containsKey(Object key) : 특정한 key 포함 여부
- void putAll(Map<K key, V value> m) : 기존 콜렉션 데이터 추가
- Set<Map.Entry<K, V>> entrySet() :

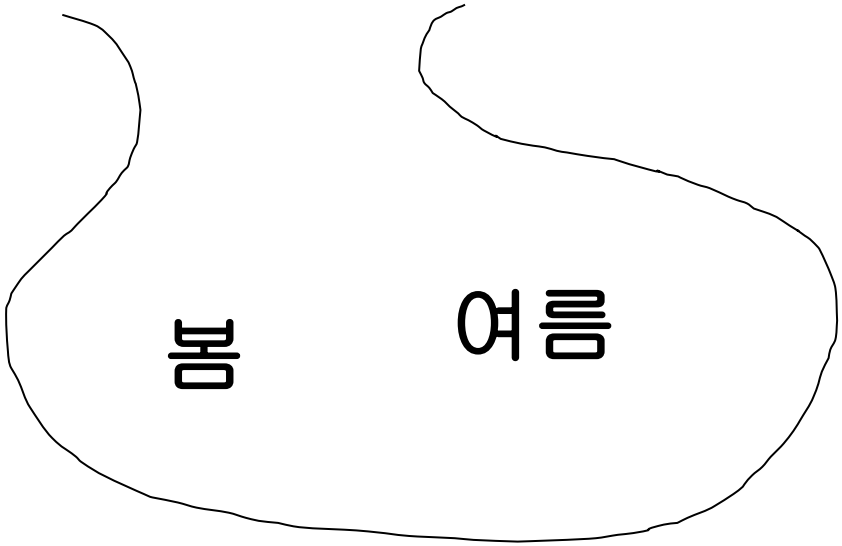
  (key 와 value) 쌍을 표현하는 Map.Entry 집합을 반환

Set

- 특징 : 순서가 없고, 중복을 허용하지 않음
- 장점 : 빠른 속도
- 단점 : 단순 집합의 개념으로 정렬하려면 별도의 처리가 필요하다.
- 구현 클래스
  - HashSet
  - TreeSet

Set – 메소드

- boolean add(E e) : 데이터 입력

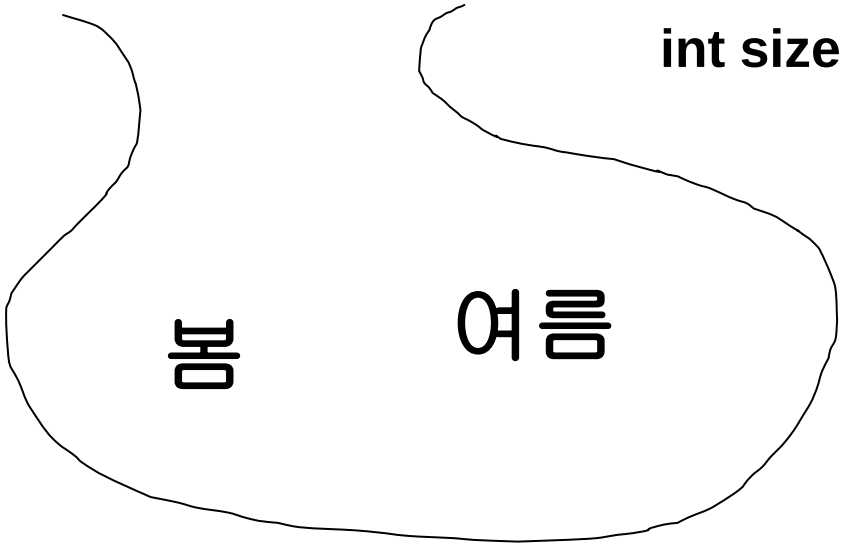


set.add (“봄”);

set.add (“여름”);

Set – 메소드

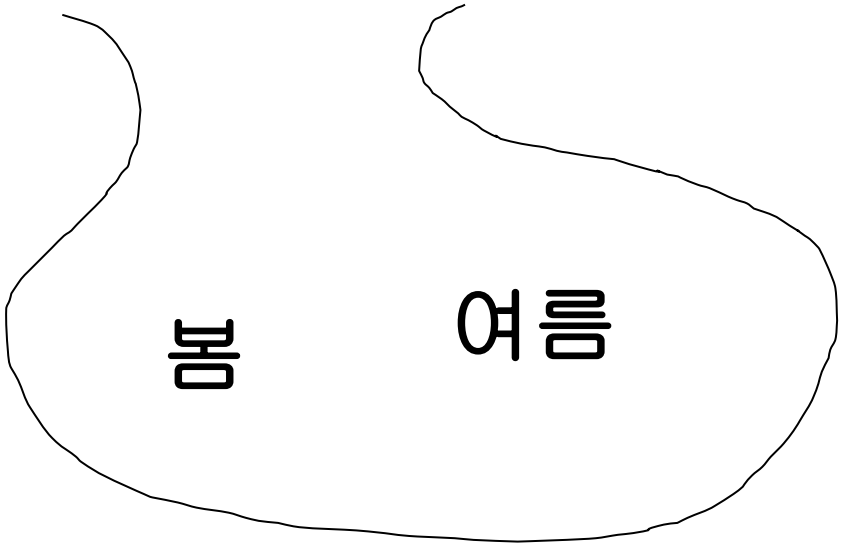
- int size( ) : 크기 반환



int size = set.size( ); → 2가 반환

Set – 메소드

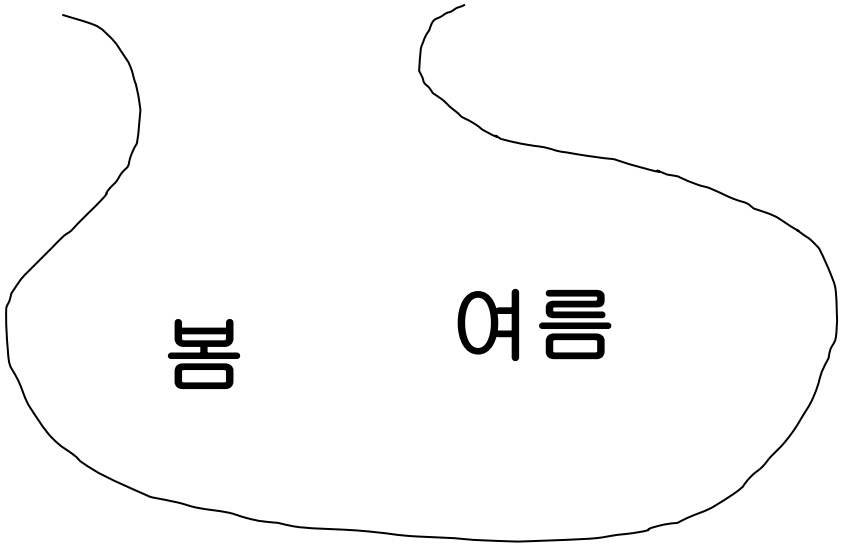
- boolean remove(Object o) : 동일한 데이터를 삭제



set.remove(“봄”);

