



12-2. 스레드 제어

혼자 공부하는 자바 (신용권 저)

- 시작하기 전에
- 스레드 상태
- 스레드 상태 제어
- 키워드로 끝내는 핵심 포인트
- 확인문제



시작하기 전에

[핵심 키워드] : 스레드 상태, 일시정지, 안전한 종료, 데몬 스레드

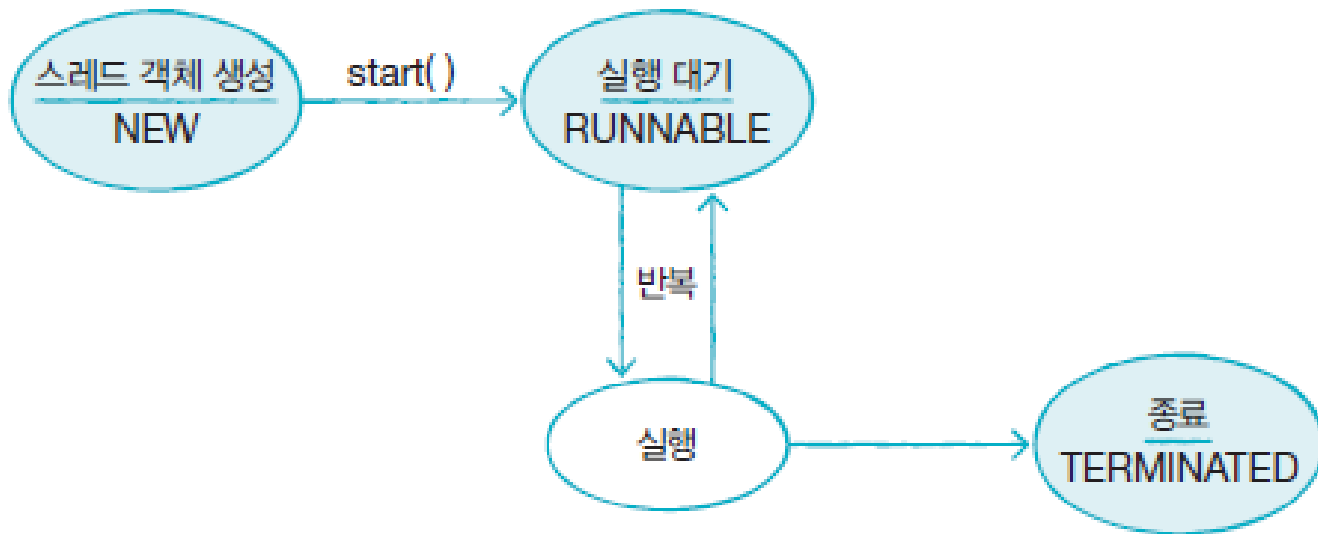
[핵심 포인트]

스레드를 생성하고 시작하면 다양한 상태를 가지게 된다. 이러한 스레드 상태는 자동으로 변경될 수도 있고 코드에 의해 변경될 수도 있다. 스레드의 상태를 변경해 제어하는 방법에 대해 알아본다.



시작하기 전에

- ❖ 스레드 객체 생성하고 start() 메소드를 호출하면 바로 실행되는 것이 아니라 실행 대기 상태가 됨
 - 실행 상태 스레드는 run() 메소드를 모두 실행하기 전 다시 실행 대기 상태로 돌아갈 수 있음
 - 실행 대기 상태에 있는 다른 스레드가 선택되어 실행 상태가 되기도 함
 - 실행 상태에서 run() 메소드의 내용이 모두 실행되면 스레드 실행이 멈추고 종료 상태가 됨



