chapter 13.

스레드

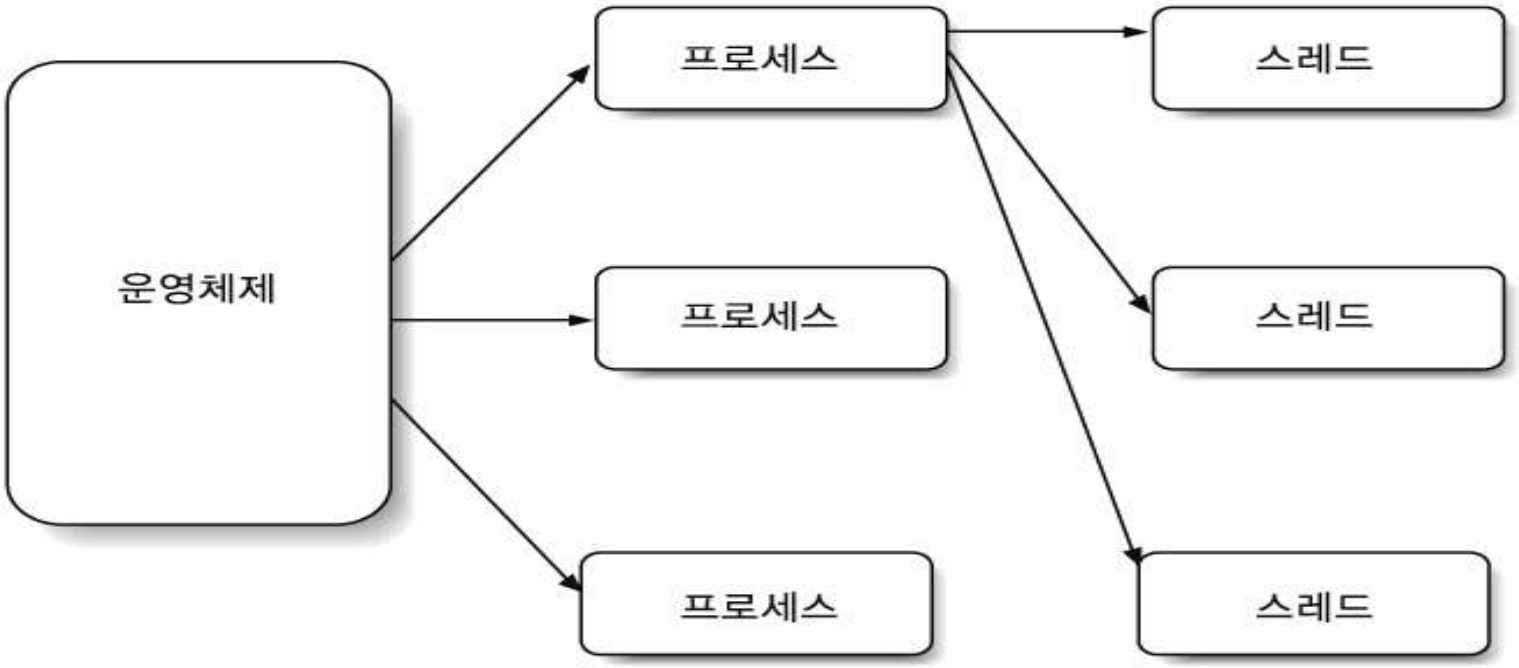
❍ 멀티 태스킹

- 프로세스란 운영체제에서 실행중인 하나의 프로그램을 말한다.
- 멀티 프로세스란 두 개 이상의 프로세스가 실행되는 것을 말한다.
- 멀티 태스킹이란 두 개 이상의 프로세스를 실행하여 일을 처리하는 것을 말한다.

❍ 멀티 스레드

- 스레드란 프로세스 내에서 실행되는 세부 작업 단위이다.
- 멀티 스레드란 하나의 프로세스에서 여러 개의 스레드가 병행적으로 처리되는 것을 말한다.