Set – 메소드

- void clear ( ) : 모든 데이터를 삭제

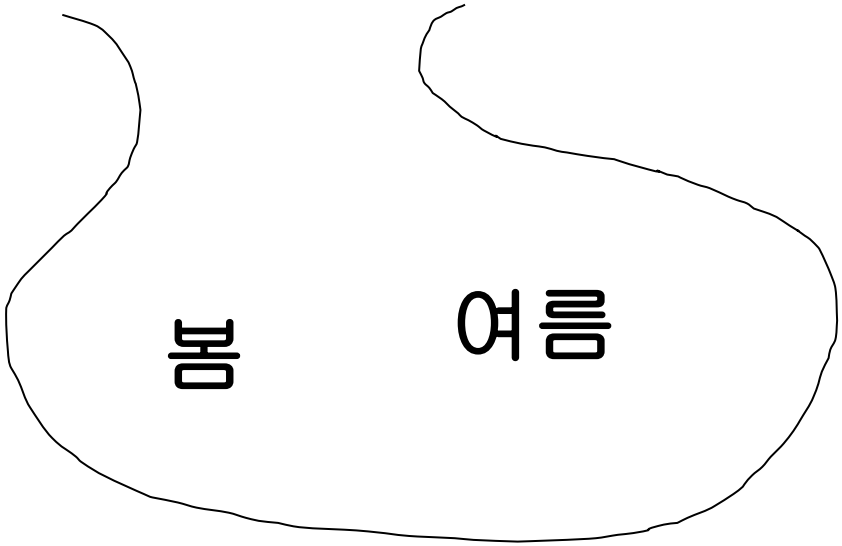



```
set.clear( );
```

Set – 메소드

- contains(Object o) : 특정 데이터가 있는지 체크

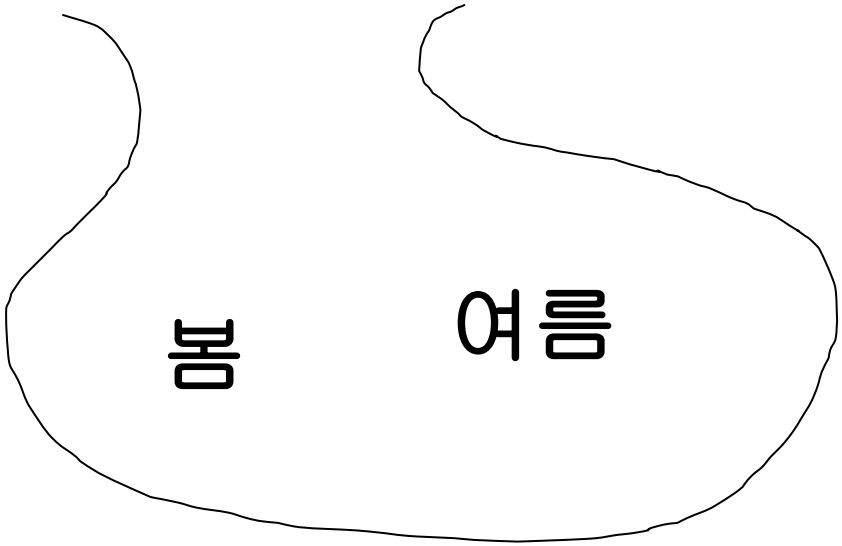
boolean b = set.contains(“봄”);



ArrayList – 메소드

- isEmpty( ) : 데이터가 존재하는지 체크

boolean b = set.isEmpty( );



Set - 메소드

- add(E e) : 데이터 입력
- size() : 입력된 데이터의 크기 반환
- remove(Object o) : 특정한 데이터를 삭제
- clear() : 모든 데이터 삭제
- contains(Object o) : 특정 객체가 포함되어 있는지 체크
- isEmpty() : 비어있는지 체크(true, false)
- iterator() : Iterator 인터페이스 객체 반환
- toArray () : Set의 내용을 Object 형의 배열로 반환

