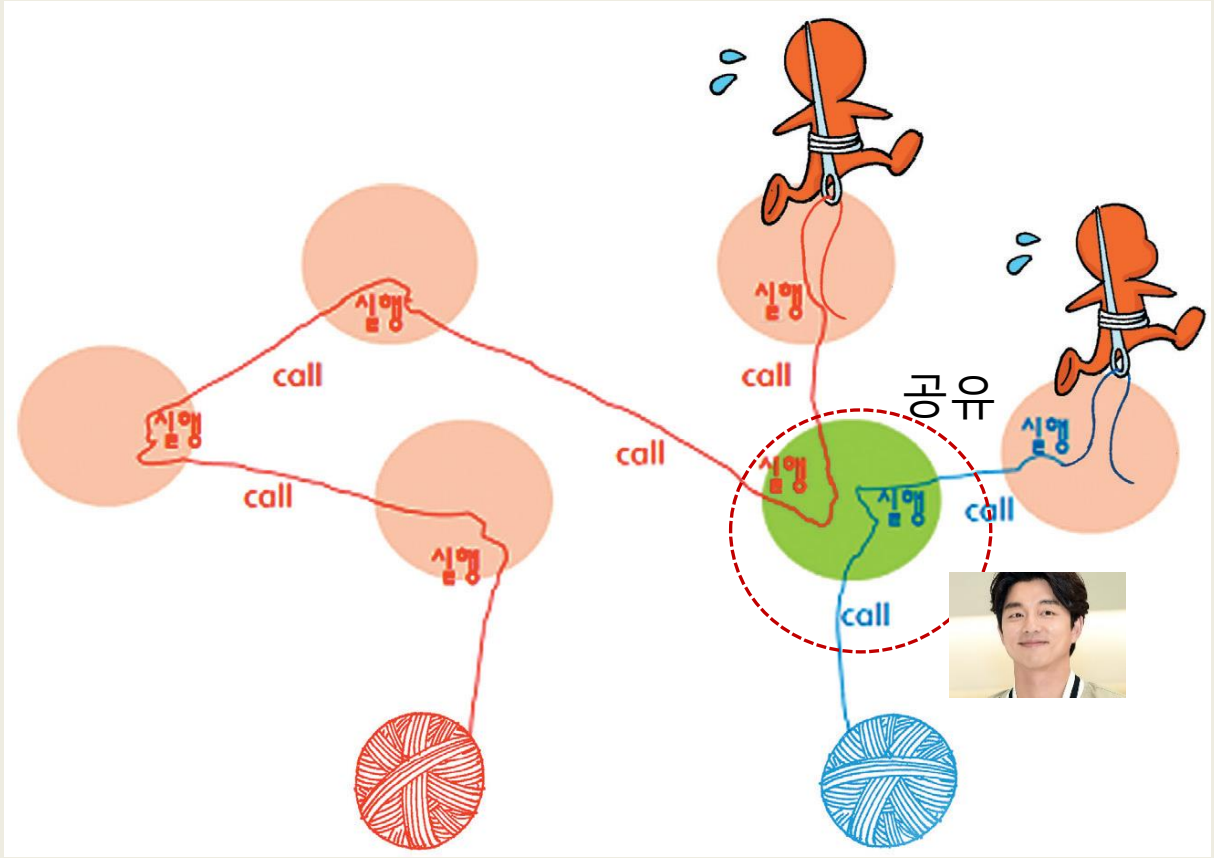
제 14 장
스레드와 멀티태스킹
Part-1 스레드 생성

스레드(Thread) 개념과 실(thread)



- 마치 바늘이 하나의 실(thread)을 가지고 바느질하는 것과 자바의 스레드는 일맥 상통함
- 이러한 실이 모여지면 하나의 의류가 탄생
  – 장갑, 셔츠, 모자 등

