12-1. 멀티 스레드

혼자 공부하는 자바 (신용권 저)

목차

시작하기 전에

[핵심 키워드] : 프로세스, 멀티 스레드, 메인 스레드, 작업 스레드, 동기화 메소드

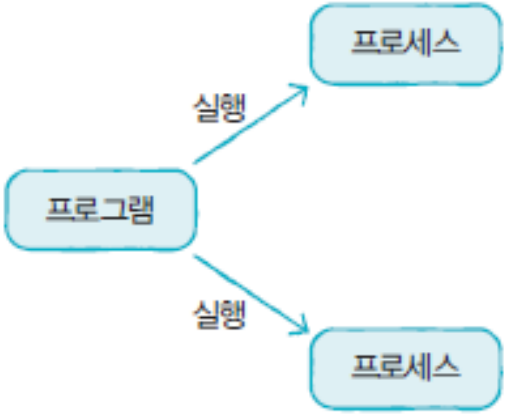
[핵심 포인트]

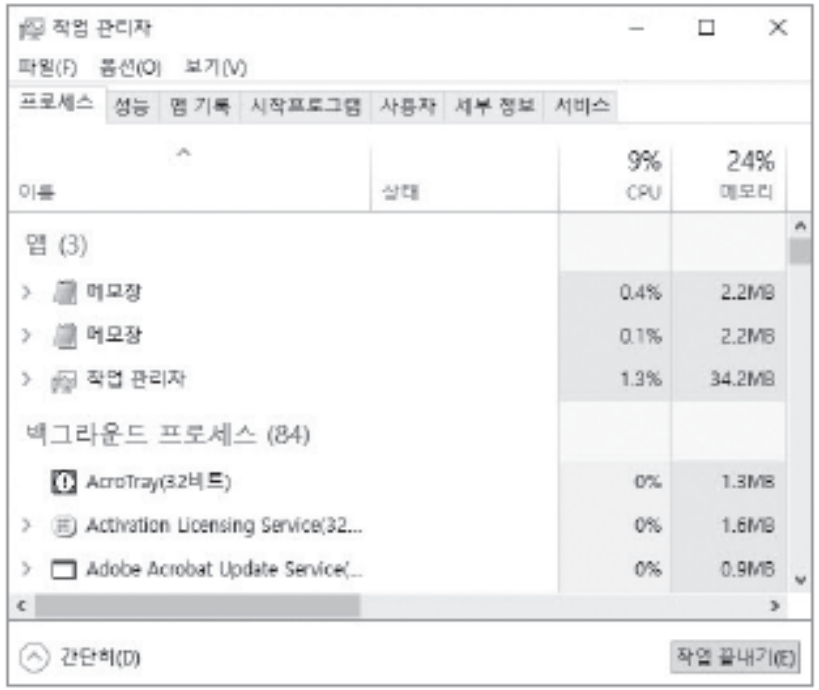
애플리케이션을 실행하면 운영체제로부터 실행에 필요한 메모리를 할당받아 애플리케이션이 실행되는데, 이것을 프로세스라 한다. 그리고 프로세스 내부에서 코드의 실행 흐름을 스레드라 한다. 애플리케이션 개발에 필수 요소인 스레드에 대해 살펴본다.

시작하기 전에

❖ 프로세스 (process)

- 실행 중인 하나의 애플리케이션
- 애플리케이션이 실행되면 운영체제로부터 실행에 필요한 메모리 할당받아 코드를 실행함
- 멀티 프로세스 역시 가능함

