

# 제19장

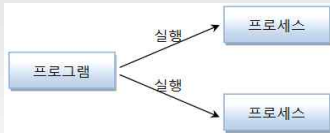
## 멀티 스레드-Part1



# 1. 프로세스(process)란?

■ 보통 프로세스라고 하면 실행 중인 하나의 프로그램을 칭한다.

— 하나의 프로그램이 여러 개의 프로세스를 만들기도 한다.(ex. 인터넷 창 등)



## 2. 멀티 태스킹(multi tasking)

- **멀티 태스킹은 통상 두 가지 이상의 작업을 동시에 처리하는 것을 칭한다.**

ex) 음악을 들으면서, 워드 작업이나 웹 쇼핑을 하는 등...)

- 구현 방법

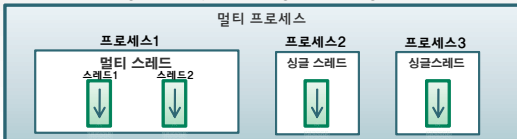
- **멀티 프로세스**

-> 말 그대로 독립적 프로그램을 여러 번 실행하고 작업을 처리하는 것

- **멀티 스레드**

-> 하나의 프로그램을 실행하여, 내부적으로 여러 가지 작업을 처리하는 것

ex) 카카오톡 - 채팅 스레드, 파일 전송 스레드 등



### 3. 메인 스레드(main thread)

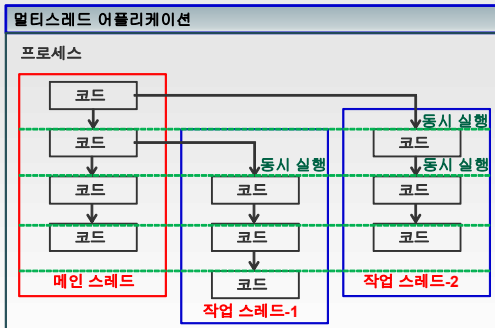
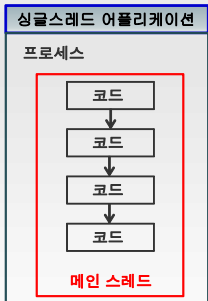
- 모든 자바 프로그램은 JVM의 메인스레드가 main()를 실행하면서 시작한다.
- main()의 첫 코드부터 아래로 순차적으로 실행한다.
- 실행 종료 조건은 마지막 코드를 실행하거나, return문을 만나면 종료한다.

```
public static void main(String[] args) {  
    String str = "부장";  
    //문자열이 조건으로 왔다.  
    switch(str) {  
        case "부장" :  
            System.out.println("450만원");  
            break;  
        case "차장" :  
            System.out.println("400만원");  
            break;  
        case "과장" :  
            System.out.println("350만원");  
            break;  
        case "대리" :  
            System.out.println("250만원");  
            break;  
        case "사원" :  
            System.out.println("200만원");  
            break;  
    }  
}
```

일련의 코드의 실행 흐름 → 스레드

## 4. 메인 스레드와 작업 스레드

- 메인 스레드는 작업 스레드들을 만들어서 병렬로 코드를 실행할 수 있으며,  
그것이 곧, 멀티 태스킹을 수행하는 것이다.

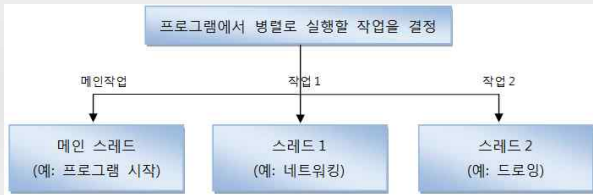


## 5. 프로세스의 종료

- 프로세스 종료는 싱글, 멀티 스레드에 따라 달리 종료되어 진다.
  - ▶ 싱글 스레드: 메인 스레드가 종료하면 프로세스도 함께 종료됨.
  - ▶ 멀티 스레드: 실행 중인 스레드가 하나라도 있다면, 프로세스 종료되지 않는다.
    - ☞ 메인 스레드가 작업 스레드보다 먼저 종료되더라도, 작업 스레드가 계속 실행 중이라면, 프로세스는 종료되지 않는다.

## 6. 작업 스레드 생성과 실행 - 1

- 몇 개의 작업을 병렬로 실행해야 할지 설계 단계에서 결정해야 한다.



- 작업 스레드를 생성하는 방법은 아래와 같다.

- Thread 클래스로부터 직접 생성한다.
- Thread 하위 클래스로부터 생성한다.