스레드 A 생성

스레드 A

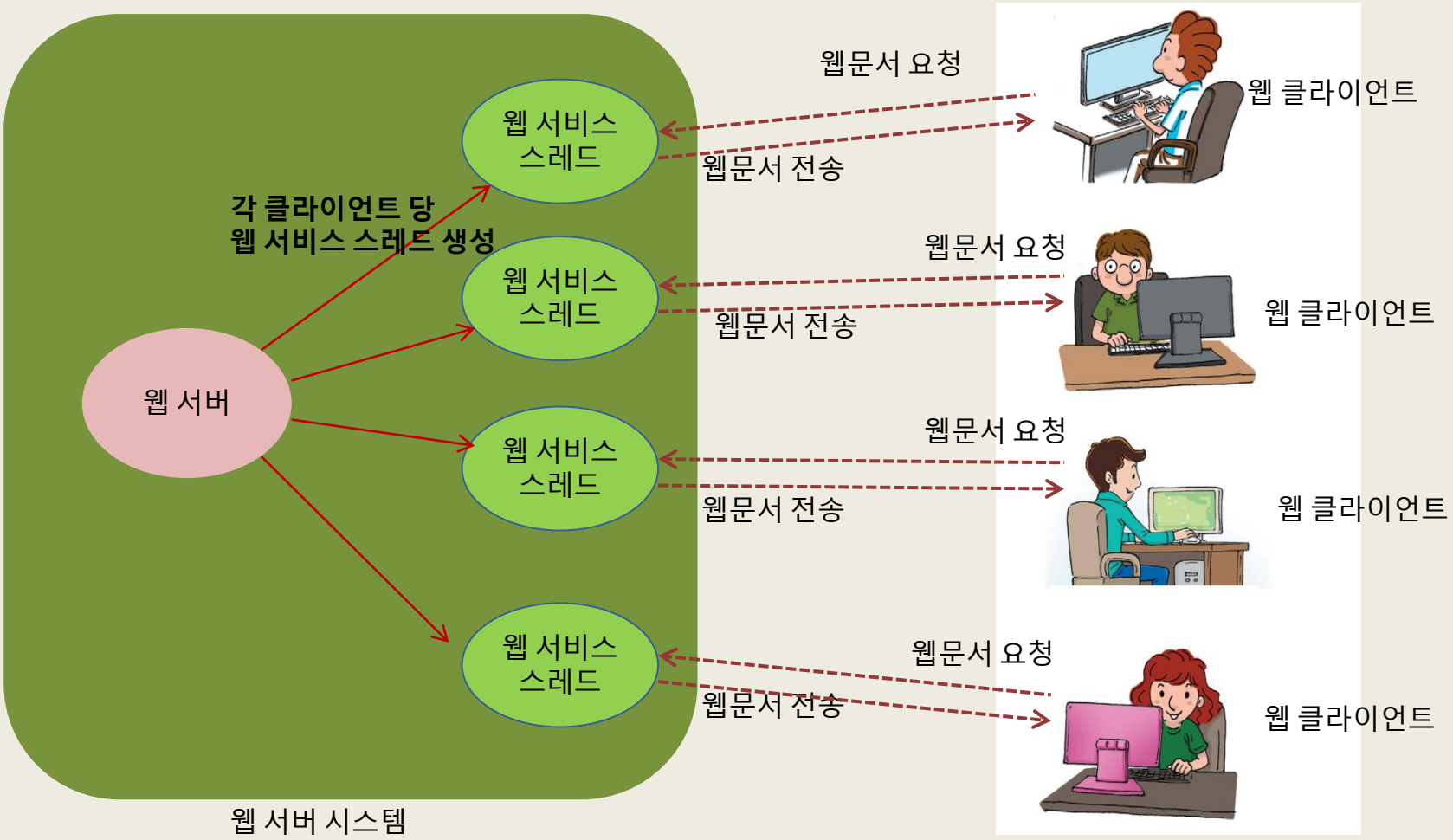
스레드 B 생성

스레드 B

스레드와 멀티스레딩

- 스레드
  - 프로그램 코드를 이동하면서 실행하는 하나의 경량급 프로세스(light-weight process)
- 자바의 멀티태스킹
  - 멀티스레딩으로 이루어진다
    - 자바에 프로세스 개념은 존재하지 않고, 스레드 개념만 존재
    - 스레드가 **기본 실행 단위이자 스케쥴링 단위**
  - 하나의 응용프로그램은 여러 개의 스레드로 구성이 가능
    - 멀티 스레딩
    - 스레드 사이의 통신에 따른 오버헤드가 크지 않음