❍ 프로세스와 스레드의 관계

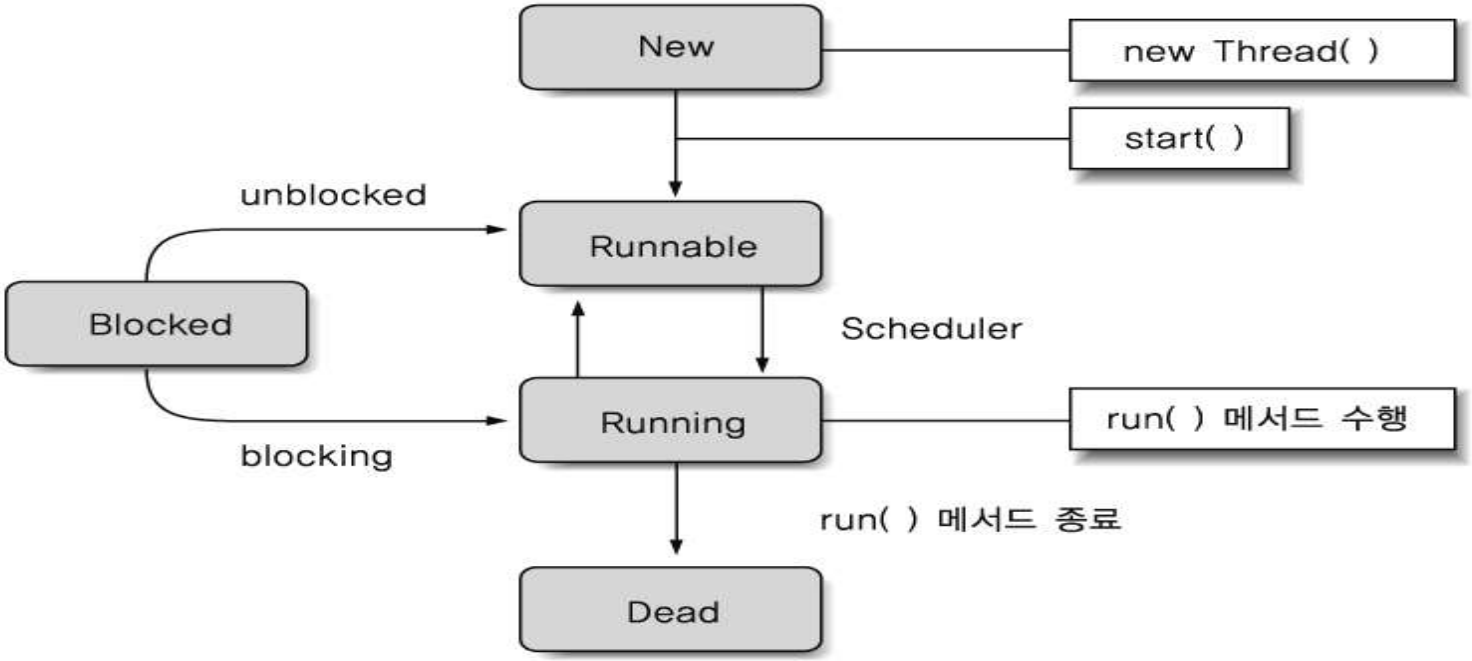


[그림 13-1] 프로세스와 스레드의 관계

스레드의 생명주기

- 스레드는 Thread 객체가 생성되면 생명주기를 갖게 되는데 크게 5가지로 나누게 된다.
- New – 스레드가 만들어진 상태.
- Runnable – 스레드 객체가 생성된 후에 start() 메서드를 호출하면 Runnable 상태로 이동하게 된다.
- Running – Runnable 상태에서 스레드 스케줄러에 의해 Running 상태로 이동하게 된다.
- Blocked – 스레드가 다른 특정한 이유로 Running 상태에서 Blocked 상태로 이동하게 된다.
- Dead – 스레드가 종료되면 그 스레드는 다시 시작할 수 없게 된다.

❍ 스레드의 생명주기



[그림 13-2] 스레드 생명 주기

❍ 스레드의 생성 방법

- Thread 클래스를 상속 받는 방법
- Runnable 인터페이스를 구현하는 방법

❍ Thread 클래스 생성자

[표 13-1] Thread 클래스의 주요 생성자

생성자	설명
Thread( )	가장 일반적인 형태의 생성자다. 이 생성자를 이용해서 Thread 객체를 생성하게 되면 Thread의 이름은 'Thread-' +n의 형태가 된다.
Thread(Runnable target)	Runnable 객체를 이용해서 Thread 객체를 생성할 수 있는 생성자다.
Thread(Runnable target, String name)	Runnable 객체를 이용해서 Thread 객체를 생성할 수 있는 생성자며, 스레드의 이름을 지정할 수 있는 생성자다.
Thread(String name)	스레드의 이름을 지정하면서 Thread 객체를 생성할 수 있는 생성자다.

