(INTERMEDIATE) JAVA PROGRAMMING

22. Thread and Multi-threading

Chapter 13

JAVA:
THREAD

스레드란?

- **멀티 태스킹(muli-tasking)**는 여러 개의 애플리케이션을 동시에 실행하여서 컴퓨터 시스템의 성능을 높이기 위한 기법

그림23-1. 병렬 처리의 예

스레드란?

- 다중 스레딩(multi-threading)은 하나의 프로그램이 동시에 여러 가지 작업을 할 수 있도록 하는 것
- 각각의 작업은 스레드(thread)라고 불린다.

그림23-2. 다중 스레드의 개념

프로세스와 스레드

- 프로세스(process): 자신만의 데이터를 가진다.
- 스레드(thread): 동일한 데이터를 공유한다.

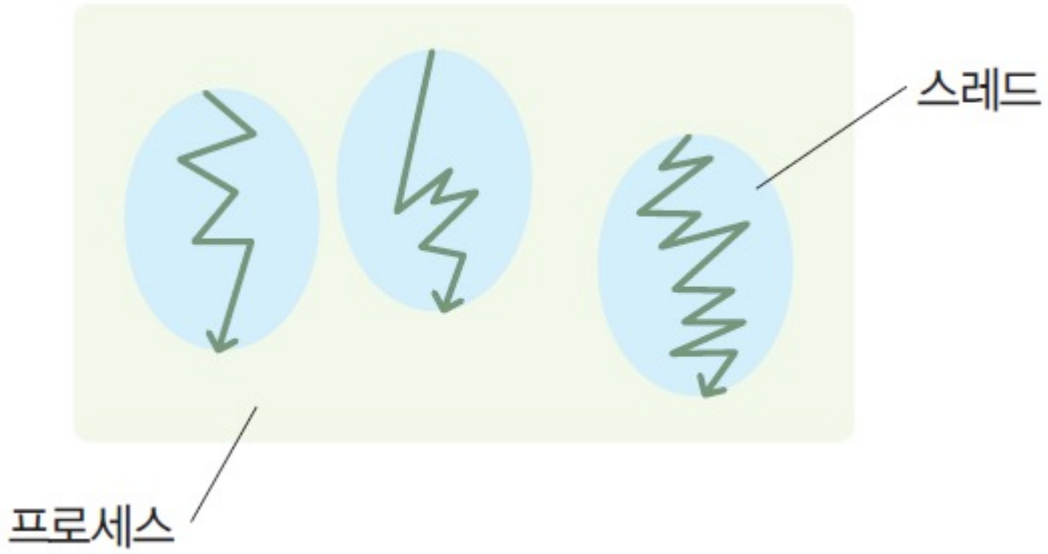


그림23-3. 스레드는 하나의 프로세스 안에 존재한다.

스레드를 사용하는 이유

- 웹 브라우저에서 웹 페이지를 보면서 동시에 파일을 다운로드할 수 있도록 한다.
- 워드 프로세서에서 문서를 편집하면서 동시에 인쇄한다.
- 게임 프로그램에서는 응답성을 높이기 위하여 많은 스레드를 사용한다.
- GUI에서는 마우스와 키보드 입력을 다른 스레드를 생성하여 처리한다.

중간 점검 문제

중간점검

1. 스레드와 프로세스의 결정적인 차이점은 무엇인가?
2. 스레드를 사용해야만 하는 프로그램을 생각하여 보자.
3. 멀티 스레딩에서 발생할 수 있는 문제에는 어떤 것들이 있을까? 추측하여 보라.

스레드 생성과 실행

- 스레드는 Thread 클래스가 담당한다.

```
Thread t = new Thread();     // 스레드 객체를 생성한다.
t.start();                   // 스레드를 시작한다.
```

- 스레드의 작업은 Thread 클래스의 run() 메소드 안에 기술한다.

스레드 생성과 실행

스레드 생성 방법

- Thread 클래스를 상속하는 방법
  - Thread 클래스를 상속받은 후에 run() 메소드를 재정의한다.
- Runnable 인터페이스를 구현하는 방법
  - run() 메소드를 가지고 있는 클래스를 작성하고 ,이 클래스의 객체를 Thread 클래스의 생성자를 호출할 때 전달한다.