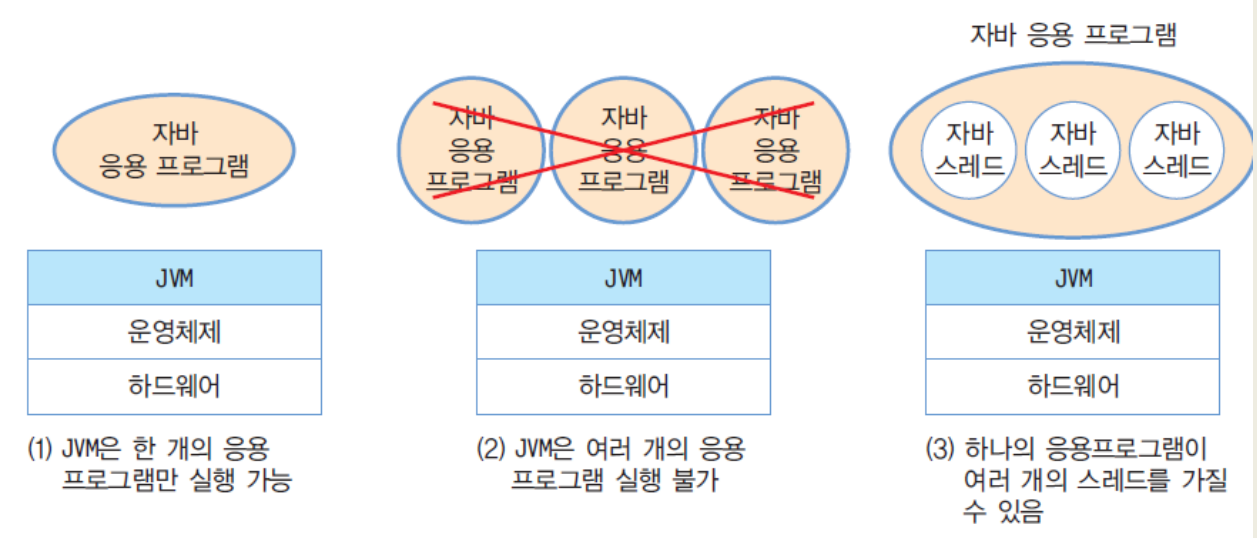
웹 서버의 멀티스레딩 사례



자바 스레드

- 자바 스레드
  - 자바 가상 기계(JVM)에 의해 스케쥴되는 기본 실행 단위의 코드
  - 스레드의 생명 주기(life cycle)는 JVM에 의해 관리됨
    - JVM은 스레드 단위로 스케쥴링
- JVM과 멀티스레드의 관계
  - 하나의 JVM은 하나의 자바 응용프로그램만 실행, 즉 하나의 프로세스만 존재
    - 자바 응용프로그램이 시작될 때 JVM이 함께 실행됨
    - 자바 응용프로그램이 종료하면 JVM도 함께 종료함
  - 하나의 응용프로그램은 하나 이상의 스레드로 구성 가능

