Synchronization Ⅲ - 쓰레드 접근순서 동기화

쓰레드 접근순서의 동기화 필요성

- **NewsWriter, NewsReader의 인스턴스는 NewsPaper의 인스턴스에 동시접근하고 있다.**
- **Writer는 글을 쓰고, Reader는 글을 가져간다. 그러므로 실행순서는 Writer가 먼저 되야 한다.**
  이러한 경우는 메모리 접근의 동시접근 제한 뿐만이 아니라 실행순서도 정해주어야 한다. (실행순서의 동기화!)

```
public class NewsPaper {
     String todayNews;

     public void setTodayNews(String news){ todayNews = news; }

     public String getTodayNews(){ return todayNews; }
}

public class NewsWriter extends Thread {
     NewsPaper paper;

     public NewsWriter(NewsPaper paper) { this.paper = paper; }

     @Override
     public void run(){
          paper.setTodayNews("자바의 열기가 뜨겁습니다!");
     }
}

public class NewsReader extends Thread {
     NewsPaper paper;

     public NewsReader(NewsPaper paper) { this.paper = paper; }

     @Override
     public void run(){
          System.out.println("오늘의 뉴스 : "+paper.getTodayNews());
     }
}

public class SyncThread {
     public static void main(String[] args) {
          NewsPaper paper = new NewsPaper();
          NewsReader reader = new NewsReader(paper);
          NewsWriter writer = new NewsWriter(paper);

          // 문장의 선언 순서만으로 쓰레드에 실행순서를 보장받을 수 없다! (여러 환경적 요인이 있으므로)
          reader.start();
          writer.start();

          try {
               writer.join();
               reader.join();
          }catch(Exception e){ e.printStackTrace(); }
     }
}
```

---

wait( ), notify( ), notifyAll( ) 메소드에 의한 실행순서 동기화 - 잠들고 깨우기

- **public final void wait() throws InterruptedException**

wait() 메소드를 호출한 쓰레드는 notify() 또는 notifyAll() 메소드가 호출될 때까지 **블로킹 상태로 잠든다. (실행 X)**

- **public final void notify()**
  wait() 함수의 호출을 통해서 블로킹 상태에 잠든 **쓰레드 하나를 깨운다.**

- **Public final void notifyAll()**
  wait() 함수의 호출을 통해서 블로킹 상태에 잠든 **모든 쓰레드를 깨운다.**

- **위의 메소드들은 쓰레드에 안전하지 않으므로 반드시 동기화 블록 처리를 해주어야 한다.**

```
synchronized(this){
    wait();
}
```

Reader 쓰레드가 먼저 실행되는 경우를 생각 해보자

```
public class NewsPaper {
    String todayNews;
    boolean isTodayNews = false; // Writer 쓰레드가 접근했는지 체크

    ③ 뒤늦게 Writer 쓰레드가 NewPapar 인스턴스의 setTodayNews 메소드를 실행한다.
    public void setTodayNews(String news){
        todayNews = news;
        isTodayNews = true;

        ④ Writer 쓰레드가 작업을 처리 후 잠들어 있던 모든 쓰레드를 깨운다 (Reader 쓰레드)
        synchronized (this){
            System.out.println("모두 일어나세요!");
            notifyAll();
        }
    }

    ① Reader 쓰레드가 NewPapar 인스턴스의 getTodayNews 메소드를 실행한다.
    public String getTodayNews(){

        // wait() 진입을 위한 조건 판별
        if(isTodayNews == false){
            try{

                ② isTodayNews가 false 이므로 wait( ) 메소드로 Reader가 블로킹 상태로 잠든다.
                synchronized (this){
                    System.out.println("한숨 자면서 기다리겠습니다!");
                    wait();
                }
            }catch (InterruptedException e) { e.printStackTrace(); }
        }
```

```
            ⑤ Reader 쓰레드가 깨어나면 wait() 메소드 밑의 나머지 문장을 실행하게 된다!
            return todayNews;
        }
}
```