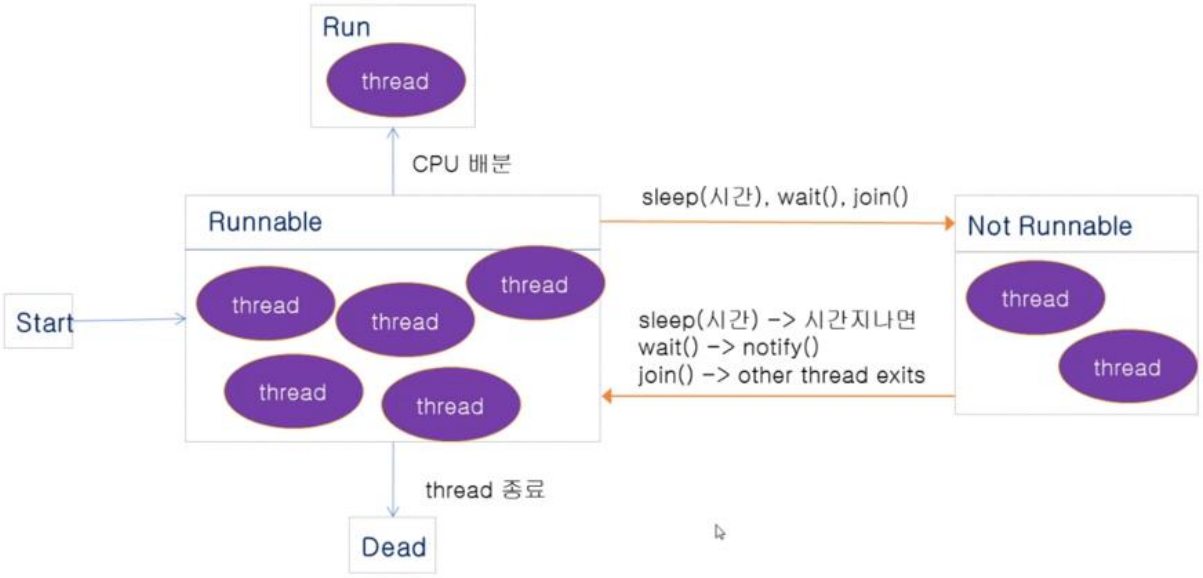
# 19. 스레드(thread)

## 1)스레드(thread)

스레드(thread)란 하나의 프로세스 내부에서 독립적으로 실행되는 하나의 작업 단위를 말하며, 세부적으로는 운영체제에 의해 관리되는 하나의 작업 혹은 태스크(task)를 의미한다. 스레드와 태스크(혹은 작업)은 바꾸어 사용해도 무관합니다. 일반적으로 동작하고 있는 프로그램을 프로세스(Process)라고 하는데 보통 한 개의 프로세스는 한 가지의 일을 하지만, 스레드를 이용하면 한 프로세스 내에서 두 가지 또는 그 이상의 일을 동시에 할 수 있으며 이를 멀티 스레드(multi thread)라고 한다. 이렇게 두가지 이상의 스레드를 가지는 프로세스를 멀티 스레드 프로세스 라고한다.

### (1) 스레드의 상태

스레드는 다음과 같은 생명주기 상태를 가진다.




**Runnable :** 쓰레드가 실행되기 위한 준비 단계 상태이다. start단계에서 스레드가 생성되고 start()메소드를 통해 Runnable로 넘어와 실행 되기를 기다리는 상태이다.

**Run :** 스레드가 cup를 점유하여 실행하고 있는 상태이며 JVM이 우선순위를 결정하고 run()메소드를 호출, run상태로 진입힌다.

**Not Runnable** : 스레드가 특정 메소드(sleep(), wait(), join() ..)로 인해 cup점유를 이탈한 상태다. 조건을 만족하면 다시 runnable로 돌아간다.

Ex) Sleep()에 의해 not runnable 상태가 되었다면 명시한 시간이 지나고,
Wait()에 의해 not runnable 상태가 되었다면 notify()메소드가 호출되면 runnable 상태로 돌아간다.