❍Thread 클래스의 주요 메서드

[표 13-2] Thread 클래스의 주요 메서드

반환형	메서드	설명
static void	sleep(long millis)	millis에 지정된 시간만큼 대기한다.
String	getName( )	스레드의 이름을 반환한다.
void	setName(String name)	스레드의 이름을 반환한다.
	start( )	스레드를 시작시킨다.
int	getPriority( )	스레드의 우선순위를 반환한다.
void	setPriority(int newPriority)	스레드의 우선순위를 지정한다.
	join( )	현재 스레드는 join( ) 메서드를 호출한 스레드가 종료할 때까지 기다리게 된다.
static void	yield( )	수행중인 스레드 중 우선순위가 같은 다른 스레드에게 제어권을 넘긴다.
static Thread	currentThread( )	현재 수행되는 스레드 객체를 리턴한다.

❍Thread 클래스를 이용한 스레드 생성 방법

```
public class CreateThread extends Thread{
        public void run(){

        }
        public static void main(String[] args){
                CreateThread ct = new CreateThread();
                ct.start();
        }
}
```

■ 예제[13-1] SingleThreadEx.java

❍Runnable 인터페이스를 이용한 스레드 생성 방법

```
public class CreateRunnable implements Runnable{
        public void run(){
        }
        public static void main(String[] args){
                CreateRunnable ct = new CreateRunnable();
                Thread t = new Thread(ct);
                t.start();
        }
}
```

■ 예제[13-2] SingleRunnableEx.java

join() 메서드 사용법

- join() 메서드는 join() 메서드를 호출한 스레드가 종료할 때까지 종료할 때까지 현재의 스레드를 기다리게 된다.
- 예제[13-4] JoinEx.java

❍ 스레드 스케줄링 방식

- 선점형 스레드 스케줄링 방식은 스레드의 우선권을 가지고 우선순위가 높은 스레드를 먼저 수행시키는 방식이다.
- 협력형 스레드 스케줄러는 실행중인 스레드가 CPU 사용권을 다른 스레드에게 넘길 때까지 기다리는 방식이다.
- JVM은 우선순위에 따른 선점형 스레드 스케줄링 방식을 사용하고 있다.

❍ 스레드 스케줄러

- 멀티 스레드가 수행될 때 어떤 스레드가 먼저 수행될지는 스레드 스케줄러가 결정하게 된다.
- 자바 애플리케이션에서는 우선순위가 높은 선점형 스레드 스케줄러를 사용하고 있다.
  - 예제[13-5] MultiThreadEx.java

❍ 스레드 우선순위

- Thread 클래스에서는 스레드의 우선순위를 부여하는 setPriority(int newPriority) 메서드를 제공한다.