Thread 클래스

메소드	설명
Thread()	매개 변수가 없는 기본 생성자
Thread(String name)	이름이 name인 Thread 객체를 생성한다.
Thread(Runnable target, String name)	Runnable을 구현하는 객체로부터 스레드를 생성한다.
static int activeCount()	현재 활동 중인 스레드의 개수를 반환한다.
String getName()	스레드의 이름을 반환
int getPriority()	스레드의 우선순위를 반환
void interrupt()	현재의 스레드를 중단한다.
boolean isInterrupted()	현재의 스레드가 중단될 수 있는지를 검사
void setPriority(int priority)	스레드의 우선순위를 지정한다.
void setName(String name)	스레드의 이름을 지정한다.
static void sleep(int milliseconds)	현재의 스레드를 지정된 시간만큼 재운다.
void run()	스레드가 시작될 때 이 메소드가 호출된다. 스레드가 하여야하는 작업을 이 메소드 안에 위치시킨다.
void start()	스레드를 시작한다.
static void yield()	현재 스레드를 다른 스레드에 양보하게 만든다.

Thread 클래스를 상속하기

- Thread를 상속받아서 클래스를 작성한다.
- run() 메소드를 재정의한다.

```
class MyThread extends Thread {
    public void run() {
        ...                  <---------- 여기에 수행하여야 하는 작업을 적어준다.
    }
}
```

- Thread 객체를 생성한다.

```
Thread t = new MyThread();
```

- start()를 호출하여서 스레드를 시작한다.

```
t.start();
```

Thread 클래스를 상속하기

MyThreadTest.java

```
01  class MyThread extends Thread {
02      public void run() {
03          for (int i = 10; i >= 0; i--)
04              System.out.print(i + " ");
05      }
06  }
07
08  public class MyThreadTest {
09      public static void main(String args[]) {
10          Thread t = new MyThread();
11          t.start();
12      }
13  }
```

MyThread 클래스는 Thread를 상속받는다. Thread 클래스는 java.lang 패키지에 들어 있어서 따로 import할 필요가 없다. MyThread 클래스는 하나의 메소드 run()만을 가지고 있는데 run()은 이 스레드가 시작되면 자바 런타임 시스템에 의하여 호출된다. 스레드가 실행하는 모든 작업은 이 run() 메소드 안에 있어야 한다. 현재는 단순히 10부터 0까지를 화면에 출력한다.

스레드를 실행시키려면 Thread에서 파생된 클래스 MyThread의 인스턴스를 생성한 후 start()를 호출한다. Thread 타입의 변수 t가 선언되고 MyThread의 객체가 생성하였다. 객체가 생성되었다고 스레드가 바로 시작되는 것은 아니다. start() 메소드를 호출해야만 스레드가 실행된다.

실행결과

```
10 9 8 7 6 5 4 3 2 1 0
```

Runnable 인터페이스를 구현하는 방법

- Runnable 인터페이스를 구현한 클래스를 작성한다.
- run() 메소드를 재정의한다.

```
class MyRunnable implements Runnable {
    public void run() {
        ...
    }
}
```

- Thread 객체를 생성하고 이때 MyRunnable 객체를 인수로 전달한다.

```
Thread t = new Thread(new MyRunnable());
```

- start()를 호출하여서 스레드를 시작한다.

```
t.start();
```

Runnable 인터페이스를 구현하는 방법

MyRunnableTest.java

```
01  class MyRunnable implements Runnable {
02      public void run() {
03          for (int i = 10; i >= 0; i--)
04              System.out.print(i + " ");
05      }
06  }
07
08  public class MyRunnableTest {
09      public static void main(String args[]) {
10          Thread t = new Thread(new MyRunnable());
11          t.start();
12      }
13  }
```

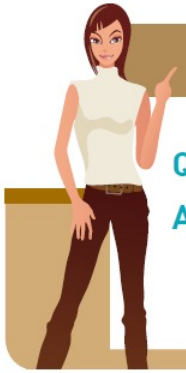
Runnable을 구현하는 클래스를 작성한다. run() 메소드를 재정의하여 작업에 필요한 코드를 넣는다.

