Thread Ⅲ - Characteristic (LifeCycle, Memory)

쓰레드의 라이프 사이클 (생명주기)

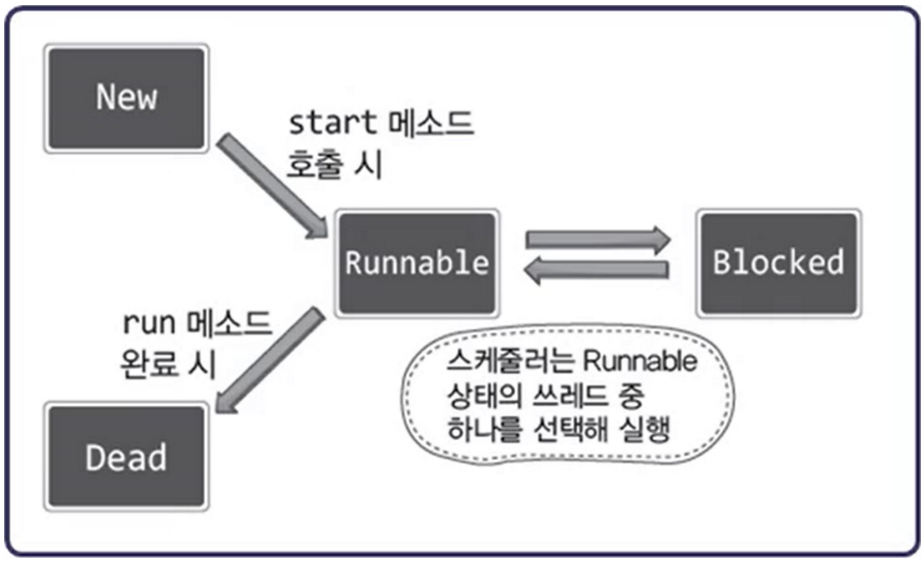
- sleep( ) Method가 갖는 의미를 알아보자

- **New** : Thread의 Instance가 생성이 된 상태의 시점

- **Runnable** : start( ) Method 호출 시, 실행 가능한 준비상태의 시점
  - JVM에는 쓰레드 스케쥴러가 있는데, 우선순위에 따라 스케쥴러가 Runnable 상태의 쓰레드 실행을 결정.

- **Blocked** : sleep( ), join( ) Method의 호출로 인해 Thread가 휴식상태가 되는 시점
  - 쓰레드 스케쥴러는 실행 가능성이 있는 쓰레드에서 제외시킨다. 실행 X
  - 쓰레드들은 Runnable 상태와 Blocked 상태를 빈번하게 바꾼다.
  - 이 때문에 우선순위가 낮은 쓰레드들도 번갈아가며 실행이 가능하게 된다!

- **Dead** : run( ) Method 실행종료 시점

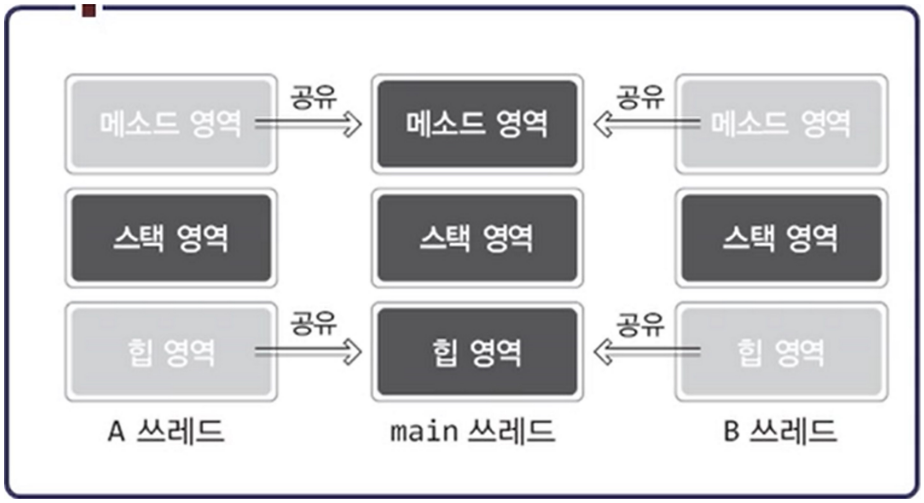


쓰레드의 메모리 구성

- Main Thread의 실행을 위해서는 Stack 영역이 필요하다.
- Stack 하나가 존재한다는 것은 하나의 실행흐름이 존재하는 것

- Thread 간에 Stack 영역은 공유가 불가능하고, 각각의 Thread 별로 Stack이 독립으로 존재해야 한다.
  - Thread간의 지역변수는 공유 불가

- Thread 간에 Heap 영역은 공유가 가능하다.
  - Instance는 참조값만 있다면 모든 Thread가 공유 가능



아래의 예제는 둘 이상의 쓰레드가 한 인스턴스의 메모리공간에 동시접근하기 때문에 오류가 발생할 수 있다!

```
class Sum {
     int num;
     public Sum(){ num=0; }
     public void addNum(int n){ num += n; }
     public int getNum(){ return num; }
}

public class AdderThread extends Thread {
     Sum sumInst;
     int start, end;

     public AdderThread(Sum sum, int s, int e) {
          sumInst = sum;
          start = s;
          end = e;
     }

     @Override
     public void run() {
          for(int i=start; i<=end; i++)
               sumInst.addNum(i);
     }
}

public class ThreadHeapMultiAccess {
     public static void main(String[] args) {
          Sum s = new Sum();
          AdderThread at1 = new AdderThread(s, 1, 50);
          AdderThread at2 = new AdderThread(s, 51`, 100);

          at1.start();
          at2.start();

          try {
```

join() : 너가 끝날 때까지 실행안하고 기다린다는 뜻
Main Thread 내에서 at1 Thread를 대상으로 join() Method를 호출하면 at1 Thread가 종료될 때까지 Main Thread는 멈춰서서 대기

```
            at1.join();
            at2.join();

        } catch (InterruptedException e) { e.printStackTrace(); }

        System.out.println("1~100까지의 합 : "+s.getNum());
    }
}
```