❍ Thread 클래스의 우선순위를 정하는 멤버변수

[표 13-3] Thread 클래스의 우선순위를 정하는 멤버변수

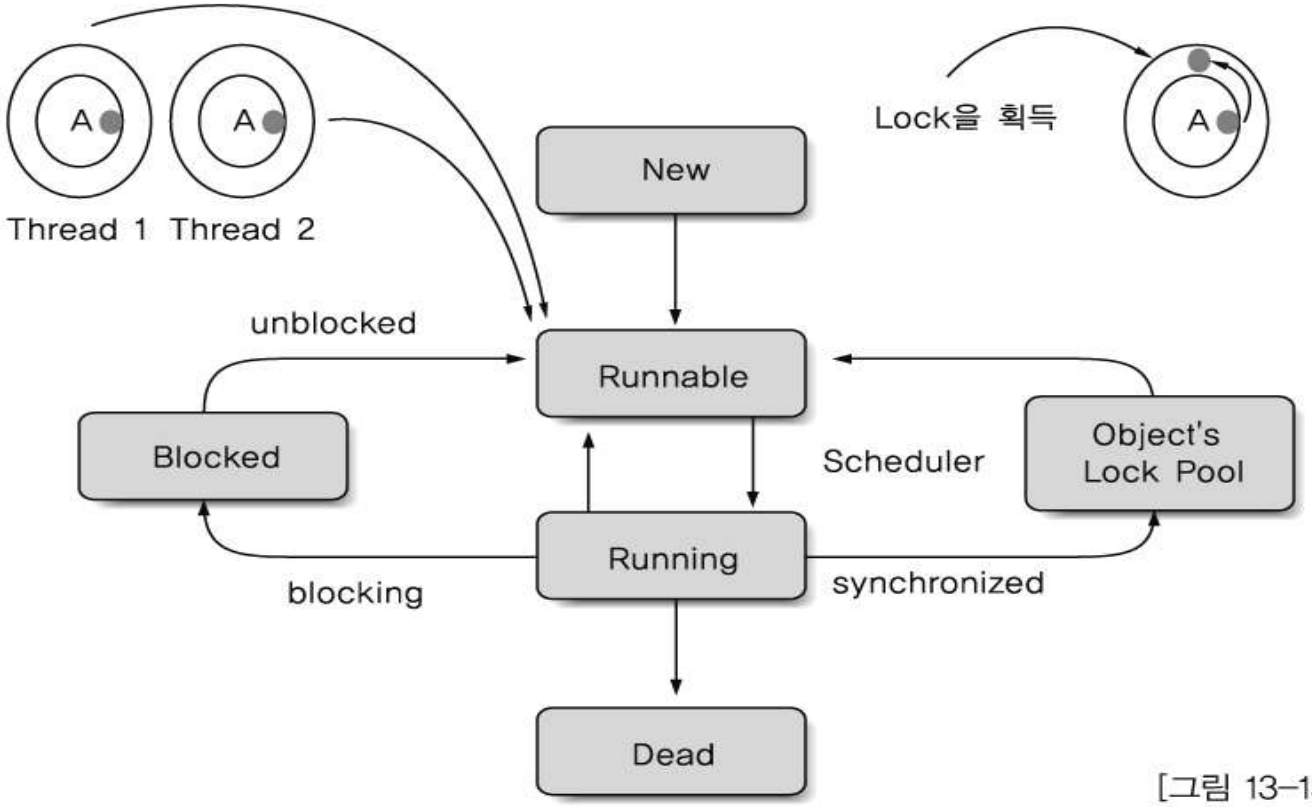
반환형	메서드	설명	사용 예
static int	MAX_PRIORITY	스레드가 가질 수 있는 최대 우선순위값(10)	setPriority(Thread.MAX_PRIORITY)
	NORM_PRIORITY	스레드가 가질 수 있는 기본 우선순위값(5)	setPriority(Thread.MIN_PRIORITY)
	MIN_PRIORITY	스레드가 가질 수 있는 최소 우선순위값(1)	setPriority(Thread.NORM_PRIORITY)

- 예제[13-6] PriorityChangeEx.java

동기화

- 임계영역이란 멀티 스레드에 의해 공유자원이 참조될 수 있는 코드의 범위를 말한다.
- 멀티 스레드 프로그램에서 임계영역을 처리하는 경우 심각한 문제가 발생할 수 있다.
- 이러한 상황을 해결할 수 있는 방법이 동기화를 이용하는 것이다.
- 동기화를 처리하기 위해 모든 객체에 락(lock)을 포함 시켰다.
- 락이란 공유 객체에 여러 스레드가 동시에 접근하지 못하도록 하기 위한 것으로 모든 객체가 힙 영역에 생성될 때 자동으로 만들어 진다.

synchronized 흐름도



[그림 13-13] synchronized 흐름도

❍동기화 방법

- 메서드의 동기화 방법

```
public synchronized void synchronizedMethod(){
        //임계영역 코딩
}
```

