제19장
멀티 스레드-Part1

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

Thread를 상속 받아서, run()메서드를 작성함

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

Thread를 익명자식객체로 생성해서 run()메서드를 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

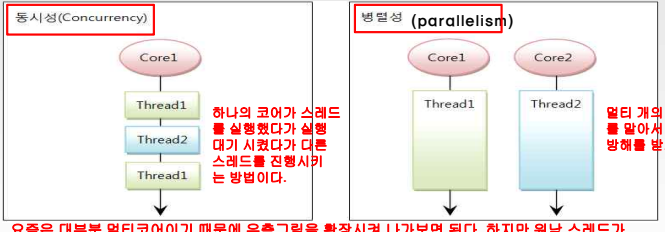
▣ 동시성과 병렬성

● 동시성

▶ 멀티작업 위해 하나의 코어에서 멀티 스레드가 번갈아 가며 실행하는 성질

● 병렬성

▶ 멀티작업을 위해 멀티 코어에서 개별 스레드를 동시에 실행하는 성질

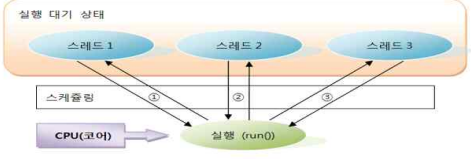


요즘은 대부분 멀티코어이기 때문에 우측그림을 확장시켜 나가보면 된다. 하지만 워낙 스레드가 많아서 동시성과 병렬성을 동시에 가지는 형태가 대부분이다.

9. 스레드 우선 순위-1

❖ 스레드 스케줄링

- **스레드의 개수가 코어의 수보다 많을 경우**
  - ▶ 스레드를 어떤 순서로 동시성으로 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 사용함.
      - : 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();
```

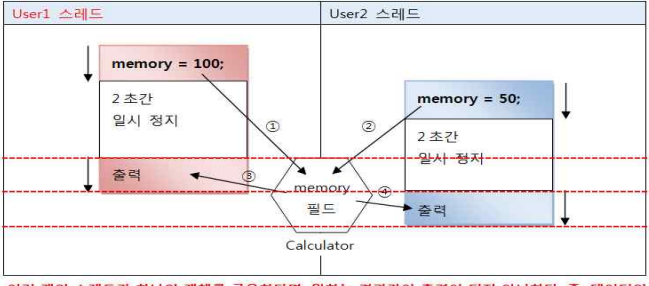
```
thread.setPrioroty(Thread.MAX_PRIORITY);
thread.setPrioroty(Thread.NORM_PRIORITY);
thread.setPrioroty(Thread.MIN_PRIORITY);
```

- 우선 순위 효과
  - ▶ 싱글코어인 경우
    - → 우선 순위가 높은 스레드가 실행기회를 더 많이 가지기 때문에, 우선순위가 낮은 스레드 보다 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 CPU스케줄링에 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
}
```

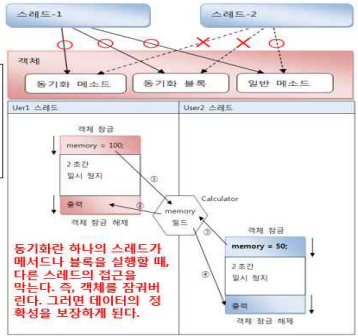
- 동기화 블록

```
public void method()
{
    //여러 개의 스레드가 실행가능한 영역
    .. ..
    synchronized(공유객체) → 보통 this
    {
        임계 영역;  //오로지 단, 하나의 스레드만 실행
    }

    //여러 개의 스레드가 실행가능한 영역
}
```

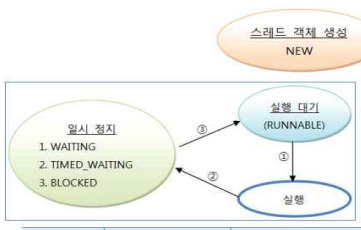
```
* Class A {
    synchronized void m1() { ..};
    synchronized void m2() { ..};
    void m3(){
        synchronized(this){ ... }
}
위와 같이, m3()가 실행 중이라면,
Synchronized가 붙어있는 것은
실행 자체가 안된다.
```

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



11. 스레드 상태

❖ 스레드 상태



상태	열거 상수	설명
객체 생성	NEW	스레드 객체가 생성, 아직 start() 메소드가 호출되지 않은 상태
실행 대기	RUNNABLE	실행 상태로 언제든지 갈 수 있는 상태
일시 정지	BLOCKED	사용코저하는 객체의 락이 풀릴 때까지 기다리는 상태 synchronized메서드, 블록
	WAITING	다른 스레드가 통지할 때까지 기다리는 상태 Object의 wait()메서드, notify()메서드, notifyAll() 메서드
	TIMED_WAITING	주어진 시간 동안 기다리는 상태 Thread의 sleep()메서드
종료	TERMINATED	실행을 마친 상태

감사합니다.