입출력 API

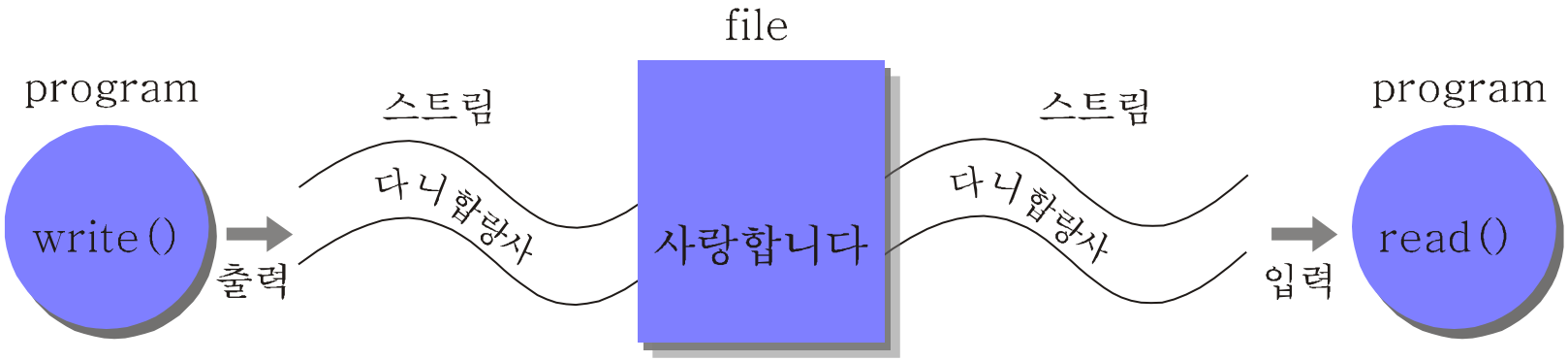
입출력 API

- 입출력이란
- **java.io** 패키지
- 바이트 스트림
- 문자 스트림

입출력 스트림의 개요

java.io Package를 제공함

입,출력을 위해서 스트림을 사용함**(byte, character)**



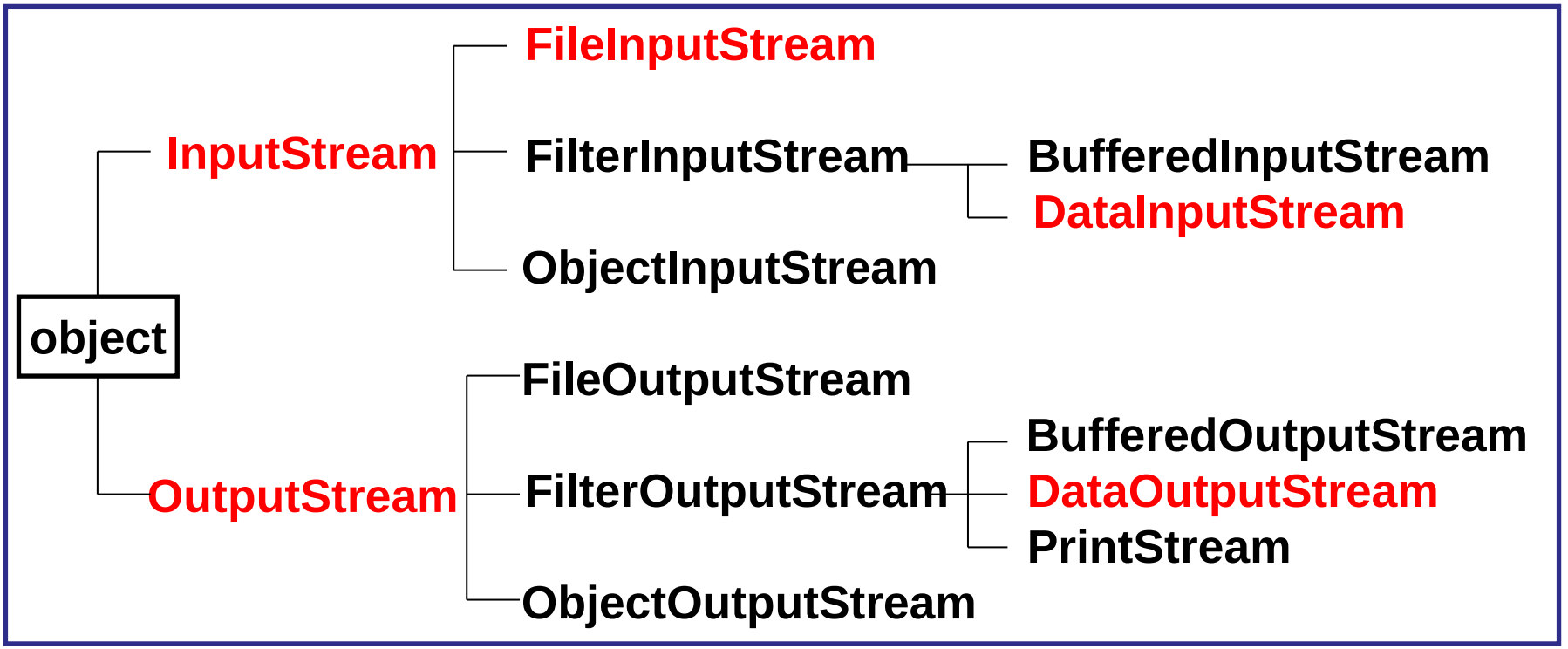
특 징

1. **FIFO**
2. 단방향이다.
3. 지연될 수 있다.

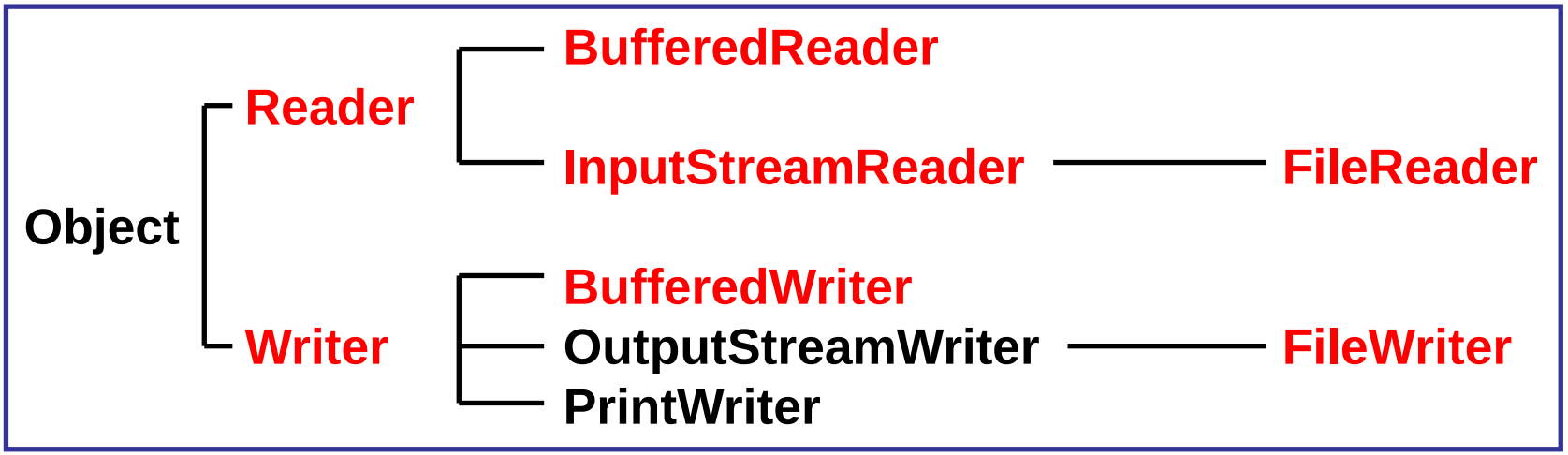
IO 처리단위

	byte	Char
입 력	InputStream	Reader
출 력	OutputStream	Writer

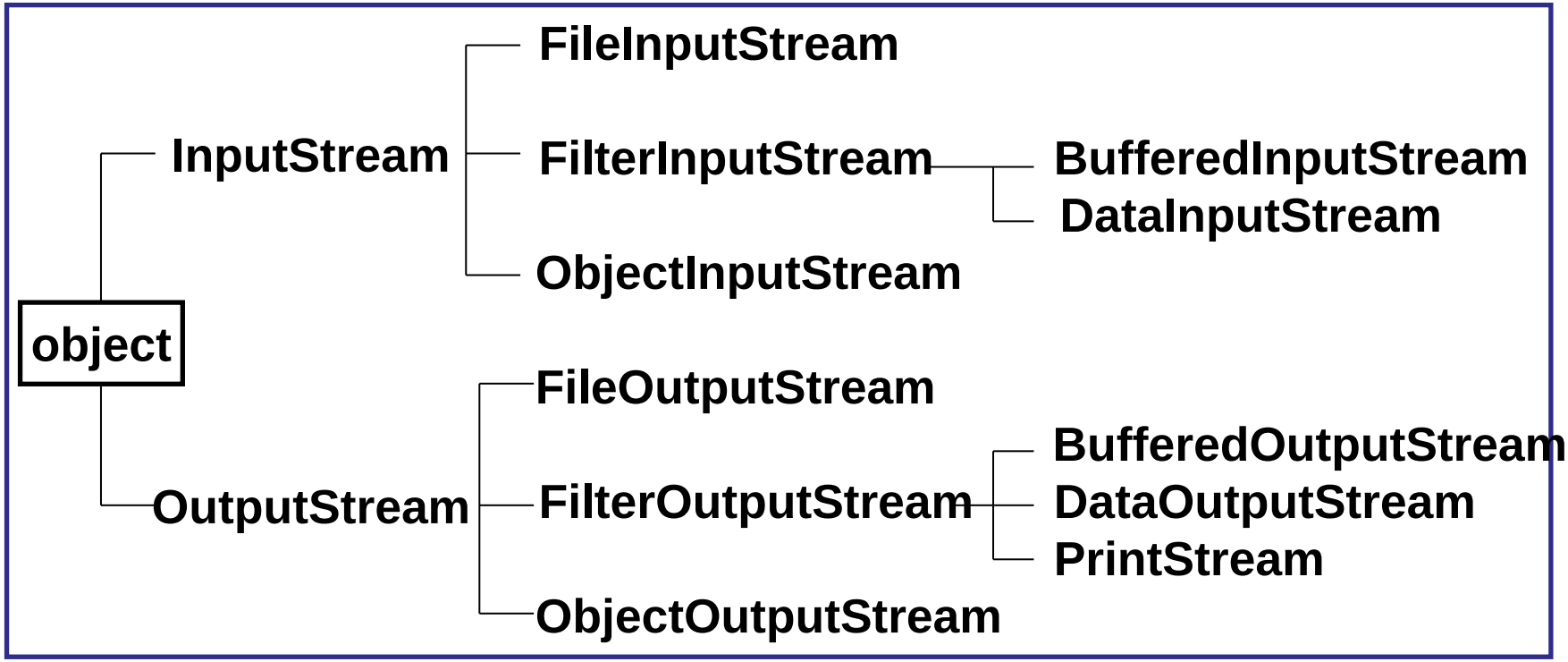
바이트 스트림


object
InputStream
FileInputStream
FilterInputStream
BufferedInputStream
DataInputStream
ObjectInputStream
OutputStream
FileOutputStream
FilterOutputStream
BufferedOutputStream
DataOutputStream
PrintStream
ObjectOutputStream

문자 스트림


Object
Reader
BufferedReader
InputStreamReader
FileReader
Writer
BufferedWriter
OutputStreamWriter
FileWriter
PrintWriter

바이트 스트림


object
InputStream
FileInputStream
FilterInputStream
BufferedInputStream
DataInputStream
ObjectInputStream
OutputStream
FileOutputStream
FilterOutputStream
BufferedOutputStream
DataOutputStream
PrintStream
ObjectOutputStream