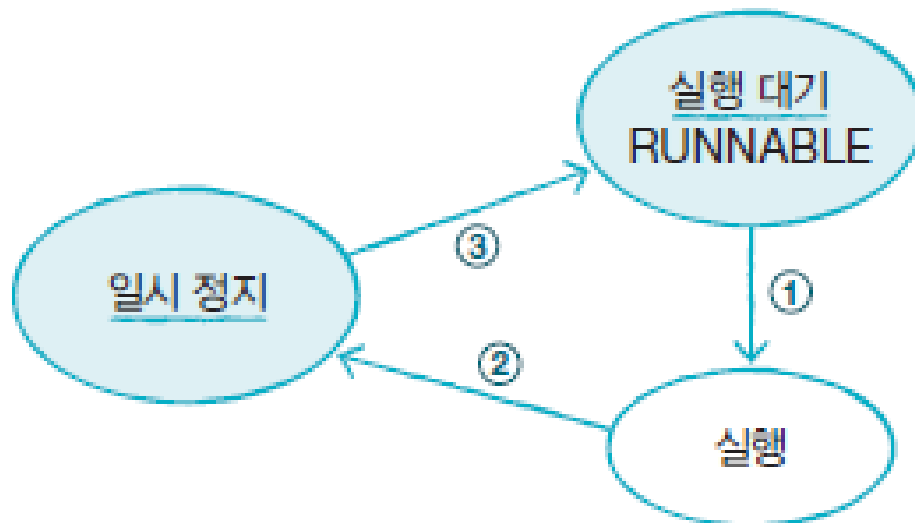
스레드 상태

❖ 실행 (running) 상태

- 실행 대기 상태의 스레드 중에서 운영체제가 하나를 선택하여 CPU가 run() 메소드를 실행하도록 하
- run() 메소드 모두 실행하기 전에 다시 실행 대기 상태로 돌아갈 수 있음

❖ 종료 (terminated) 상태

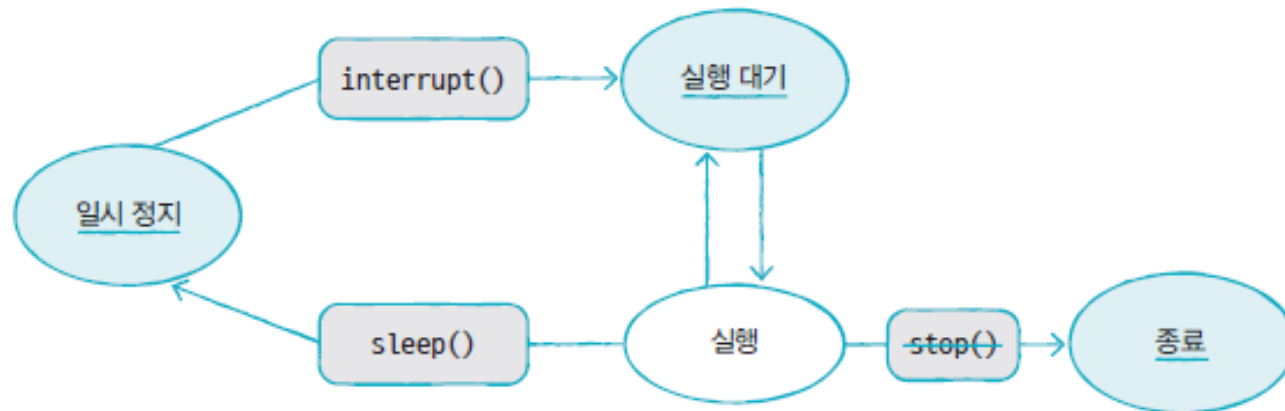
- 실행 상태에서 run() 메소드 종료되면 더 이상 실행할 코드 없기 때문에 스레드 실행이 정지됨



스레드 상태 제어

❖ 스레드 상태 제어

- 실행 중인 스레드의 상태를 변경
- 스레드 상태 변화에 필요한 메소드를 정확히 파악해야



메소드	설명
interrupt()	일시 정지 상태의 스레드에서 InterruptedException을 발생시켜, 예외 처리 코드(catch)에서 실행 대기 상태로 가거나 종료 상태로 갈 수 있도록 합니다.
sleep(long millis)	주어진 시간 동안 스레드를 일시 정지 상태로 만듭니다. 주어진 시간이 지나면 자동적으로 실행 대기 상태가 됩니다.
stop()	스레드를 즉시 종료합니다. 불안정한 종료를 유발하므로 사용하지 않는 것이 좋습니다.



