

자바의 멀티스레드
<p>가. 멀티스레드란 하나의 프로그램내에서 여러 개의 작은 작업이 동시에 수행되는 것을 말한다.</p> <p>나. 멀티스레드 구현 방법</p> <p>첫째, Thread 클래스 상속받는법.</p> <p>Thread 클래스를 상속받아 멀티스레드를 구현하면 방법은 쉽지만, 단일 상속만 가능하다는 단점이 있다.</p> <p>둘째, Runnable 인터페이스를 상속받는 법</p> <p>다중 상속을 받을 수 있다는 장점이 있다. 하지만 멀티스레드를 시작하는 start() 메서드가 없기 때문에 이 인터페이스를 상속받은 자식클래스 객체를 Thread 클래스 생성자 인자값으로 넘겨줘서 다시 한번더 객체를 생성해서 start()메서드를 호출해야 한다는 단점이 있다.</p> <p>다. 멀티스레드에서 start() 메서드를 호출하면 멀티스레드가 시작되면서 run() 메서드를 자동 호출한다. 명시적인 run()메서드를 호출하지 않아도 start() 메서드만 호출하면 run() 메서드는 자동 호출하도록 자바는 설계 되어져 있다.</p> <p>바로 자동 호출되는 run()메서드 내에 멀티스레드 문장을 구현한다.</p>

쓰레드 스케줄링 메서드
<ol style="list-style-type: none"> <li>1. sleep(), suspend(), wait(),join() : 쓰레드를 일시정지 시킨다.</li> <li>2. resume(), notify(), interrupt() : 일시정지 상태를 벗어나 다시 실행대기 상태로 만든다.</li> <li>3. yield() : 실행중에 자신에게 주어진 실행시간을 다른 쓰레드에게 양보하고 자신은 실행대기 상태가 된다.</li> <li>4. stop() : 쓰레드를 즉시 종료 시킨다.</li> <li>5. sleep(long millis) -&gt;밀리세컨드 1000분의 1초 단위. 일정시간동안 쓰레드를 멈추게 한다.</li> <li>6. sleep()에 의해서 일시정지 상태가 된 쓰레드는 지정된 시간이 다 되거나 interrupt() 가 호출되면 InterruptedException 예외가 발생해 다시 실행대기 상태가 된다.</li> <li>7. suspend()에 의해서 일시 정지된 쓰레드는 resume()을 호출해야 다시 실행대기 상태가 된다.</li> </ol>

8. `suspend()`, `resume()`, `stop()` 은 자바 API를 보시면 전에는 사용되었지만 앞으로 사용하지 않을 것을 권장한다는 뜻의 `@Deprecated` 애너테이션으로 정의 되어 있다. 이 3개의 메서드는 쓰레드를 교착상태(`dead-lock`)로 만들기 쉽게 한다. 교착상태란 두 쓰레드가 자원을 점유한 상태에서 서로 상대방이 점유한 자원을 사용하려고 기다리느라 진행이 멈춰있는 상태를 말한다.

#### 쓰레드의 동기화

1. 동기화란 한 쓰레드가 진행 중인 작업을 다른 쓰레드가 간섭하지 못하도록 막는 것을 말한다.

2. 하나의 쓰레드에 의해서만 처리할 수 있도록 하는 영역을 임계영역이라 한다.

3. 임계영역을 지정하기 위해서는 하나의 쓰레드가 이 영역에 진입할 때 락을 걸어서 다른 쓰레드가 수행되지 못하도록 하고, 이 영역에서 벗어날 경우 락을 풀어서 다른 쓰레드가 수행하도록 한다. 즉 임계영역 내에서는 한 순간에 하나의 쓰레드만 동작하도록 제약을 주어야 한다. 이러한 제약을 위해서는 자바에서는 동기화 기법을 제공하는데 하나의 쓰레드만 동작하도록 하고자 하는 메서드나 블록에 `synchronized` 로 지정한다.

#### 4. 동기화 기법

가. 메서드 전체를 동기화 처리

```
public synchronized void 메서드명(){
    //임계영역
}
```

나. 특정한 영역을 임계 영역으로 지정

```
synchronized(객체의 참조변수){
    //임계영역
}
```

5. 동기화 영역내에서 한 쓰레드가 락을 보유하고 계속해서 작업을 진행할 상황이 아니면 일단 락을 풀고 기다리게 하는 경우가 발생한다. 예를 들면 출금계좌에 잔고가 부족할때 한 쓰레드가 락을 가진 상태로 돈이 입금될 때까지 무한정 기다리면 다른 쓰레드는 모두 해당객체의 락을 기다리느라 아무런 작업을 못하는 경우가 발생한다.

이럴 경우 일단 `wait()`를 호출하여 락을 반납하고 기다리게 한다. 그러면 다른 쓰레드가 락을 얻어 해당 객체에 대해서 작업을 수행할 수 있게 된다.

나중에 다시 작업할 상황이 되면 `notify()`를 호출해서 중단했던 쓰레드가 다시 락을 얻어 작업을 진행할 수 있게 한다.

6. `notify()`를 호출했다면 `wait()`에 의해서 락을 반납하고 대기실에서 대기 중인 쓰레드 중에서 하나를 임의로 선택해서 통지할 뿐, 락을 반납한 해당 쓰레드를 선택해서 통지할수 없다. 운 좋게 락을 반납한 해당 쓰레드가 통지를 받으면 다행인데, 그렇지 않

으면 계속 통지를 받지 못하고 무한정 기다리게 되는 현상이 발생한다. 이런 현상을 ‘기아 현상’ 이라 한다. 이 현상을 방지하기 위해 `notify()` 대신 `notifyAll()`을 사용해야 한다. 이 메서드는 대기실에 대기중인 모든 스레드에게 통지를 하기 때문에 기아 현상은 막았지만, 통지 받은 스레드가 서로 락을 얻기 위해 경쟁을 하게 된다. 이처럼 여러 스레드가 서로 락을 얻기 위해 경쟁하는 것을 ‘경쟁 상태’ 라 한다.

이 경쟁 상태를 개선하기 위해서 각 스레드를 구분해서 통지하는 것이 필요하다.

Lock과 컨디션을 이용하면 부분적인 선별적 통지가 가능하다. 컨디션을 사용하면 `wait()` 대신 `await()`를 사용하면 되고, `notify()` 대신 `signal()`을 이용한다.