Thread 클래스의 인스턴스를 생성하고, Runnable 객체를 Thread 생성자의 매개변수로 넘긴다. Thread 객체의 start() 메소드를 호출하여야 한다.

실행결과

```
10 9 8 7 6 5 4 3 2 1 0
```

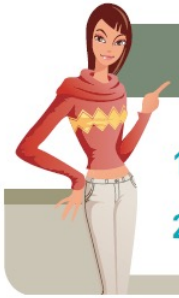
중간 점검



Q&A

Q: 그렇다면 스레드를 생성하기 위해서는 어떤 방법을 사용하는 것이 좋은가?

A: Runnable 인터페이스를 사용하는 편이 더 일반적이다. Runnable 객체는 Thread가 아닌 다른 클래스를 상속받을 수 있다. Thread 클래스에서 상속받으면 다른 클래스를 상속받을 수 없다. 따라서 인터페이스 방법은 유연할 뿐아니라 고수준의 스레드 관리 API도 사용할 수 있는 장점이 있다.



중간점검

1. 스레딩을 담당하는 클래스의 이름은?
2. Thread를 상속받는 방법의 문제점은 무엇인가?

예제: sleep()

SleepTest.java

```
01  public class SleepTest {
02     public static void main(String args[]) throws InterruptedException {
03        String messages[] = { "Pride will have a fall.",
04        "Power is dangerous unless you have humility.",
05        "Office changes manners.",
06        "Empty vessels make the most sound." };
07
08        for (int i = 0; i < messages.length; i++) {
09           Thread.sleep(1000);
10           System.out.println(messages[i]);
11        }
12     }
13  }
```

sleep()가 다른 메소드에 의하여 중단되면 발생하는 예외, 여기서 처리하지 않고 상위 메소드로 전달한다. 사실 여기서는 다른 메소드가 sleep()을 방해할 일이 없다.

1000밀리 초 동안 실행을 중지한다.

실행결과

```
Pride will have a fall.
Power is dangerous unless you have humility.
Office changes manners.
Empty vessels make the most sound.
```

인터럽트

- 인터럽트(interrupt)는 하나의 스레드가 실행하고 있는 작업을 중지하도록 하는 메카니즘이다.

```
for (int i = 0; i < messages.length; i++) {
    try {
        Thread.sleep(1000);
    } catch (InterruptedException e) {
        // 인터럽트를 받은 것이다. 단순히 리턴한다.
        return;
    }
    System.out.println(messages[i]);
}
```

<---------- 인터럽트 처리는 여기에서 해준다.

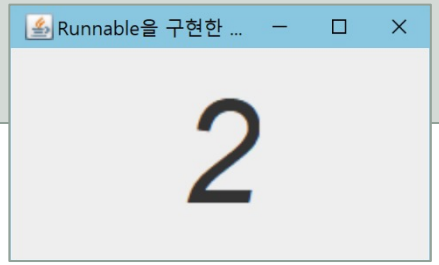
- 그런데 만약 스레드가 실행 중에 한번도 sleep()을 호출하지 않는다면 InterruptedException를 받지못한다.

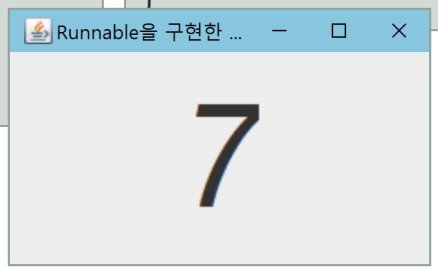
```
if (Thread.interrupted()) {
    // 인터럽트를 받은 것이다. 단순히 리턴한다.
    return;
}
```