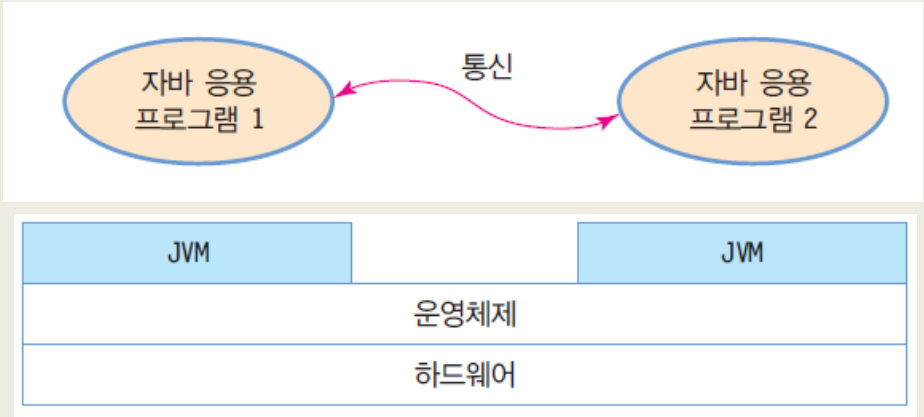
JVM과 자바 응용프로그램, 스레드의 관계



(1) JVM은 한 개의 응용 프로그램만 실행 가능

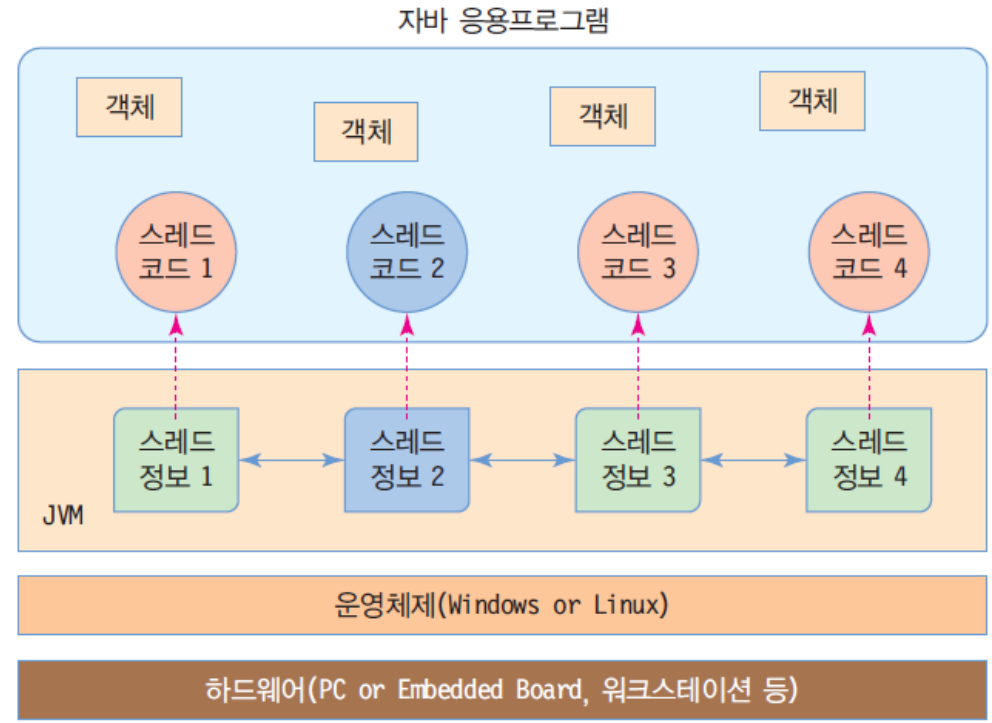
(2) JVM은 여러 개의 응용 프로그램 실행 불가

(3) 하나의 응용프로그램이 여러 개의 스레드를 가질 수 있음



두 개의 자바 응용프로그램이 동시에 실행시키고자 하면
두 개의 JVM을 이용하고 응용프로그램은 서로 소켓 등을 이용하여 통신

자바 스레드와 JVM



현재 하나의 JVM에 의해 4 개의 스레드가 실행 중이며
그 중 스레드 2가 JVM에 의해 스케쥴링되어 실행되고 있음

- 각 스레드의 코드는 응용프로그램 내에 존재

JVM이 스레드를 관리
- 스레드가 몇 개인지?
- 스레드 코드의 위치가 어디인지?
- 스레드의 우선순의는 얼마인지? 등

스레드 생성

- 스레드 실행을 위해 개발자가 하는 작업
  - 스레드 코드 작성
  - JVM에게 스레드를 생성하고 스레드 코드를 실행하도록 요청

- **자바에서 스레드 만드는 2 가지 방법**
  - java.lang.Thread 클래스를 상속받아 스레드 코드 작성하는 경우
    - Thread 클래스의 run() 메소드를 오버라이딩한다
  - java.lang.Runnable 인터페이스를 구현하는 경우
    - 함수인터페이스(하나의 추상메소드 run()만 포함)
    - 추상메소드 run()을 구현

Thread 클래스의 메소드

- 생성자
  - Thread()
  - Thread(Runnable target)
  - Thread(String name)
  - Thread(Runnable target, String name)
- 스레드 스타트
  - void start()
- 스레드 코드 작성
  - void run()
- 스레드 수면
  - static void sleep(long mills)
- 다른 스레드 종료시키기
  - void interrupt()
- 다른 스레드에게 양보
  - static void yield()
  - 현재 스레드의 실행을 중단하고 다른 스레드가 실행될 수 있도록 양보한다.
- 다른 스레드가 종료될 때까지 기다리기
  - void join()
- 현재 스레드 객체 얻기
  - static Thread currentThread()
- 스레드 ID 얻기
  - long getId()
- 스레드 이름 알아내기
  - String getName()