InputStream

void close() 입력 스트림을 닫는다

int read()

➔ 입력 스트림으로 부터 한 바이트를 읽어 **int** 형 값을 반환하다.
읽은 바이트가 파일의 끝이면 **-1**을 반환

int read(byte *buffer[]*)

➔ 입력 스트림으로부터 ***buffer*** 배열 크기만큼의 문자를 읽어 ***buffer***에 저장

int read(byte *buffer[]*, int *offset*, int *numbytes*)

➔ 입력스트림으로부터 ***numbytes***에 지정한 만큼의 바이트를 읽어 ***buffer***의 ***offset***위치에 저장하고 읽은 바이트의 개수를 반환

int available() 현재 읽기 가능한 바이트의 수를 반환

int skip(long *numChars*)

➔ ***numChars***로 지정된 바이트 수 만큼을 스킵하고 스킵 된 바이트의 수를 반환

OutputStream

void close() 출력 스트림을 닫는다

void flush() 출력 버퍼에 저장된 모든 데이터를 출력 장치로 전송

void write(int c) c의 하위 8비트를 스트림으로 출력

void write(byte buffer[]) buffer 배열에 있는 바이트들을 스트림으로 출력

void write(byte buffer[], int index, int size)

➔ buffer 배열의 index 위치부터 size 크기 만큼의 바이트들을 스트림으로 출력

FileInputStream

construct

FileInputStream (String *filepath*)

FileInputStream(File *fileObj*)

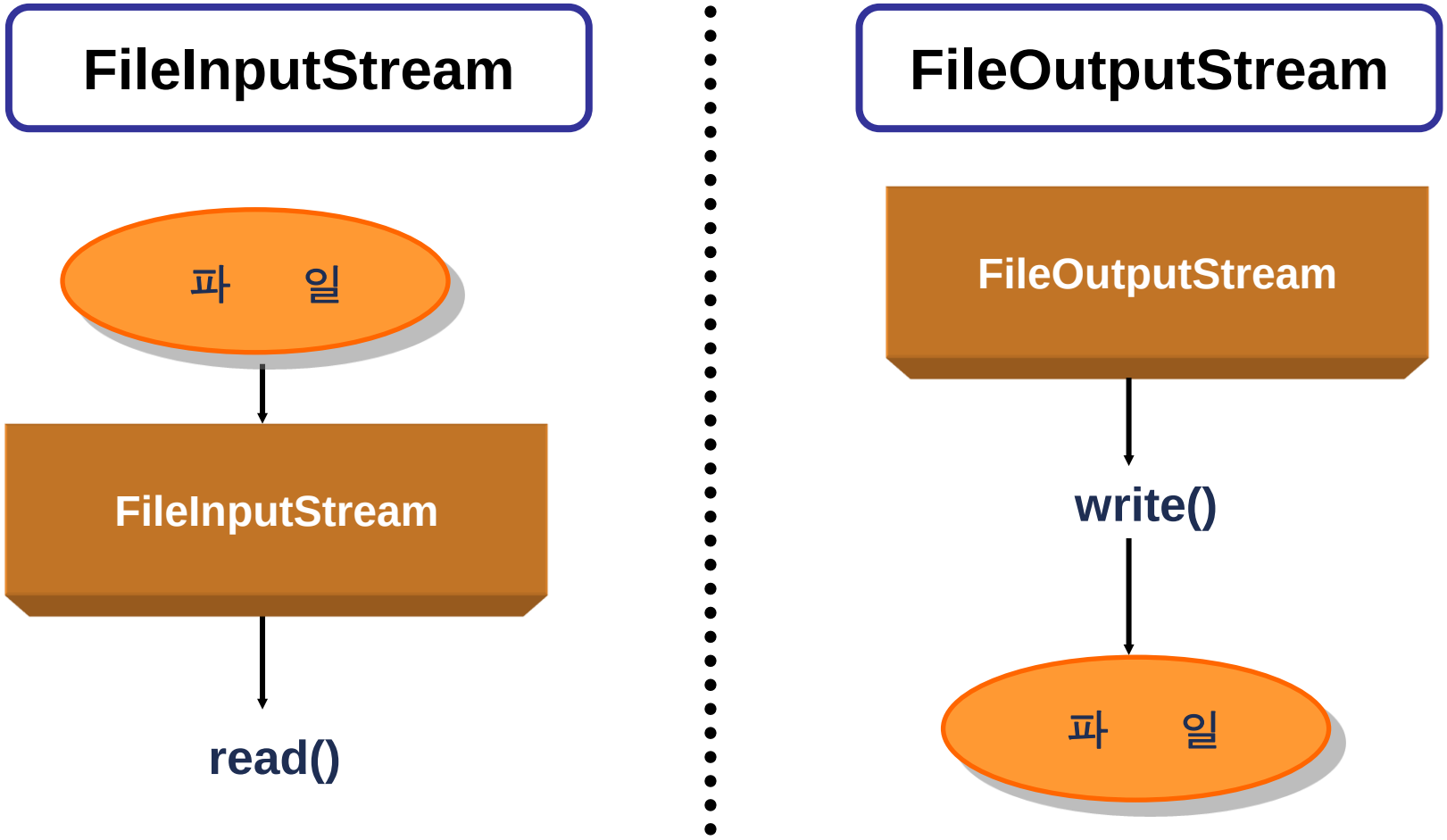
FileOutputStream

construct

FileOutputStream(String *filepath*)

FileOutputStream (String *filepath*, boolean *append*)

FileOutputStream (File *fileObj*)


FileInputStream
파 일
FileInputStream
read()
FileOutputStream
FileOutputStream
write()
파 일

BufferedInputStream

construct

BufferedInputStream(InputStream *InputStream*)

BufferedInputStream(InputStream *InputStream*, int *bufSize*)

BufferedOutputStream

construct

BufferedOutputStream(OutputStream *outputStream*)

BufferedOutputStream(OutputStream *outputStream*, int *bufSize*)

DataInputStream

construct

DataInputStream(InputStream *InputStream)*

DataOutputStream

construct

DataOutputStream(OutputStream *outputStream)*

DataInput, DataOutput 인터페이스를 사용한 클래스

기본 자료형 데이터를 바이트 스트림으로 입,출력

DataInput

method

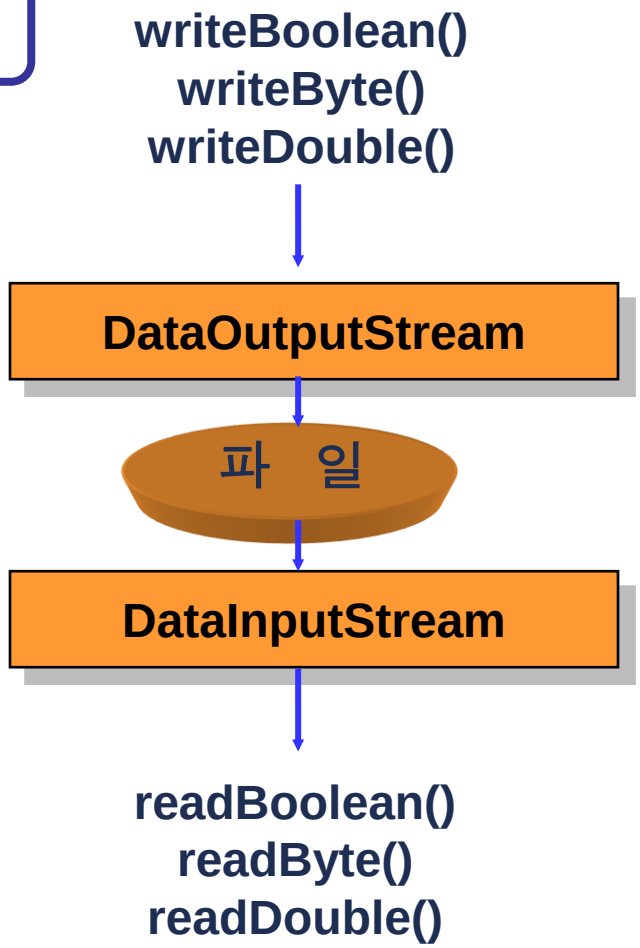
boolean readBoolean(boolean *b*)	스트림으로부터 읽은 boolean을 반환
byte readByte() throws IOException	스트림으로부터 읽은 byte를 반환
char readChar() throws IOException	스트림으로부터 읽은 char를 반환
double readDouble() throws IOException	스트림으로부터 읽은 double을 반환
float readFloat() throws IOException	스트림으로부터 읽은 float를 반환
long readLong() throws IOException	스트림으로부터 읽은 long을 반환
short readShort() throws IOException	스트림으로부터 읽은 short를 반환
int readInt() throws IOException	스트림으로부터 읽은 int를 반환