## 6. 작업 스레드 생성과 실행 - 2

### ❶ ■ Thread클래스로부터 직접 생성(매우 중요함 모르면 암기)

```
Class Task implements Runnable {  
    @Override  
    public void run() {  
        스레드가 실행할 코드;  
    }  
}
```

\*동시에 실행할 작업이 있으면 좌측과 같이 클래스를 만든다. Task클래스는 작업스레드가 실행할 수 있는 코드를 작성하는데, Runnable인터페이스에는 run()이란 추상 메서드가 있어서 이것을 재정의 해줘야 한다. 이게 바로 작업스레드가 실행할 코드가 되는 것이다.

#### ▶ 방법 1(인터페이스를 통한 구현객체 이용) - 가장 많이 이용함

```
Runnable task = new Task();
```

인터페이스 변수에  
구현객체를 생성후  
대입(다형성)

```
Thread thread = new Thread(task);
```

#### ▶ 방법 2(익명 구현객체 이용) 간단히 이용

```
Thread thread = new Thread(new Runnable(){  
    @Override  
    public void run(){  
        스레드가 실행할 코드  
    }  
});
```

#### ▶ 방법 3(람다(함수적 인터페이스)식 이용) 코드 최소화

```
Thread thread = new Thread( ()-> {  
    스레드가 실행할 코드  
});
```

```
thread.start();
```

\* 자바에서는 하나의 스레드도 객체화 시켜서 사용한다.

\* Thread의 start()를 호출하면, 자동으로 run()이 실행됨.



## 6. 작업 스레드 생성과 실행 - 3

### ○ ■ Thread 하위 클래스로부터 생성(중요함 모르면 암기)

```
public class WorkerThread extends Thread{  
    @Override  
    public void run() {  
        스레드가 실행할 코드;  
    }  
}
```

Thread를 상속 받아서, run()에서드를 작성함

```
Thread thread = new WorkerThread();
```

```
Thread thread = new Thread(){  
    @Override  
    public void run() {  
        스레드가 실행할 코드;  
    }  
};
```

Thread를 익명자식객체로 생성해서 run()에서드를 작성함.

```
thread.start();
```

## 7. 작업 스레드의 이름

### ■ 스레드의 이름

- 메인 스레드 이름: main → JVM이 생성과 동시에 부여함
- 작업 스레드 이름 (자동 설정) : Thread-n

```
thread.getName();
```

- 작업 스레드 이름 변경

```
thread.setName("스레드 이름");
```

- 코드 실행하는 현재 실행 중인 스레드 객체의 참조 얻기

```
Thread thread = Thread.currentThread();
```

```
thread.getName();
```

\*실행 중인 스레드 객체를 얻어서, 그 스레드의 이름을 얻으면 된다.

프로그램에서 빈번하게 사용은 안 된다. 단지, 개발할 때 디버깅 과정에서 현재 실행 중인 Thread가 무엇인지를 프로그래머가 알고자 할 때 사용한다.

## 8. 동시성과 병렬성

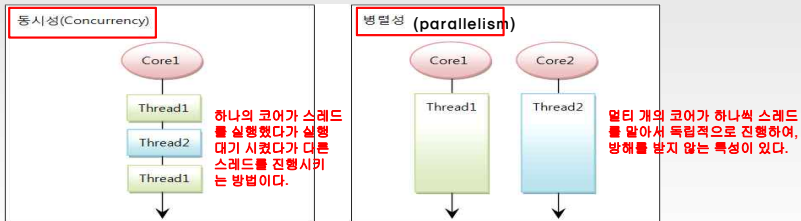
### ■ 동시성과 병렬성

#### ● 동시성

▶ 멀티작업 위해 하나의 코어에서 멀티 스레드가 번갈아 가며 실행하는 성질

#### ● 병렬성

▶ 멀티작업을 위해 멀티 코어에서 개별 스레드를 동시에 실행하는 성질



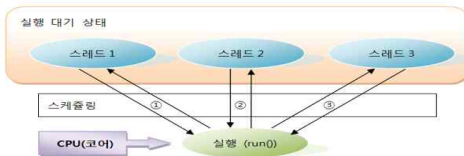
요즘은 대부분 멀티코어이기 때문에 우측그림을 확장시켜 나가보면 된다. 하지만 워낙 스레드가 많아서 동시성과 병렬성을 동시에 가지는 형태가 대부분이다.

## 9. 스레드 우선 순위-1

### ❖ 스레드 스케줄링

#### ● 스레드의 개수가 코어의 수보다 많을 경우

- ▶ 스레드를 어떤 순서로 **동시성**으로 실행할 것인가 결정 → 스레드 스케줄링
- ▶ 스케줄링 의해 스레드들은 번갈아 가며 run() 메서드를 조금씩 실행



우선 순위 스케줄링  
1) 1 ~ 10까지 우선순위 있음  
(사용자 설정)  
2) 우선순위가 높을수록 실행  
기회가 높아 빨리 종료되는  
현상이 나옴

#### ● 자바의 스레드 스케줄링

- ▶ 우선 순위(Priority) 방식과 순환 할당(Round-Robin) 방식 사용

→ 우선 순위 방식 (**코드로 제어 가능**) → 사용자가 설정하는 방식임.