- 스레드 우선순위값 알아내기
  - int getPriority()
- 스레드의 상태 알아내기
  - Thread.State getState()

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 활동중인 스레드의 개수를 반환한다.
String getName()	스레드의 이름을 반환
int getPriority()	스레드의 우선 순위를 반환
void interrupt()	현재의 스레드를 중단한다.
boolean isInterrupted()	현재의 스레드가 중단될 수 있는지를 검사
void setPriority(int *priority*)	스레드의 우선 순위를 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 하여야 하는 작업을 이 메소드 안에 위치시킨다.
void start()	스레드를 시작한다.
static void yield()	현재 스레드를 다른 스레드에 양보하게 만든다.

자바 스레드 생성

- **java.lang.Thread 클래스를 상속받아 스레드 코드 작성**하는 경우
  - Thread 클래스의 run() 메소드를 오버라이딩한다
- **java.lang.Runnable 인터페이스를 구현하는** 경우
  - 함수인터페이스(하나의 추상메소드 run()만 포함)
  - 추상메소드 run()을 구현

Thread 클래스 상속

- 스레드 클래스 작성
  - Thread 클래스 상속. 새 클래스 작성
- 스레드 코드 작성
  - run() 메소드 오버라이딩
    - run() 메소드를 스레드 코드라고 부름
    - run() 메소드에서 스레드 실행 시작
- 스레드 객체 생성
- 스레드 시작
  - start() 메소드 호출
    - 스레드로 작동 시작
    - JVM에 의해 스케쥴 시작

```
class TimerThread extends Thread {
    ..........................................

    public void run() { // run() 오버라이딩
        ..........................
    }
}
```

```
TimerThread  th = new TimerThread();
```

```
th.start();
```

* Thread를 상속받아 1초 단위로 초 시간을 출력하는 TimerThread 스레드 작성

스레드 클래스 정의

스레드 코드 작성

1초에 한 번씩 n을 증가시켜 콘솔에 출력한다.