**Dead :** run 상태에서 스레드가 모두 실행되고 난 후 dead상태로 넘어온 완료 상태이다.

자바에서 스레드를 구현하는 방법은 **Runnable** 인터페이스를 구현하는 방법과

**Thread** 클래스를 상속받는 방법 두가지가 있다.

두 방법 모두 스레드를 통해 실행하려하는 로직의 내용을 run()메소드 안에 오버라이드 하면 된다.

**(2) Runnable**

Runnable 인터페이스를 구현한 target객체를 만들어 사용한다. 하나의 target객체를 여러 스레드객체가 매개변수로 입력받아 사용할수 있다. 즉 n개의 스레드가 하나의 target객체를 공유할 수 있다.

```java
// "안녕하세요 10번" 하는 target 또는 task라고도 한다.
public class TargetEx01 implements Runnable {
            //thread 객체 만들기 위해 Runnable 인터페이스의 run을 오버라이드 하여 사용해야한다.
    @Override
    public void run() {      //객체 생성후 .strat()를 실행시키면 객체 내부의 run()메소드를 실행
        for (int i = 0; i < 10; i++) {
            System.out.println("안녕하세요 - " + i);
            try {
                Thread.sleep(500);          //현재 스레드(작업)를 0.5초동안 대기상태로
                                            //현재 스레드라하면 해당 메소드의 동작 하나하나
                                            // for문의 반복 하나하나가 스레드에 해당한다.
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
    }
}

// "반갑습니다. 10번" 하는 target
public class TargetEx02 implements Runnable {

    @Override
    public void run() {
        for (int i = 0; i < 10; i++) {
            System.out.println(i + "번째, 반갑습니다.");
            try {
                Thread.sleep(500);              // 현재 스레드(작업)를 0.5초동안 대기상태로
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
    }
}

public class TargetExTestMain {
    public static void main(String[] args) { // main treadA treadB
        TargetEx01 target1 = new TargetEx01();
        TargetEx02 target2 = new TargetEx02();
      //Runnable target3 = new TargetEx01();  //상위 인터페이스 Runnable 형 객체 가능

                // A라는 이름의 thead객체 생성 target1.run() 수행
        Thread threadA = new Thread(target1, "A");  // 스레드 이름 설정 안할땐 자체적으로 생성됨
                // B라는 이름의 thread객체 생성 target2.run() 수행
        Thread threadB = new Thread(target2, "B");
      //Thread threadB = new Thread(target1, "B"); //하나의 타겟을 여러 스레드가 사용 할수있음
                          //파라미터로 들어가는 타겟은 Runnable인터페이스 구현한 클래스의 객체
        threadA.start();
        threadB.start();        // 실행가능한 준비상태

        for (int i = 0; i < 10; i++) {
            System.out.println("나는 main :" + Thread.currentThread().getName() + i);
                    // 현재 실행중인 프로세스 스레드 이름을 가져와서 출력 // main함수의 이름인 셈
            try {
                Thread.sleep(500);
            } catch (InterruptedException e) {
            }
        }
```

### (2) Thread

Thread클래스를 상속받아 오버라이딩하고 객체를 생성해 사용한다..

```java
//Thread t1 = new ThreadEx01();
//Thread t1 = new ThreadEx01("A");
public class TargetEx01 extends Thread {    //스레드는 일반클래스
    public TargetEx01() {}          //객체를 생성후 매개변수로 target을 넣지않고 실행
    public TargetEx01(String name) {
        super(name);
    }
    @Override           //일반 메소드이기에  run()메소드의 오버라이드를 강제 하지않는다.
    public void run() {
        for (int i = 0; i < 10; i++) {
            System.out.println(Thread.currentThread().getName());
    //현재 실행되고있는 해당 스레드의 이름을 가져오는 함수 Thread.currentThread().getName()
            System.out.println("안녕하세요 -" + i);
            try {
                Thread.sleep(500);  //Thread.sleep은 로직이 중단될수 있는 예외가 있다.
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
    }
}
public class TargetEx02 extends Thread {
    @Override
    public void run() {
        for (int i = 0; i < 10; i++) {
            System.out.println(Thread.currentThread().getName());
            System.out.println(i + "번째, 반갑습니다.");
            try {
                Thread.sleep(500);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
    }
}
public class TestMain {
    public static void main(String[] args) {

        Thread t1A = new TargetEx01();
        Thread t2B = new TargetEx02();
      //t2B.setName("B"); //t2B객체의 스레드 이름 생성

        t1A.start();
        t2B.start();

        System.out.println(Thread.currentThread().getName()); //메인 스레드명
        System.out.println("main 함수끝");
    }
}


```

**2)멀티 스레드**

멀티스레드란 하나의 프로세스에서 둘 이상의 스레드가 동시에 작업을 수행하는 것을 말한다.