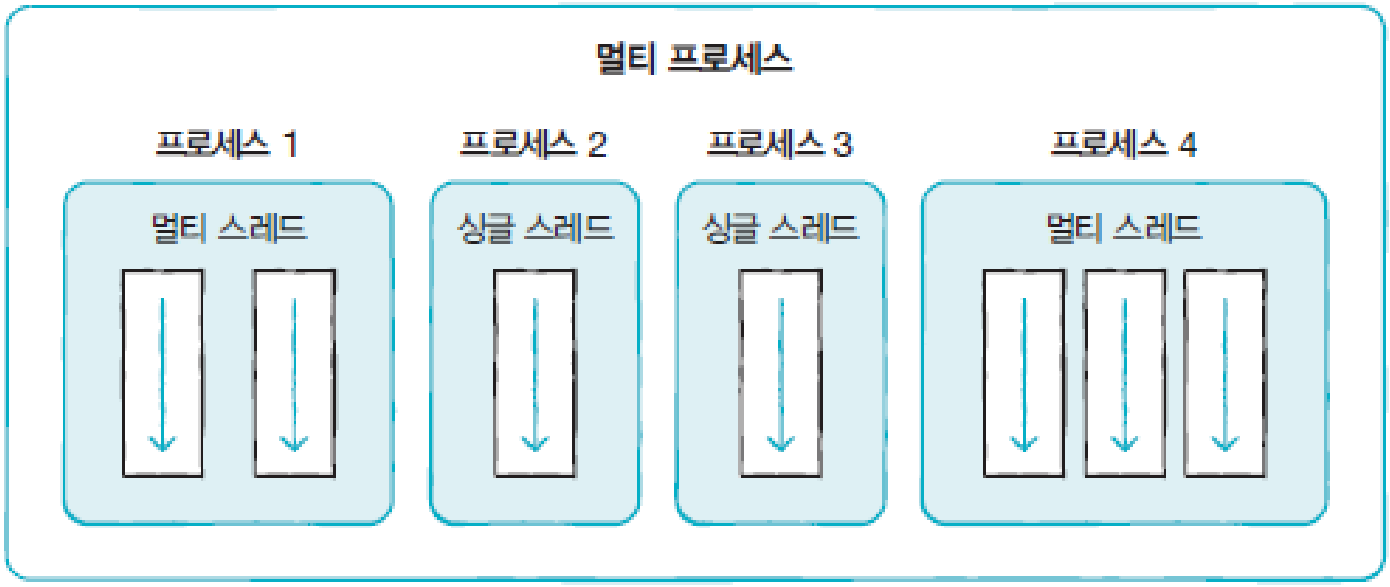




스레드

❖ 스레드 (thread)
- 한 가지 작업을 실행하기 위해 순차적으로 실행할 코드를 이어놓은 것
- 하나의 스레드는 하나의 코드 실행 이름

❖ 멀티 스레드 (multi thread)
- 하나의 프로세스로 두 가지 이상의 작업을 처리
- 데이터 분할하여 병렬로 처리하거나 다수 클라이언트 요청 처리하는 서버 개발하는 등의 용도
- 한 스레드가 예외 발생시킬 경우 프로세스 자체가 종료될 수 있음

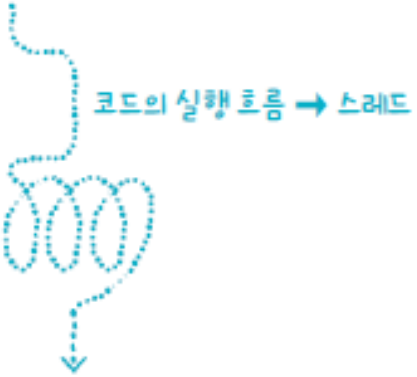


메인 스레드

❖ 메인 스레드 (main thread)

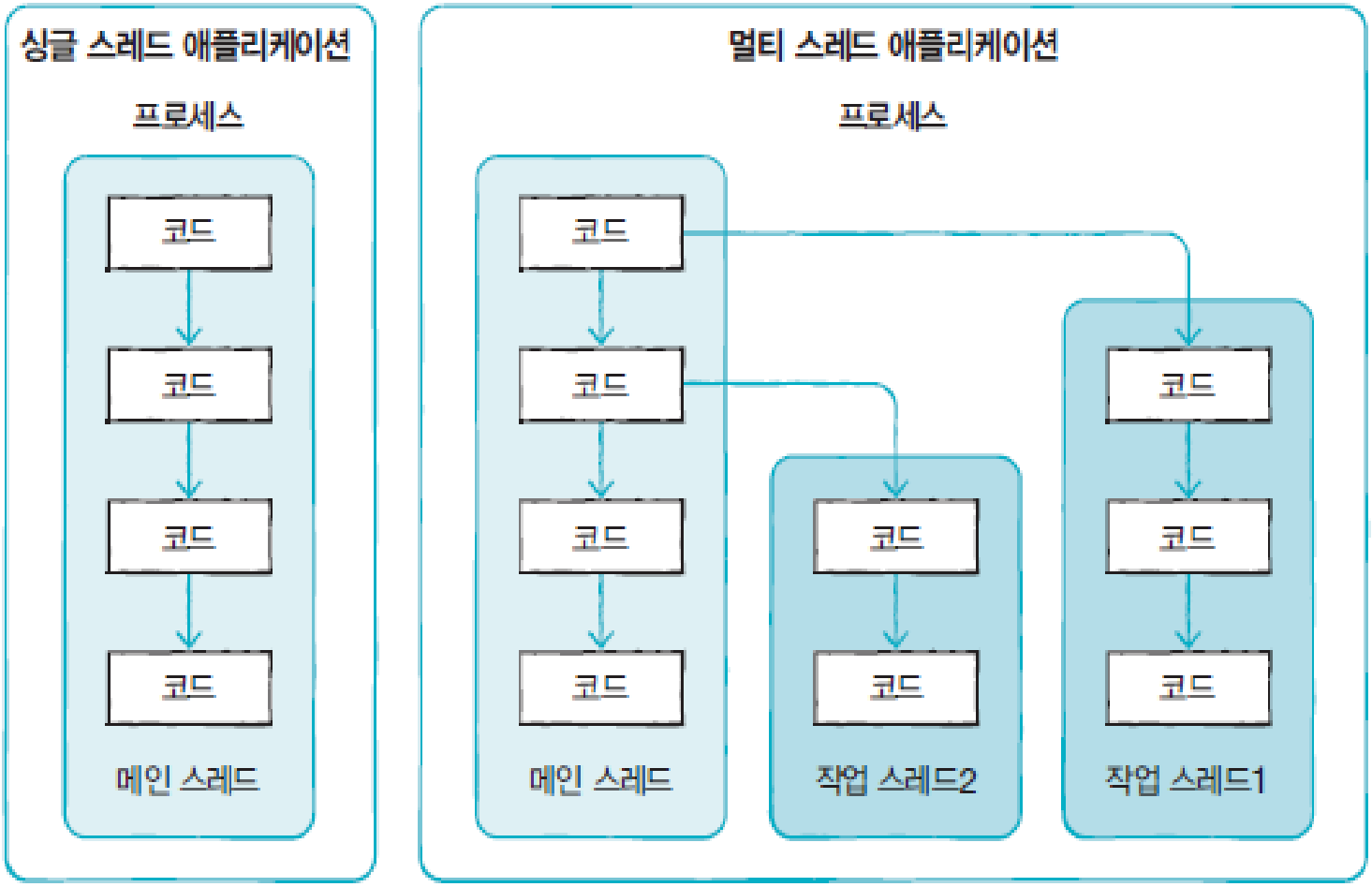
- 모든 자바 애플리케이션은 메인 스레스가 main() 메소드 실행하면서 시작됨
- main() 메소드의 첫 코드부터 아래로 순차적으로 실행

```
public static void main(String[] args) {
    String data = null;
    if(…) {
    }
    while(…) {
    }
    System.out.println("…");
}
```