스레드 상태 제어

❖ 주어진 시간 동안 일시 정지

- Thread 클래스의 정적 메소드 sleep() 사용

```
try {  
    Thread.sleep(1000);  
} catch (InterruptedException e) {  
    //interrupt() 메소드가 호출되면 실행  
}
```



스레드 상태 제어

■ 예시 - 3초 주기로 10번 비프음 발생

```
01 package sec02.exam01;
02
03 import java.awt.Toolkit;
04
05 public class SleepExample {
06     public static void main(String[] args) {
07         Toolkit toolkit = Toolkit.getDefaultToolkit();
08         for(int i=0; i<10; i++) {
09             toolkit.beep();
10             try {
11                 Thread.sleep(3000); ← 3초 동안 메인 스레드를
12             } catch (InterruptedException e) { }           일시 정지 상태로 만들음
13
14         }
15     }
16 }
```



스레드 상태 제어

❖ 스레드의 안전한 종료

- 실행 중인 스레드를 즉시 종료해야 하는 경우
- stop 플래그를 이용하는 방법

```
public class XXXThread extends Thread {  
    private boolean stop; //stop 플래그 필드  
  
    public void run() {  
        while( !stop ) { ← stop이 true가 되면 run()이 종료  
            스레드가 반복 실행하는 코드;  
        }  
        //스레드가 사용한 자원 정리  
    }  
}
```



스레드 상태 제어

■ 예시 - 1초 출력 후 스레드를 중지

```
01 package sec02.exam02;
02
03 public class StopFlagExample {
04     public static void main(String[] args) {
05         PrintThread1 printThread = new PrintThread1();
06         printThread.start();
07
08         try { Thread.sleep(1000); } catch (InterruptedException e) {}
09
10         printThread.setStop(true); ← 스레드를 종료하기 위해
11     }                                stop 필드를 true로 변경
12 }
```