: 우선 순위가 높은 스레드가 실행 상태를 더 많이 가지도록 스케줄링 하는 방식

→ 순환 할당 방식 (**코드로 제어할 수 없음**) → **자바에서는 기본적으로 이 방식을 사용됨.**

: 시간 할당량(Time Slice)을 정해서 하나의 스레드를 정해진 시간 만큼 CPU가 실행하는 방식

## 9. 스레드 우선 순위-2

### ❖ 스레드 우선 순위

- 스레드들이 동시성을 가질 경우 **우선적으로 실행할 수 있는 순위**
- 우선 순위는 1(낮음)에서부터 10(높음)까지로 부여

▶ **모든 스레드의 우선순위는 5의 우선 순위**

- **우선 순위 변경 방법**

```
thread.setPriority();
```

```
thread.setPriority(Thread.MAX_PRIORITY);  
thread.setPriority(Thread.NORM_PRIORITY);  
thread.setPriority(Thread.MIN_PRIORITY);
```

- **우선 순위 효과**

#### ▶ 싱글코어인 경우

→ 우선 순위가 높은 스레드가 실행기회를 더 많이 가지기 때문에, 우선순위가 낮은 스레드보다 작업을 빨리 끝내는 경향이 많다.

#### ▶ 멀티코어인 경우

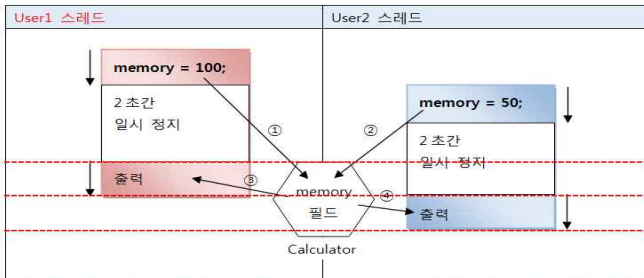
→ 쿼드코어인 경우 4개의 스레드가 병렬성으로 실행될 수 있기 때문에, 4개 이하의 스레드들은 우선 순위 방식은 크게 영향을 받지 못한다. 최소 5개 이상 되어야 우선 순위에 영향을 받는다.



## 10. 동기화 메서드와 동기화 블록-1

### ❖ 공유 객체를 사용할 때의 주의할 점

#### ● 멀티 스레드가 하나의 객체를 공유해서 생기는 오류



여러 개의 스레드가 하나의 객체를 공유한다면, 원하는 결과값이 출력이 되질 아니한다. 즉, 데이터의 신뢰성이 없어질 뿐만 아니라, 프로그램 사용자가 원하는 결과를 정확하게 주지 않을 수 있다. 그 이유는 CPU 스케줄링에 따라, 스레드는 객체(멤버변수, 멤버메서드)를 임의로 사용하여 값을 변경할 수 있는 경우도 생기기 때문이다.

## 10. 동기화 메서드와 동기화 블록-2

### ❖ 동기화 메서드 및 동기화 블록 - **synchronized**

- 단, 하나의 스레드만 실행할 수 있는 메서드 또는 블록을 말한다.
- 다른 스레드는 메서드나 블록이 실행이 끝날 때까지 대기해야 한다.
- 동기화 메소드

```
public synchronized void method()
{
    임계 영역; //오로지 단, 하나의 스레드만 실행
}
```

### ● 동기화 블록

```
public void method()
{
```

//여러 개의 스레드가 실행가능한 영역

...

**synchronized(공유객체) → 보통 this**

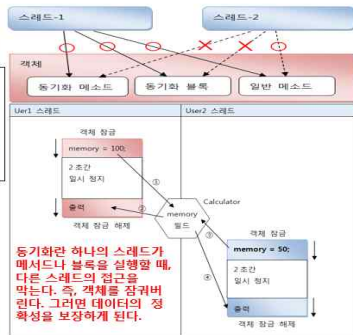
```
{
    임계 영역; //오로지 단, 하나의 스레드만 실행
}
```

//여러 개의 스레드가 실행가능한 영역

**임계영역 : 둘 이상의 스레드가 동시에 접근해서는 안 되는 공유 자원 (자료 구조 또는 장치)을 접근하는 코드를 지칭 한다.**

```
* Class A {
    synchronized void m10 ( ..);
    synchronized void m20 ( ..);
    void m30{
        synchronized(this){ ... }
    }
}
```

위와 같이, m30가 실행 중이라면, Synchronized가 붙어있는 것은 실행 자체가 안된다.



감사합니다.