DataOutput

method

void writeBoolean(boolean b)	b를 스트림으로 출력
void writeByte(int i)	i의 하위 8비트를 스트림으로 출력
void writeBytes(String s)	문자열 s를 스트림으로 출력
void writeChar(int i)	i의 하위 16비트를 스트림으로 출력
void writeChars(String s)	문자열 s를 스트림으로 출력
void writeDouble(double d)	d를 스트림으로 출력
void writeFloat(float f)	f를 스트림으로 출력
void writeInt(int i)	i를 스트림으로 출력
void writeLong(long l)	l을 스트림으로 출력
void writeShort(short s)	s를 스트림으로 출력

동 작 방 식


writeBoolean()
writeByte()
writeDouble()
DataOutputStream
파 일
DataInputStream
readBoolean()
readByte()
readDouble()

Reader

void close() 입력 스트림을 닫는다

int read()

➔ 다음 문자를 읽어 반환한다. 입력 스트림에 읽을 문자가 없으면 대기한다. 읽은 문자가 파일의 끝이면 **-1**을 반환

int read(char *buffer[]*)

➔ 입력 스트림으로부터 ***buffer*** 배열 크기만큼의 문자를 읽어 ***buffer***에 저장

int read(char *buffer[]*, int *offset*, int *numChars*)

➔ 입력스트림으로부터 ***numChars***에 지정한 만큼의 문자를 읽어 ***buffer***의 ***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 **true**를 반환

Writer

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

FileReader

construct

FileReader (String *filepath*)

FileReader(File *fileObj*)

FileWriter

construct

FileWriter(String *filepath*)

FileWriter (String *filepath*, boolean *append*)

FileWriter (File *fileObj*)

BufferedReader

BufferedReader(Reader *inputStream*)
BufferedReader(Reader *inputStream*, int *bufSize*)
String readLine() throws IOException 라인 단위로 읽어 온다.

BufferedWriter

BufferedWriter(Writer *outputStream*)
BufferedWriter(Writer *outputStream*, int *bufSize*)
void newLine() throws IOException 새로운 라인에 출력

Thread API

Thread 생성자

- 생성자

생성자	설명
Thread( )	스레드를 생성
Thread(String name)	Name 이라는 이름을 가진 스레드 생성
Thread(Runnable r)	Runnable 인터페이스를 구현한 클래스 객체로 스레드를 생성

Thread 메서드

- 메서드

메서드	설명
static void sleep(long msec)	msec 시간만큼 스레드를 대기시킨다.(1/1000 초)
String getName( )	스레드의 이름을 반환한다.
void setName(String name)	스레드의 이름을 설정한다.
void start( )	스레드를 시작한다.(run( ) 메소드를 호출)
void run( )	스레드가 해야할 일을 정의한다.
void setPriority(int p)	스레드의 우선순위를 설정한다.
int getPriority( )	설정된 우선순위를 반환한다.

Thread 생성방법

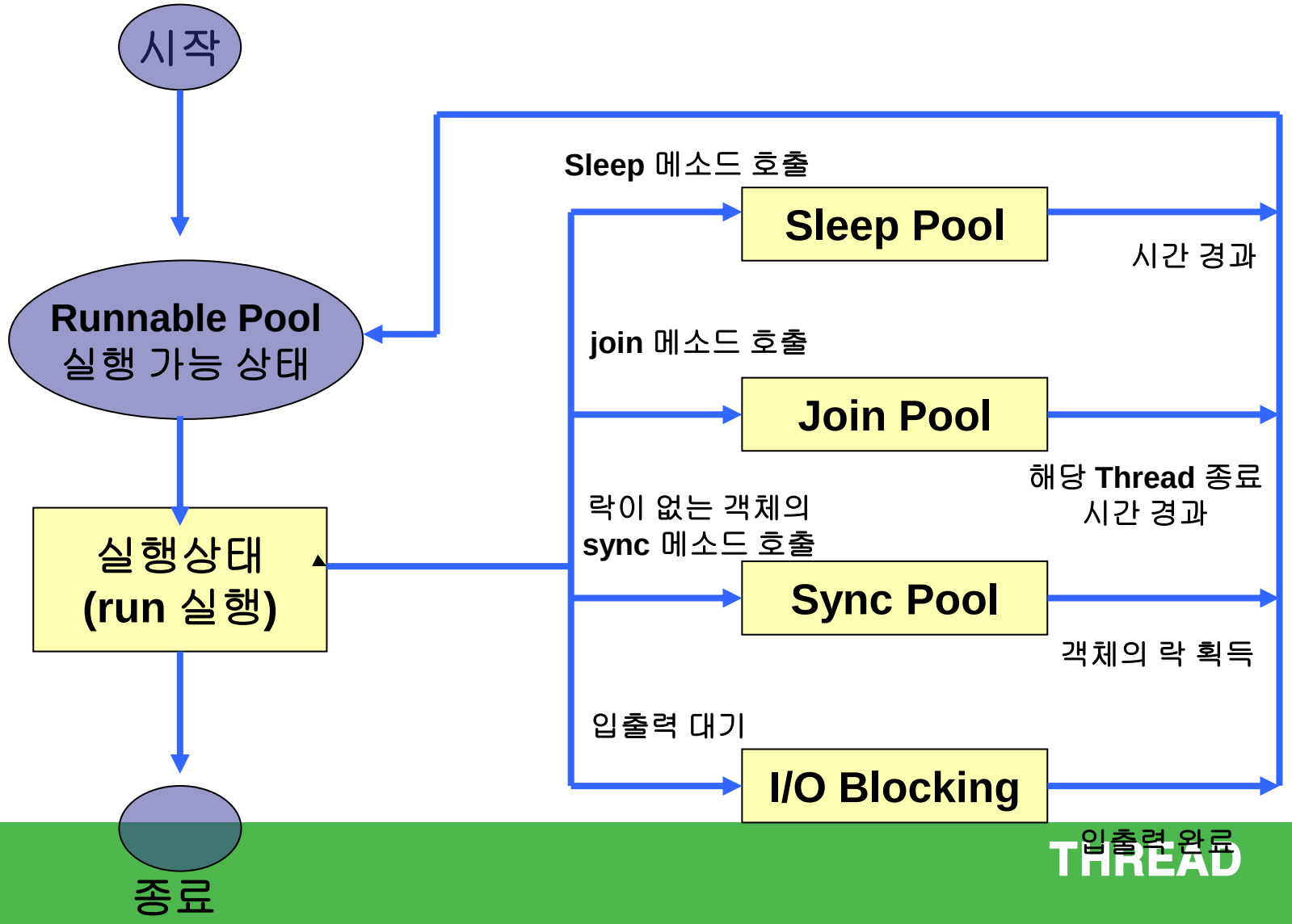
- Thread 를 생성하는 2가지 방법

Thread 클래스로부터 직접 상속 받아 스레드를 생성
Runnable 인터페이스를 사용하는 방법

```
class ThreadTest extends
                        Thread {

  ThreadTest tt = new
                    ThreadTest();
  tt.start();

  public void run() {
  }
}
```

```
class ThreadTest implements
                        Runnable {
  ThreadTest tt = new
                    ThreadTest();
  Thread t = new Thread(tt);
  t.start();

  public void run() {
  }
}
```