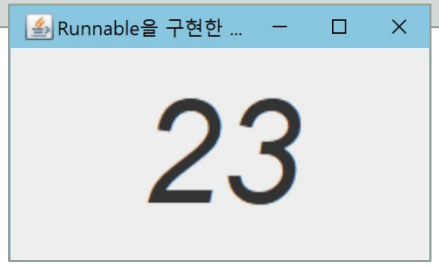
예제 13-2 : Runnable 인터페이스를 이용하여 1초 단위로 출력하는 타이머 스레드 만들기

```
import java.awt.*;
import javax.swing.*;

class TimerRunnable implements Runnable {
    private JLabel timerLabel;

    public TimerRunnable(JLabel timerLabel) {
        this.timerLabel = timerLabel;
    }
    @Override
    public void run() {
        int n=0;
        while(true) {
            timerLabel.setText(Integer.toString(n));
            n++;
            try {
                Thread.sleep(1000);
            }
            catch(InterruptedException e) {
                return;
            }
        }
    }
}
```

```
public class RunnableTimerEx extends JFrame {
    public RunnableTimerEx() {
        setTitle("Runnable을 구현한 타이머 스레드 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container c = getContentPane();
        c.setLayout(new FlowLayout());
        JLabel timerLabel = new JLabel();
        timerLabel.setFont(new Font("Gothic", Font.ITALIC, 80));
        c.add(timerLabel);

        TimerRunnable runnable = new TimerRunnable(timerLabel);
        Thread th = new Thread(runnable);

        setSize(250,150);
        setVisible(true);

        th.start();
    }
    public static void main(String[] args) {
        new RunnableTimerEx();
    }
}
```

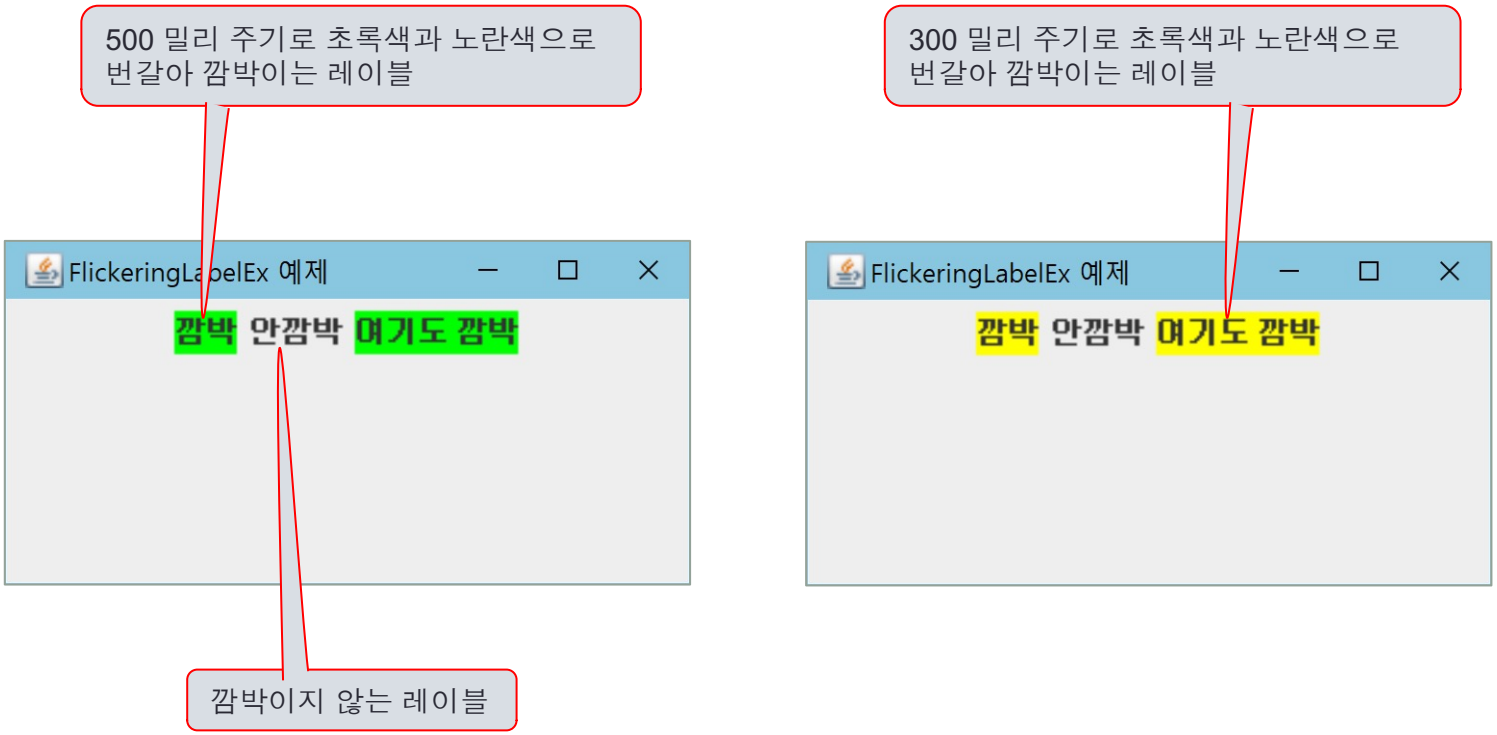






예제 13-3 : 깜박이는 문자열을 가진 레이블 만들기

JLabel을 상속받아 문자열을 깜박이는 FlickeringLabel 컴포넌트를 작성하라.