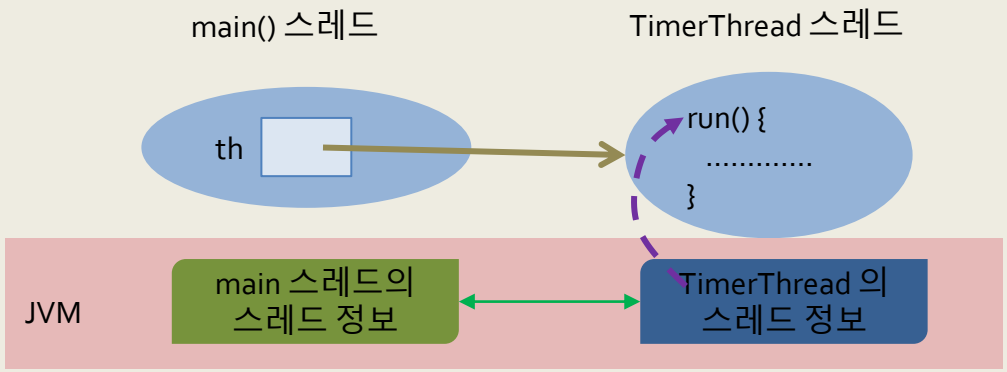
```
class TimerThread extends Thread {
    int n = 0;
    public void run() {
        while(true) { // 무한루프를 실행한다.
            System.out.println(n);
            n++;
            try {
                sleep(1000); //1초 동안 잠을 잔 후 깨어난다.
            }
            catch(InterruptedException e){return;}
        }
    }
}
```

```
0
1
2
3
4
```

스레드 객체 생성

스레드 시작

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



스레드 주의 사항

- run() 메소드가 종료하면 스레드는 종료한다.
  - 스레드가 계속 존재하게 하려면 run() 메소드 내에 무한 루프가 실행되어야 한다.
- 한번 종료한 스레드는 다시 시작시킬 수 없다.
  - 스레드 객체를 생성하여 다시 스레드로 등록하여야 한다.
- 한 스레드에서 다른 스레드를 강제 종료할 수 있다.

Runnable 인터페이스 구현

- 스레드 클래스 작성
  - Runnable 인터페이스 구현하는 새 클래스 작성
- 스레드 코드 작성
  - run() 메소드 구현
    - run() 메소드를 스레드 코드라고 부름
    - run() 메소드에서 스레드 실행 시작
- 스레드 객체 생성
- 스레드 시작
  - start() 메소드 호출

```
class TimerRunnable implements Runnable {
    ..........
    public void run() { // run() 메소드 오버라이딩
        ..........
    }
}
```

```
Thread th = new Thread(new TimerRunnable());
```

```
th.start();
```

*Runnable 인터페이스를 상속받아 1초 단위로 초 시간을 출력하는 스레드 작성

Runnable 을 클래스로 구현

스레드 코드 작성

1초에 한 번씩 n을 증가시켜 콘솔에 출력한다.

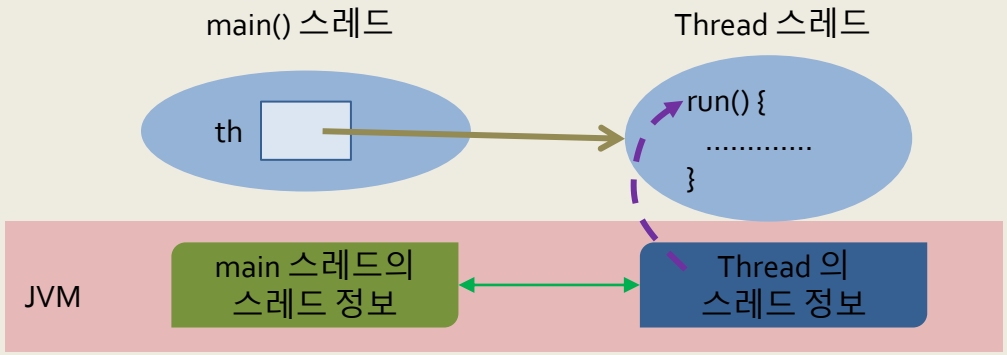
```
class TimerRunnable implements Runnable {
    int n = 0;
    public void run() {
        while(true) { // 무한루프를 실행한다.
            System.out.println(n);
            n++;
            try {
                Thread.sleep(1000); // 1초 동안 잠을 잔 후 깨어난다.
            }
            catch(InterruptedException e) { return; }
        }
    }
}
```

```
0
1
2
3
4
```