Thread 상태


시작
Runnable Pool
실행 가능 상태
실행상태
(run 실행)
종료
Sleep 메소드 호출
Sleep Pool
시간 경과
join 메소드 호출
Join Pool
해당 Thread 종료
시간 경과
락이 없는 객체의
sync 메소드 호출
Sync Pool
객체의 락 획득
입출력 대기
I/O Blocking
입출력 완료

Thread 우선순위

- 스레드에 우선 순위를 부여하여 우선 순위가 높은 스레드에게 실행의 우선권을 부여할 수 있다**(JVM**마다 다를 수 있다**)**

- **setPriority(int priority)** 메소드를 이용하여 우선 순위 부여

- **getPriority()** 메소드를 이용하여 설정된 우선 순위를 가져온다**.**

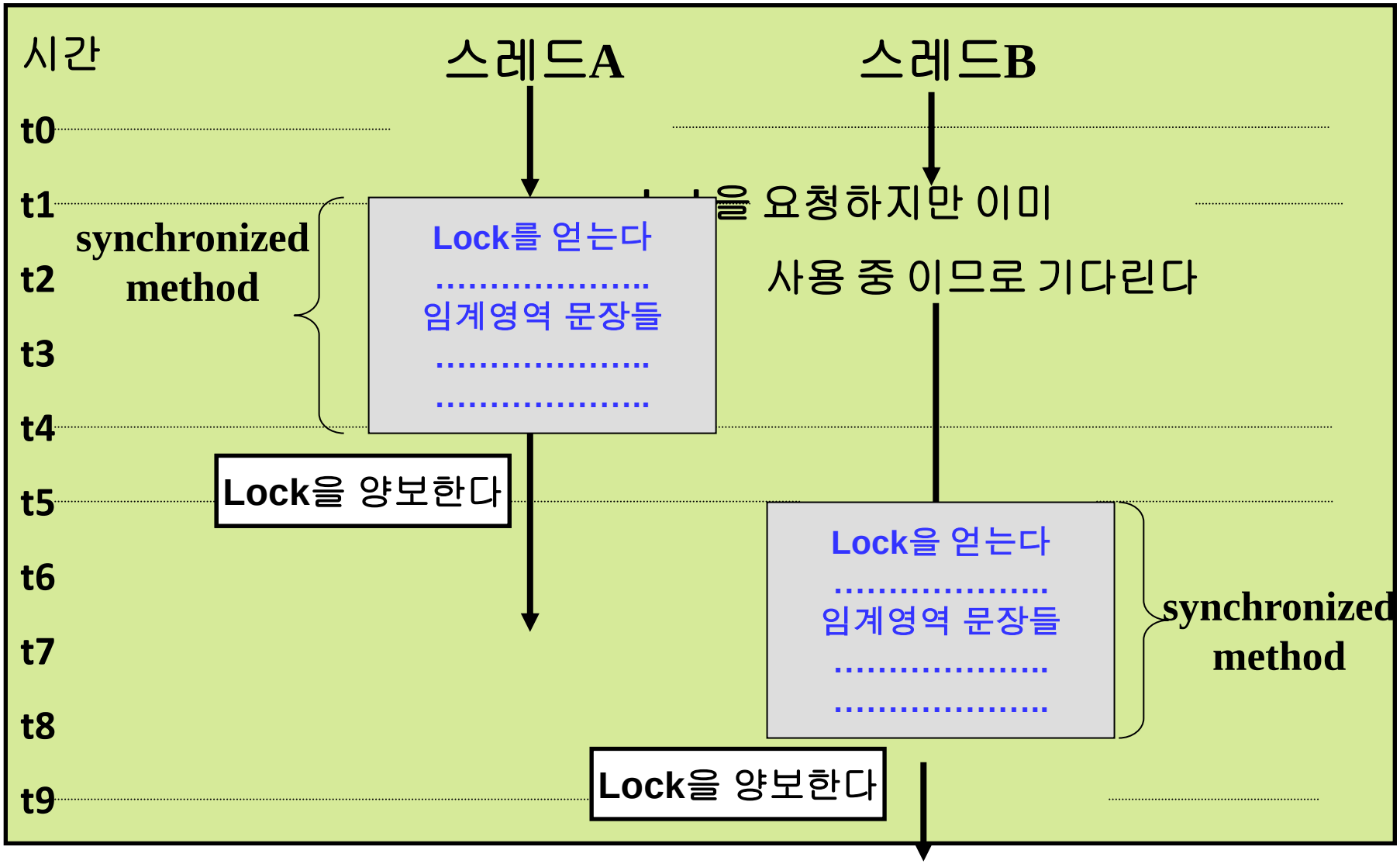
- 우선 순위를 지정하기 위한 상수 제공

  static final int MAX_PRIORITY 우선순위 **10**

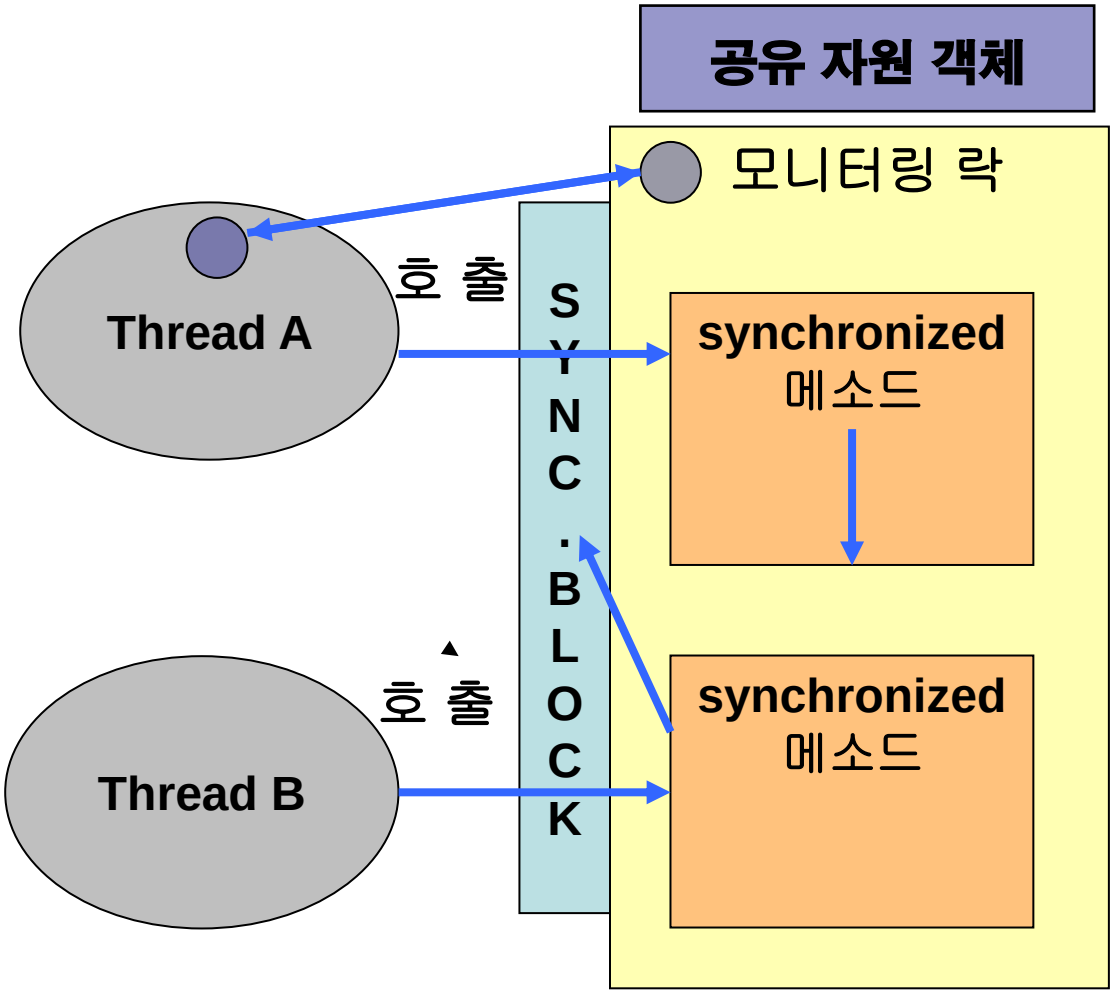
  static final int MIN_PRIORITY 우선순위 **1**