- 특정 블록의 동기화 방법

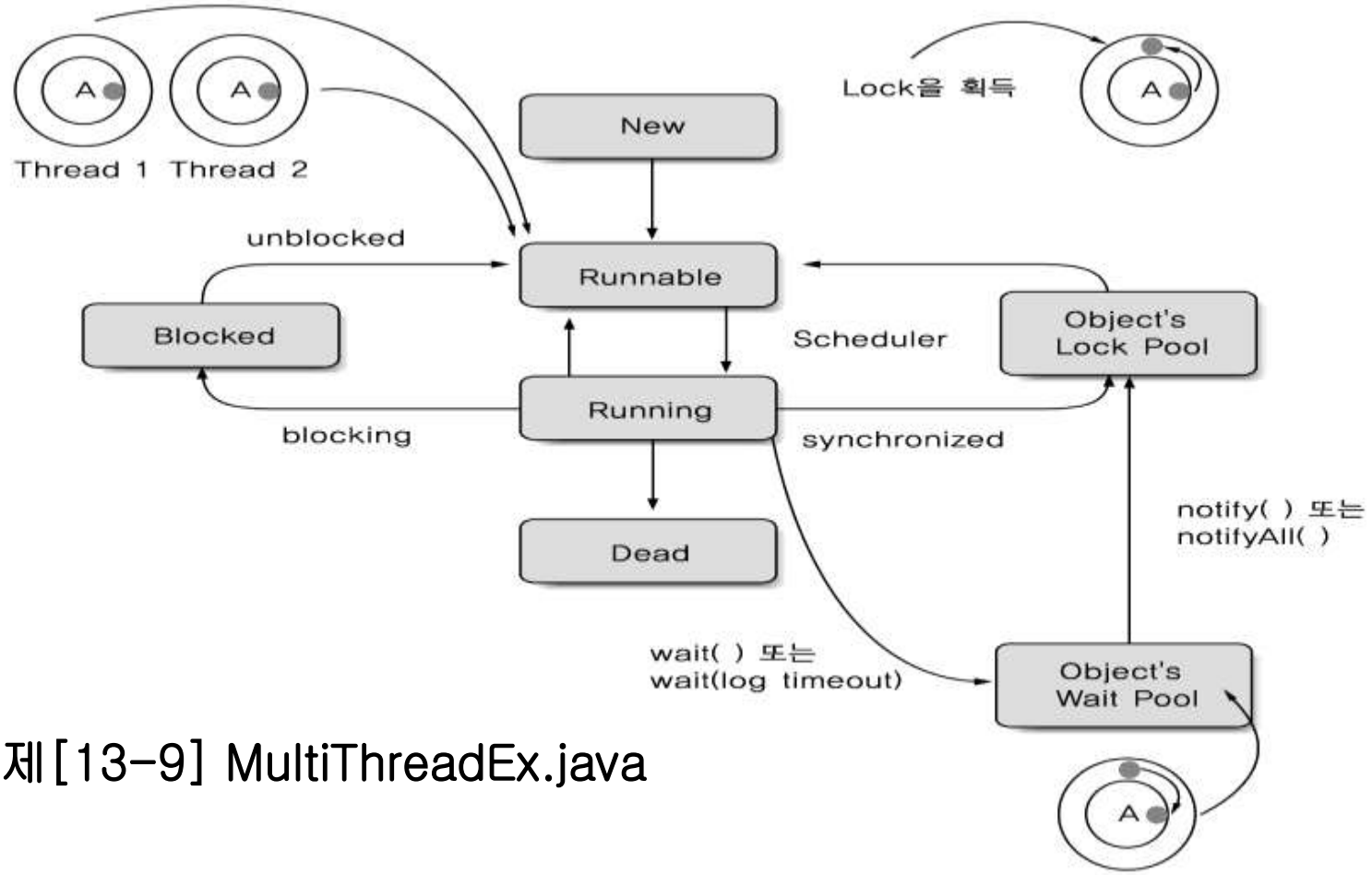
```
public void nomalMethod(){
        synchronized(동기화할 객체 또는 클래스명){
                //임계영역 코딩
        }
}
```

■ 예제[13-8] SynchronizedEx.java

- 공정(fairness)
  - 여러 개의 스레드가 하나으 컴퓨팅 자원을 사용하기 위해 동시에 접근하는 프로그램을 작성할 경우 모든 스레드는 공정하게 그 자원을 사용할 수 있도록 해 주어야 한다.
- 기아(starvation)
  - 하나 또는 그 이상의 스레드가 자원을 얻기 위해 Blocked 상태에 있고, 그 자원을 얻을 수 없게 되면 다른 작업을 못하는 상태를 말한다.
- 교착상태(deadlock)
  - 두 개 이상의 스레드가 만족하지 못하는 상태로 계속 기다릴 때 발생한다.

- Object 클래스의 wait(), notify(), notifyAll()
  - 동기화된 스레드는 동기화 블록에서 다른 스레드에게 제어권을 넘기지 못한다.
  - 이와 같이 동기화된 블록에서 스레드간의 통신(제어권을 넘김)하기 위해서는 wait(),notify(),notifyAll() 메서드를 사용해야 한다.
  - 이 메서드를 사용할 때 주의 해야할 점은 synchronzied 블록에서만 의미가 있다.
  - Synchronized 블록이 아닌 경우에 사용할 경우 java.lang.IllegalMonitorStateException 이 발생한다.

Object의 wait(),notify(), notifyAll()의 흐름도



[그림 13-17] wait( ), notify( ), notifyAll( ) 메서드를 호출할 때의 흐름도

- 예제[13-9] MultiThreadEx.java