스레드 객체 생성

스레드 시작

```
public class TestRunnable {
    public static void main(String [] args) {
        Thread th = new Thread(new TimerRunnable());
        th.start();
    }
}
```



스레드 정보

필드	타입	내용
스레드 이름	스트링	스레드의 이름으로서 사용자가 지정
스레드 ID	정수	스레드 고유의 식별자 번호
스레드의 PC(Program Count)	정수	현재 실행 중인 스레드 코드의 주소
스레드 상태	정수	NEW, RUNNABLE, WAITING, TIMED_WAITING, BLOCK, TERMINATED 등 6개 상태 중 하나
스레드 우선순위	정수	스레드 스케줄링 시 사용되는 우선순위 값으로서 1~10 사이의 값이며 10이 최상위 우선순위
스레드 그룹	정수	여러 개의 자바 스레드가 하나의 그룹을 형성할 수 있으며 이 경우 스레드가 속한 그룹
스레드 레지스터 스택	메모리 블록	스레드가 실행되는 동안 레지스터들의 값

어떤 방법이 더 좋을까?

- 어떤 것이 더 좋을 것인가에 대한 결정은 프로그래머가 결정할 사항이나,
- 자바에서 다중 상속이 불가능한 것을 감안한다면 Runnable 인터페이스를 사용하는 것이 다중 상속의 가능성을 위해서 더 낫다고 판단되며,
  - 스레드를 생성하기 위해서 Thread 클래스를 상속받았으므로 더 이상 상속은 불가능
- Runnable 인터페이스를 사용하면 고급의 스레드 관리 API도 사용할 수 있으며, Android 에서의 스레드 구현 시 효율적일 수 있다

```
public class MyThread extends Thread {
    public void run() {
        for(int i=0; i<10; i++) {
         System.out.println(getName() + ": " + i) ;
        }
    }
    public MyThread(String name) {
        super(name) ;
    }
    public static void main(String[] args) {
        MyThread mt1 = new MyThread("Won Ho Chung");
        MyThread mt2 = new MyThread("Dong Sung Hur");
        MyThread mt3 = new MyThread("Jee In Chung");
        MyThread mt4 = new MyThread("Hay In Chung");
        mt1.start();
        mt2.start();
        mt3.start();
        mt4.start();
    }
}
```

```
Won Ho Chung: 0
Won Ho Chung: 1
Won Ho Chung: 2
Won Ho Chung: 3
Dong Sung Hur: 0
Won Ho Chung: 4
Hay In Chung: 0
Jee In Chung: 0
Jee In Chung: 1
Jee In Chung: 2
Jee In Chung: 3
Jee In Chung: 4
Won Ho Chung: 5
Dong Sung Hur: 1
Won Ho Chung: 6
Jee In Chung: 5
Hay In Chung: 1
Jee In Chung: 6
Jee In Chung: 7
Won Ho Chung: 7
Dong Sung Hur: 2
Dong Sung Hur: 3
Dong Sung Hur: 4
Dong Sung Hur: 5
Dong Sung Hur: 6
Dong Sung Hur: 7
Dong Sung Hur: 8
Dong Sung Hur: 9
Won Ho Chung: 8
Won Ho Chung: 9
Jee In Chung: 8
Hay In Chung: 2
Jee In Chung: 9
Hay In Chung: 3
Hay In Chung: 4
Hay In Chung: 5
Hay In Chung: 6
Hay In Chung: 7
Hay In Chung: 8
Hay In Chung: 9
```