예제 13-3 : 정답 코드

```
import java.awt.*;
import javax.swing.*;

class FlickeringLabel extends JLabel implements Runnable {
    private long delay;
    public FlickeringLabel(String text, long delay) {
        super(text);
        this.delay = delay;
        setOpaque(true);
        Thread th = new Thread(this);
        th.start();
    }
    @Override
    public void run() {
        int n=0;
        while(true) {
            if(n == 0)
                setBackground(Color.YELLOW);
            else
                setBackground(Color.GREEN);
            if(n == 0) n = 1;
            else n = 0;
            try {
                Thread.sleep(delay);
            }
            catch(InterruptedException e) {
                return;
            }
        }
    }
}
```

```
public class FlickeringLabelEx extends JFrame {
    public FlickeringLabelEx() {
        setTitle("FlickeringLabelEx 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container c = getContentPane();
        c.setLayout(new FlowLayout());

        // 깜박이는 레이블 생성
        FlickeringLabel fLabel = new FlickeringLabel("깜박“,500);

        // 깜박이지 않는 레이블 생성
        JLabel label = new JLabel("안깜박");

        // 깜박이는 레이블 생성
        FlickeringLabel fLabel2 = new FlickeringLabel("여기도 깜박“,300);

        c.add(fLabel);
        c.add(label);
        c.add(fLabel2);

        setSize(300,150);
        setVisible(true);
    }

    public static void main(String[] args) {
        new FlickeringLabelEx();
    }
}
```

스레드 만들 때 주의 사항

- run() 메소드가 종료하면 스레드는 종료한다.
  - 스레드가 계속 살아있게 하려면 run() 메소드 내 무한루프 작성
- 한번 종료한 스레드는 다시 시작시킬 수 없다.
  - 다시 스레드 객체를 생성하고 start()를 호출해야 함
- 한 스레드에서 다른 스레드를 강제 종료할 수 있다.
  - 뒤에서 다룸