  static final int NORM_PRIORITY 우선순위 **5**

Thread 동기화


시간
스레드A
스레드B
t0
t1
t2
t3
t4
t5
t6
t7
t8
t9
synchronized method
Lock를 얻는다
.....
임계영역 문장들
.....
.....
을 요청하지만 이미
사용 중 이므로 기다린다
Lock을 양보한다
Lock을 얻는다
.....
임계영역 문장들
.....
.....
synchronized method
Lock을 양보한다

Thread 동기화 동작원리



Network API

목차

1. 관련 용어
2. **java.net** 패키지
3. 인터넷 주소와 **URL**
4. **TCP** 소켓 프로그래밍

관련용어 – 소켓(SOCKET)

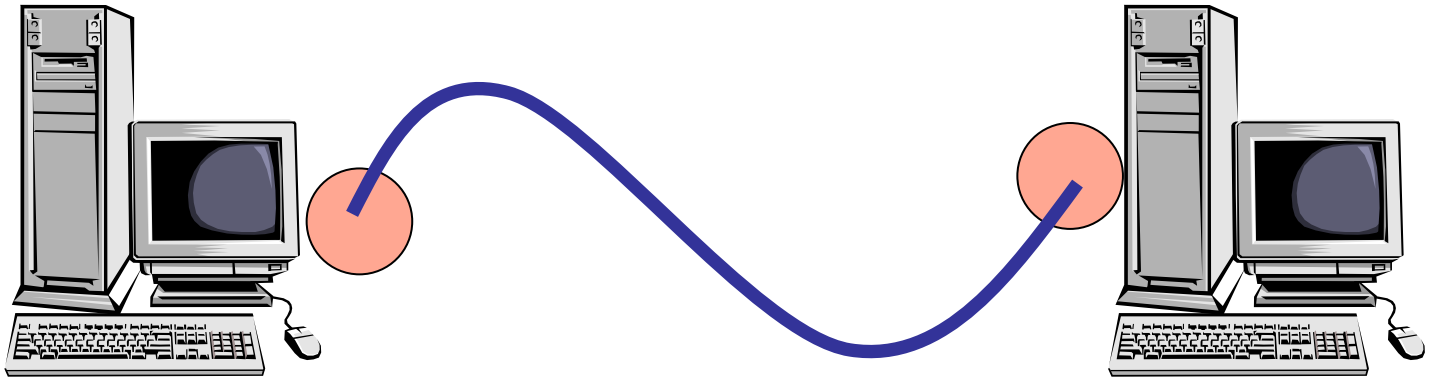
소켓 : 컴퓨터가 연결된 통신의 끝점.

소켓에 쓰는 일은 상대에게 데이터를 전달

소켓에서 읽는 일은 상대가 전송한 데이터를 수신하는 것.

자바에서 사용하는 소켓은 **TCP**와 **UDP**를 이용한다.

웹은 소켓 통신을 사용한다.**(TCP)**



관련용어 – 호스트, 포트

호스트(Host)

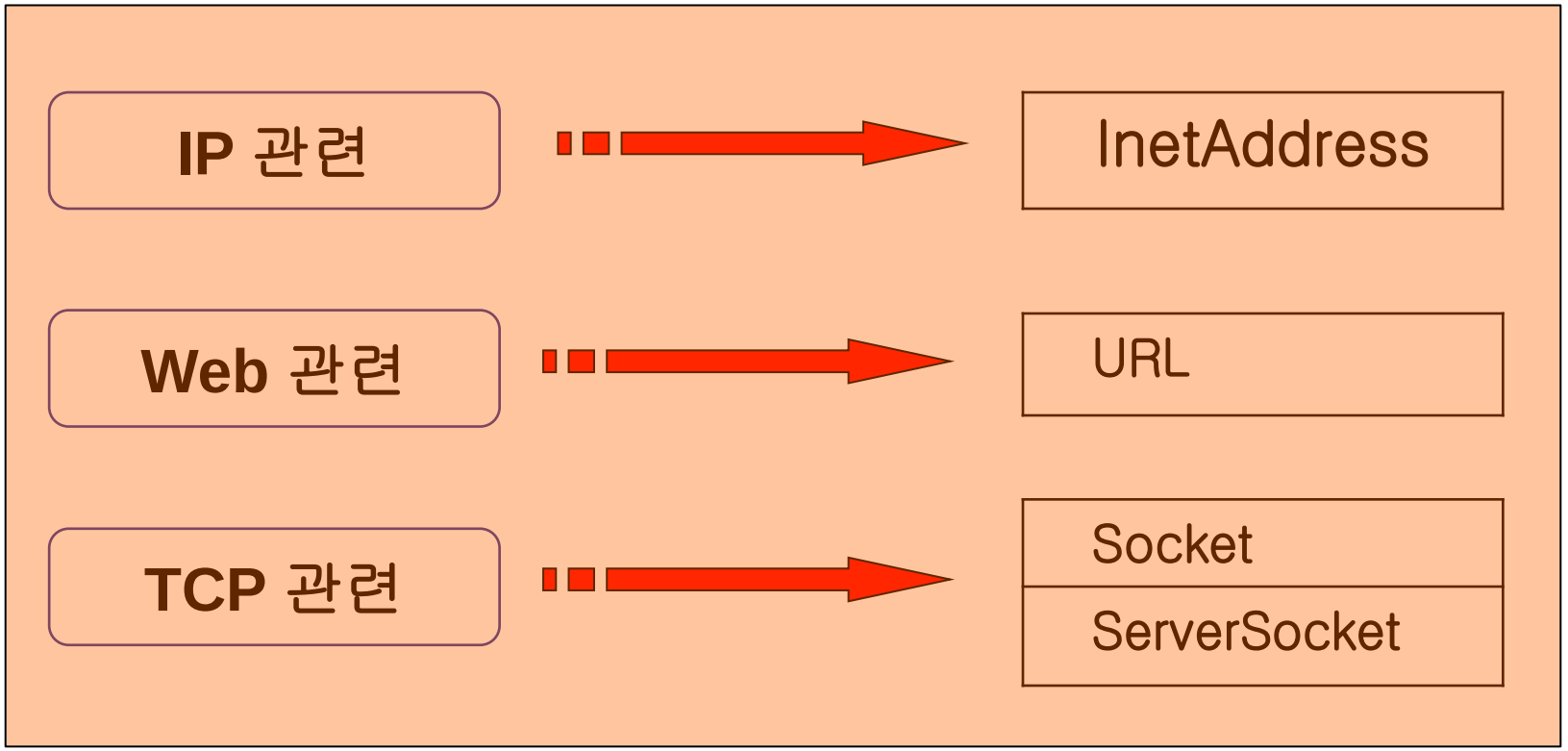
호스트 주소 : 하나의 컴퓨터에 할당된 고유 이름
인터넷 상에서 IP 주소나 도메인명으로 나타난다.