여러 프로세스로 처리하던 일을 멀티스레드로 구현할 경우 메모리 공간과 시스템 자원 소모를 줄일 수 있는 이점이 있다. 멀티 프로세스는 독자적(프로세스는 데이터, 힙, 스텍영역을 모두 공유하지않는다.)인 반면 멀티스레드는 공유(스텍영역을 제외한 데이터와 힙영역을 공유)한다는 차이점이 있어 응답시간도 더 빠르고 서로 통신할수도 있다.

**(1) Synchronized**

객체에 선언되어 있는 인스턴스 변수를 스레드에서 공유하게 되어, 인스턴스의 값에 영향을 미치게 되는데, 이때 synchronized를 이용할 수 있다. Synchronized는 먼저 수행되는 스레드의 모든 작업이 끝날 때까지 다른 스레드는 기다리도록 하는 예약어로서 Synchronized 명령어가 있는 메소드는 하나의 스레드에서 작업시 다른 스레드에선 접근할 수 없고 먼저 접근한 스레드의 로직이 끝날 때 접근가능하다.

```
1  public class TargetEx implements Runnable {      // Runnable로 구현받은 타겟
2      private int num = 0;
3      @Override
4      public void run() {
5          for (int i = 0; i < 10; i++) {
6              out();
7              try {
8                  Thread.sleep(500);
9              } catch (InterruptedException e) {
10
11             }
12         }
13     }
14     private synchronized void out() {            //메서드 단위로만 synchronized 가능하다.
15         if (Thread.currentThread().getName().equals("A")) {   // "A"스레드가 실행될 경우에만
16             System.out.println("~~~~~A스레드 수행중~~~~~~~~~");  // num이 증가한다.
17             num++;
18         }
19         System.out.println(Thread.currentThread().getName() + "의 num =" + num);
20     }
21 }
22 public class TargetExTestMain {
23
24     public static void main(String[] args) {
25         TargetEx targetObject = new TargetEx();
26
27         Thread threadA = new Thread(targetObject, "A");
28         Thread threadB = new Thread(targetObject, "B");
29
30         threadA.start();
31         threadB.start();
32
33    //synchronized 메소드가 아니었을때는 threadA가 실행도중 threadB가 먼저 실행될때도 있었지만
34    //메소드에 synchronized를 명시한후 threadB의 out()은 반드시 threadA의 out()실행후 실행되었다.
35
36         System.out.println("main 함수 끝");
37     }
38 }
39
40 main 함수 끝
41 ~~~~~A스레드 수행중~~~~~~~~~        ~~~~~A스레드 수행중~~~~~~~~~
42 A의 num =1                              A의 num =6
43 B의 num =1                              B의 num =6
44 B의 num =1
45 ~~~~~A스레드 수행중~~~~~~~~~        ~~~~~A스레드 수행중~~~~~~~~~
46 A의 num =2                              A의 num =7
47 B의 num =2                              B의 num =7
48 ~~~~~A스레드 수행중~~~~~~~~~        ~~~~~A스레드 수행중~~~~~~~~~
49 A의 num =3                              A의 num =8
50 B의 num =3                              B의 num =8
51 ~~~~~A스레드 수행중~~~~~~~~~        ~~~~~A스레드 수행중~~~~~~~~~
52 A의 num =4                              A의 num =9
53 B의 num =4                              B의 num =9
54 ~~~~~A스레드 수행중~~~~~~~~~        ~~~~~A스레드 수행중~~~~~~~~~
55 A의 num =5                              A의 num =10
56 B의 num =5
```