스레드의 상태

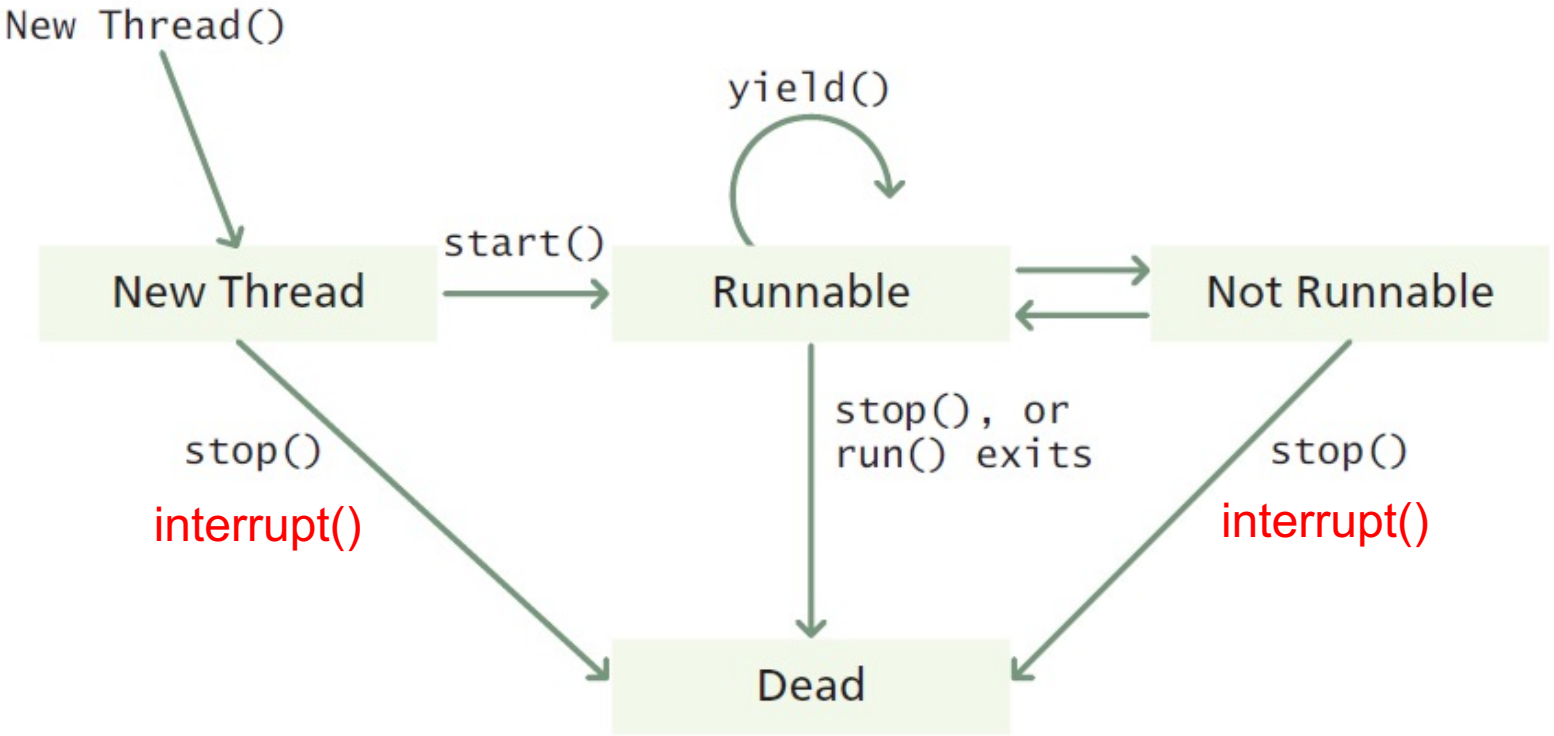


그림23-4. 스레드의 상태

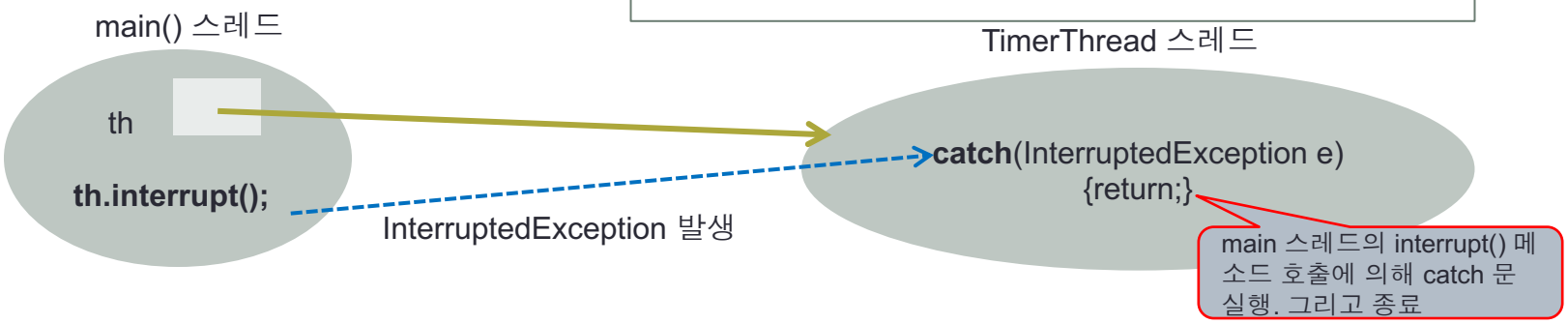
스레드 종료와 타 스레드 강제 종료

- 스스로 종료 : run() 메소드 리턴
- 타 스레드에서 강제 종료 : interrupt() 메소드 사용

```
class TimerThread extends Thread {
    private int n = 0;
    @Override
    public void run() {
        while(true) {
            System.out.println(n); // 화면에 카운트 값 출력
            n++;
            try {
                sleep(1000);
            }
            catch(InterruptedException e){
                return; // 예외를 받고 스스로 리턴하여 종료
            }
        }
    }
}
```