실행결과		×
실행	중	
실행	중	
실행	중	
자원	정리	
실행	종료	



스레드 상태 제어

■ 예시 - 무한 반복해서 출력하는 스레드

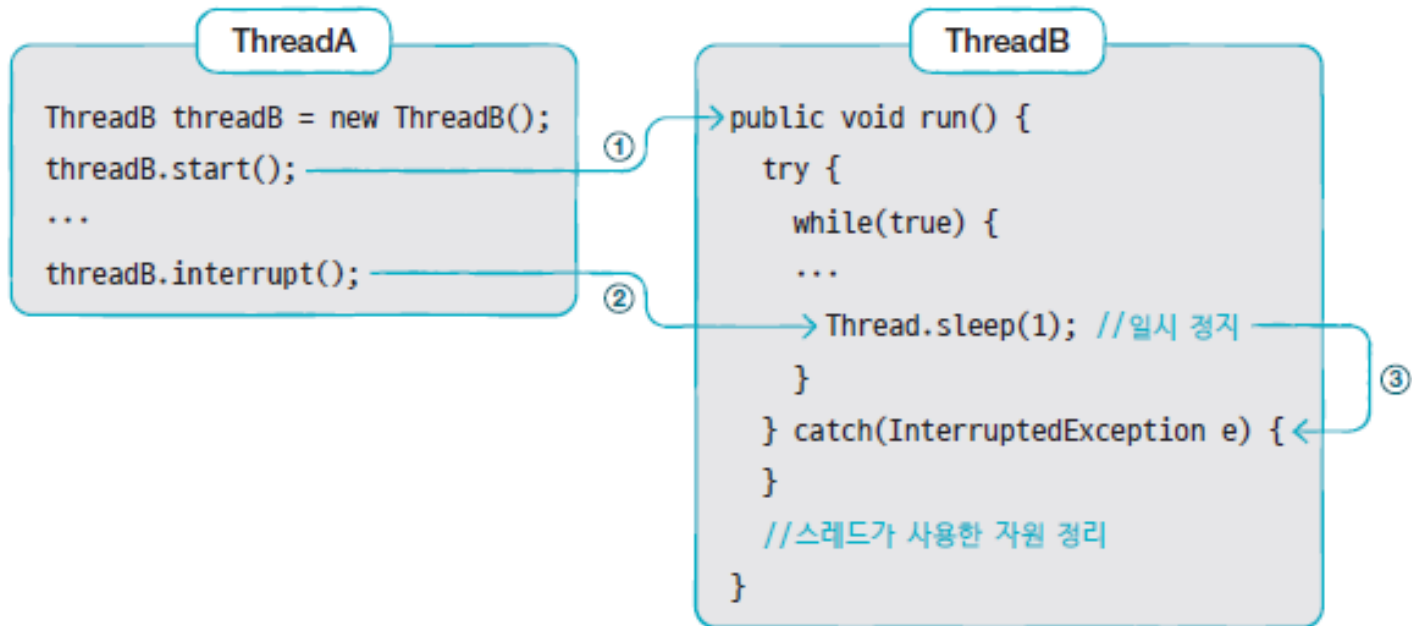
```
01 package sec02.exam02;
02
03 public class PrintThread1 extends Thread {
04     private boolean stop;
05
06     public void setStop(boolean stop) {
07         this.stop = stop;
08     }
09
10     public void run() {
11         while(!stop) {
12             System.out.println("실행 중");
13         }
14         System.out.println("자원 정리");
15         System.out.println("실행 종료");
16     }
17 }
```

stop이 true가 될 때

자
부하
는
바

스레드 상태 제어

■ interrupt() 메소드 이용하는 방법



ThreadA가 ThreadB의 interrupt() 메소드 실행하면 ThreadB가 sleep() 메소드로 일시정지 상태 될 때 ThreadB에서 InterruptedException 발생, 예외 처리 블록으로 이동



스레드 상태 제어

■ 예시 - 1초 후 스레드를 중지

```
01 package sec02.exam03;
02
03 public class InterruptExample {
04     public static void main(String[] args) {
05         Thread thread = new PrintThread2();
06         thread.start();
07
08         try { Thread.sleep(1000); catch (InterruptedException e) {}
09
10         thread.interrupt(); ← 스레드를 종료하기 위해
                               InterruptedException을 발생시킴
11     }
12 }
```

실행결과		×
실행	중	
실행	중	
실행	중	
자원	정리	
실행	종료	



스레드 상태 제어

■ 예시 - 무한 반복해서 출력하는 스레드

```
01 package sec02.exam03;
02
03 public class PrintThread2 extends Thread {
04     public void run() {
05         try {
06             while(true) {
07                 System.out.println("실행 중");
08                 Thread.sleep(1);
09             }
10         } catch (InterruptedException e) {}
11
12         System.out.println("자원 정리");
13         System.out.println("실행 종료");
14     }
15 }
```

InterruptedException 발생



스레드 상태 제어

- 스레드가 실행 대기 혹은 실행 상태에 있을 때 interrupt() 메소드 실행되면 즉시 InterruptedException 발생하는 것이 아니라, 스레드가 미래에 일시정지 상태가 되면 InterruptedException 발생하는 것
- 일시정지 만들지 않는 경우

interrupted()와 isInterrupted() 메소드는 interrupt() 메소드가 호출된 경우 true 리턴

```
boolean status = Thread.interrupted();  
boolean status = objThread.isInterrupted();
```