포트(Port)

포트번호 : 한 컴퓨터에서 여러 서비스의 제공을 가능하게 함.
한 호스트에 있는 여러 개의 서비스를 구분하기 위해서 사용

하나의 호스트는 여러 개의 포트를 가질 수 있다.
서버 어플리케이션은 클라이언트의 요청을 위해 대기할 때 미리 정해진 포트를 감시한다.
호스트는 전화번호에 포트는 내선번호에 비교할 수 있다.

java.net API



API - InetAddress

method

String getHostAddress() 주소 정보를 나타내는 문자열을 반환

String getHostName() 컴퓨터 이름을 나타내는 문자열을 반환

InetAddress getLocalHost() 현재 컴퓨터를 나타내는 **InetAddress** 객체를 반환

InetAddress getByName(String *hostName*)

➔ ***hostName***으로 지정된 컴퓨터를 나타 내는 **InetAddress** 객체를 반환

InetAddress[] getAllByName(String *hostName*)

➔ ***hostName***으로 지정된 모든 컴퓨터(하나의 도메인 이름으로 여러 대의 컴퓨터를 사용하는 경우)를 나타내는 **InetAddress** 객체들의 배열을 반환

API – URL [protocol://host:port/filename(경로포함)]

construct

URL(String protocol, String host, int port, String file)

URL(String protocol, String host, String file)

URL(String urlString)

method

String getFile() ➔ URL의 파일 이름을 반환

String getHost() ➔ URL의 호스트 이름을 반환

String getPort() ➔ URL의 포트 번호를 반환. 묵시적인 포트인 경우 –1 반환

String getProtocol() ➔ URL의 프로토콜 이름을 반환

String toExternalForm() ➔ 전체 URL의 문자열 객체를 반환

InputStream openStream() ➔ 지정된 URL로부터 정보를 읽어들이기 위한 객체를 반환

API - ServerSocket

Server

construct

ServerSocket(int port)

method

Socket accept()

➔ 클라이언트의 요청을 받아들인 다음 클라이언트와 연결된 소켓 클래스 객체를 반환함.

void close() 서버 소켓을 닫는다.

API - Socket

Client

construct

Socket(String *hostName*, int *port*)

method

InputStream getInputStream()

➔ 현재의 소켓과 관련된 InputStream 객체를 반환

OutputStream getOutputStream()

➔ 현재의 소켓과 관련된 OutputStream 객체를 반환

void close() 소켓을 닫는다

API - Socket

method

InetAddress getInetAddress()

➔ 현재 소켓에 연결된 컴퓨터의 주소를 반환

InetAddress getLocalAddress()

➔ 현재 소켓을 사용하고 있는 컴퓨터의 주소를 반환

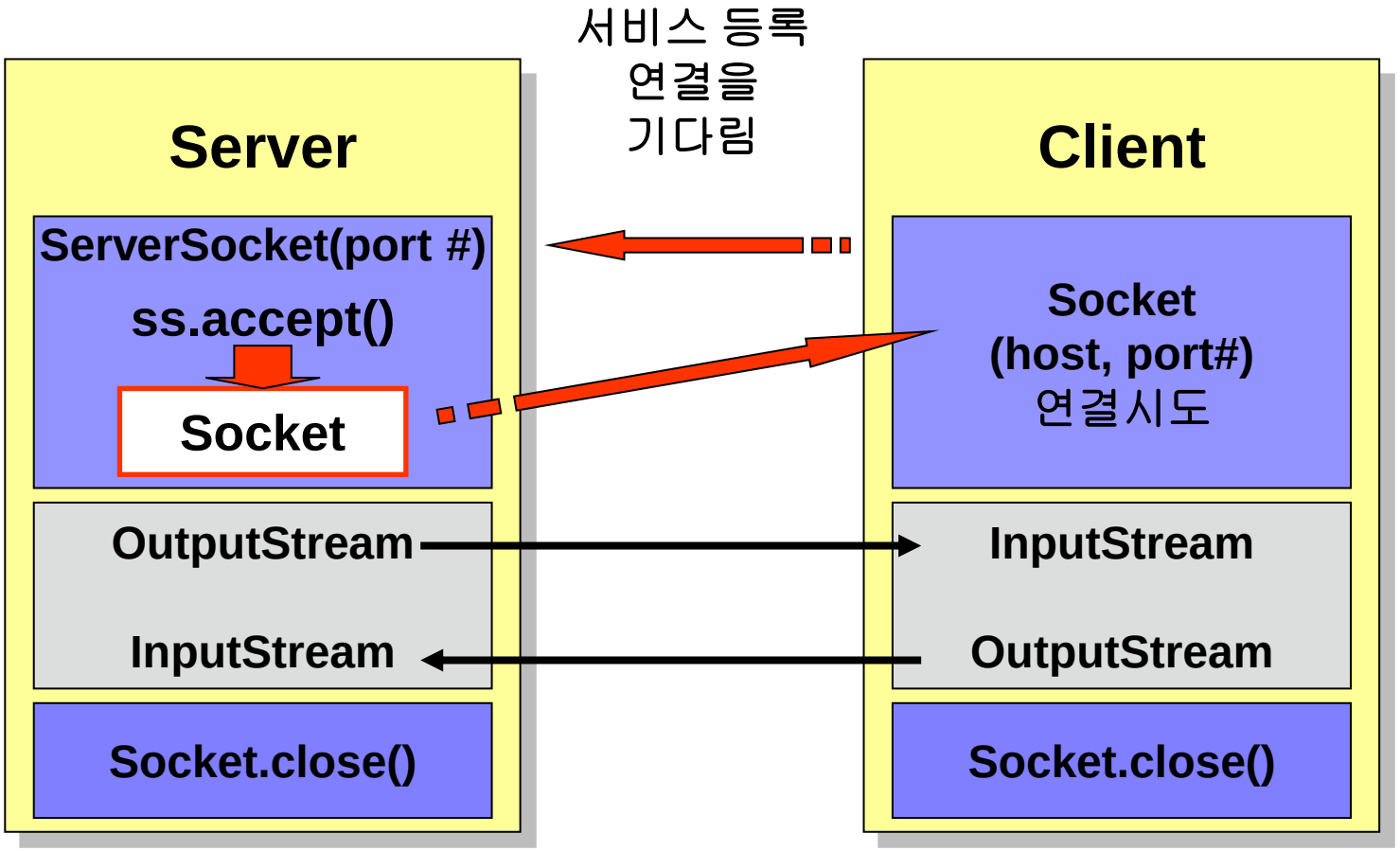
int getPort()

➔ 현재 소켓에 연결된 컴퓨터의 포트 번호를 반환

int getLocalPort()

➔ 현재 소켓이 사용하고 있는 포트 번호를 반환

동작방식


서비스 등록
연결을
기다림
Server
ServerSocket(port #)
ss.accept()
Socket
OutputStream
InputStream
Socket.close()
Client
Socket
(host, port#)
연결시도
InputStream
OutputStream
Socket.close()