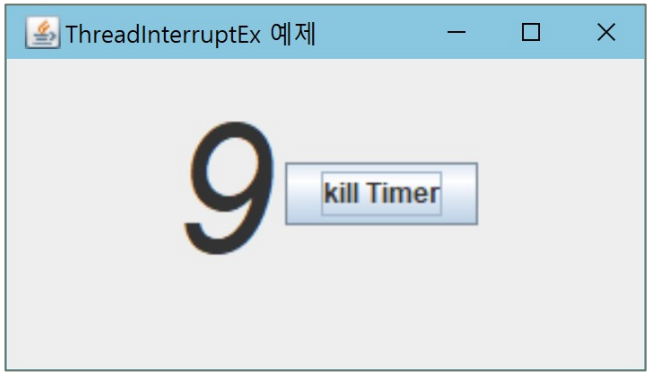
만일 return 하지 않으면 스레드는 종료하지 않음

```
public static void main(String [] args) {
    TimerThread th = new  TimerThread();
    th.start();

    th.interrupt(); // TimerThread 강제 종료
}
```



예제 13-5 : 타이머 스레드 강제 종료

아래 그림과 같이 작동하여 타이머 스레드를 강제 종료시키는 스윙 응용프로그램을 작성해보자.



타이머는 정상 작동한다.

Kill Timer 버튼을 클릭하면
타이머가 멈춘다.
버튼은 비활성화된다.

조인

- join() 메소드는 하나의 스레드가 다른 스레드의 종료를 기다리게 하는 메소드이다.

```
t.join();
```

예제

ThreadControl.java

```
01  public class ThreadControl {
02
03      static void print(String message) {
04          String threadName = Thread.currentThread().getName();
05          System.out.format("%s: %s%n", threadName, message);
06      }
07
08      private static class MessageLoop implements Runnable {
09          public void run() {
10              String messages[] = { "Pride will have a fall.",
11                    "Power is dangerous unless you have humility.",
12                    "Office changes manners.",
13                    "Empty vessels make the most sound." };
14
15              try {
16                  for (int i = 0; i < messages.length; i++) {
```

메시지를 스레드 이름과 함께 출력한다.

예제

```
17                print(messages[i]);
18                Thread.sleep(2000);
19             }
20          } catch (InterruptedException e) {
21             print("아직 끝나지 않았어요!");
22          }
23       }
24    }
25
26    public static void main(String args[]) throws InterruptedException {
27       int tries = 0;
28
29       print("추가적인 스레드를 시작합니다.");
30       Thread t = new Thread(new MessageLoop());
31       t.start();
32
33       print("추가적인 스레드가 끝나기를 기다립니다.");
34       while (t.isAlive()) {
35          print("아직 기다립니다.");
36          t.join(1000);
37          tries++;
```

인터럽트되면 메시지를 출력한다.

스레드 t가 종료하기를 1초 동안 기다린다.

실행결과

```
38              if (tries > 2) {
39                  print("참을 수 없네요!");
40                  t.interrupt();
41                  t.join();
42              }
43          }
44          print("메인 스레드 종료!");
45      }
46  }
```

스레드 t를 강제로 중단시킨다.

스레드 t가 종료하기를 기다린다.

실행결과

```
main: 추가적인 스레드를 시작합니다.
main: 추가적인 스레드가 끝나기를 기다립니다.
main: 아직 기다립니다.
Thread-0: Pride will have a fall.
main: 아직 기다립니다.
main: 아직 기다립니다.
Thread-0: Power is dangerous unless you have humility.
main: 참을 수 없네요!
Thread-0: 아직 끝나지 않았어요!
main: 메인 스레드 종료!
```

중간 점검

중간점검

1. setPrioroty()와 getPriority()의 역할은?
2. sleep() 메소드는 어떤 경우에 사용되는가?
3. 어떤 스레드가 가장 우선적으로 실행되는가?
4. Thread의 run() 메소드의 역할은?
5. Thread의 start(), stop() 메소드의 역할은?
6. 어떤 일이 발생하면 스레드가 실행 중지 상태로 가는가?