스레드 상태 제어

■ 예시 - 무한 반복해서 출력하는 스레드

```
01 package sec02.exam04;
02
03 public class PrintThread2 extends Thread {
04     public void run() {
05         while(true) {
06             System.out.println("실행 중");
07             if(Thread.interrupted()) {
08                 break;
09             }
10         }
11         ← while문을 빠져나옴
12         System.out.println("자원 정리");
13         System.out.println("실행 종료");
14     }
15 }
```



키워드로 끝내는 핵심 포인트

- **스레드 상태** : 스레드를 생성하고 시작하면 스레드는 다양한 상태를 가지게 되며, 이는 자동으로 혹은 코드에 의해 변경될 수 있다
- **일시 정지** : 실행 중인 스레드를 일정 시간 멈추게 하려는 경우 Thread 클래스의 정적 메소드인 sleep()을 사용. Thread.sleep() 메소드를 호출한 스레드는 주어진 시간 동안 일시정지 상태가 되고 다시 실행 대기 상태로 돌아감
- **안전한 종료** : 스레드를 안전하게 종료하기 위해 stop 플래그나 interrupt() 메소드를 이용할 수 있다.
- **데몬 스레드** : 주 스레드의 작업을 돕는 보조적 역할 하는 스레드. 주 스레드가 종료되면 자동 종료된다.



- ❖ 스레드 상태 제어 메소드에 대한 설명 중 틀린 것은 무엇입니까?
 - start() 메소드는 실행 대기 상태로 만들어준다
 - sleep() 메소드는 일시 정지 상태로 만들어준다
 - interrupt() 메소드는 실행 상태를 간섭해서 일시 정지 상태로 만들어준다
 - 일시 정지 상태에서 실행 상태로 변경하는 메소드는 없다
- ❖ 메인 스레드에서 1초 후 MovieThread의 interrupt() 메소드를 호출해서 MovieThread를 안전하게 종료하고 싶습니다. 빈칸에 알맞은 코드를 작성해보세요

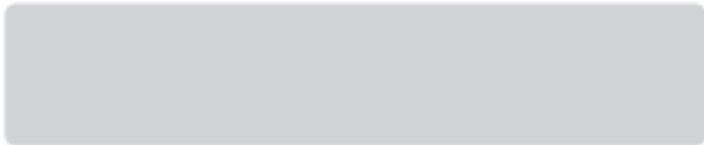
소스 코드 ThreadExample.java

```
01 package sec02.verify.exam03;
02
03 public class ThreadExample {
04     public static void main(String[] args) {
05         Thread thread = new MovieThread();
06         thread.start();
07
08         try { Thread.sleep(1000); } catch (InterruptedException e) {}
09
10         thread.interrupt();
11     }
12 }
```

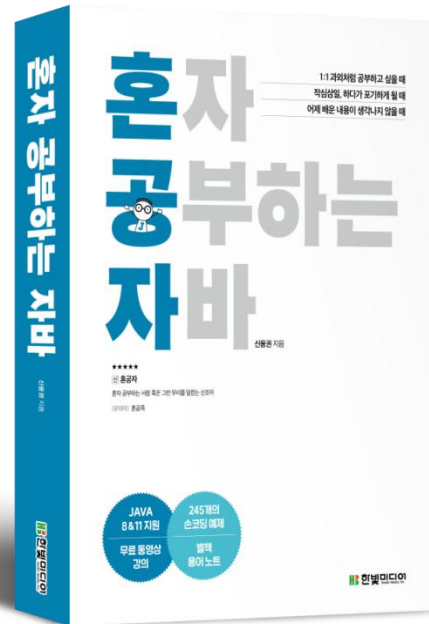


확인문제

소스 코드 MovieThread.java

```
01 package sec02.verify.exam03;
02
03 public class MovieThread extends Thread {
04     @Override
05     public void run() {
06         while(true) {
07             System.out.println("동영상을 재생합니다.");
08
09             
10
11         }
12     }
13 }
```





Thank You!