- 필요에 따라 작업 스레드들 만들어 병렬로 코드 실행 가능
- 멀티 스레드 애플리케이션에서는 실행 중인 스레드 하나라도 있으면 프로세스 종료되지 않음

메인 스레드

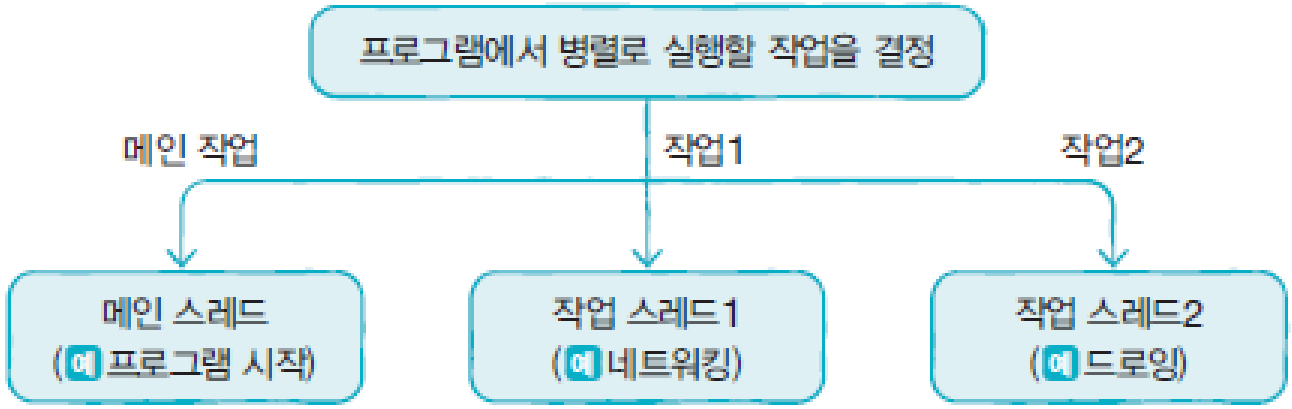

싱글 스레드 애플리케이션
프로세스
코드
코드
코드
코드
메인 스레드
멀티 스레드 애플리케이션
프로세스
코드
코드
코드
코드
메인 스레드
코드
코드
작업 스레드2
코드
코드
코드
작업 스레드1


혼자
공부하는
자바

작업 스레드 생성과 실행

❖ 작업 스레드

- 멀티 스레드로 실행하는 애플리케이션 개발하려면 몇 개의 작업을 병렬로 실행할지 우선 결정한 뒤 각 작업별로 스레드 생성해야
- 작업 스레드 역시 객체로 생성되므로 클래스 필요
  - Thread 클래스 상속하여 하위 클래스 만들어 사용할 수 있음



작업 스레드 생성과 실행

❖ Thread 클래스로부터 직접 생성

- Runnable을 매개값으로 갖는 생성자 호출

```
Thread thread = new Thread(Runnable target);
```

- 구현 객체 만들어 대입 필요

```
class Task implements Runnable {
  public void run() {
    스레드가 실행할 코드;
  }
}
```

- 구현 객체 매개값으로 Thread 생성자 호출하면 작업 스레드 생성됨

```
Runnable task= new Task();

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

작업 스레드 생성과 실행

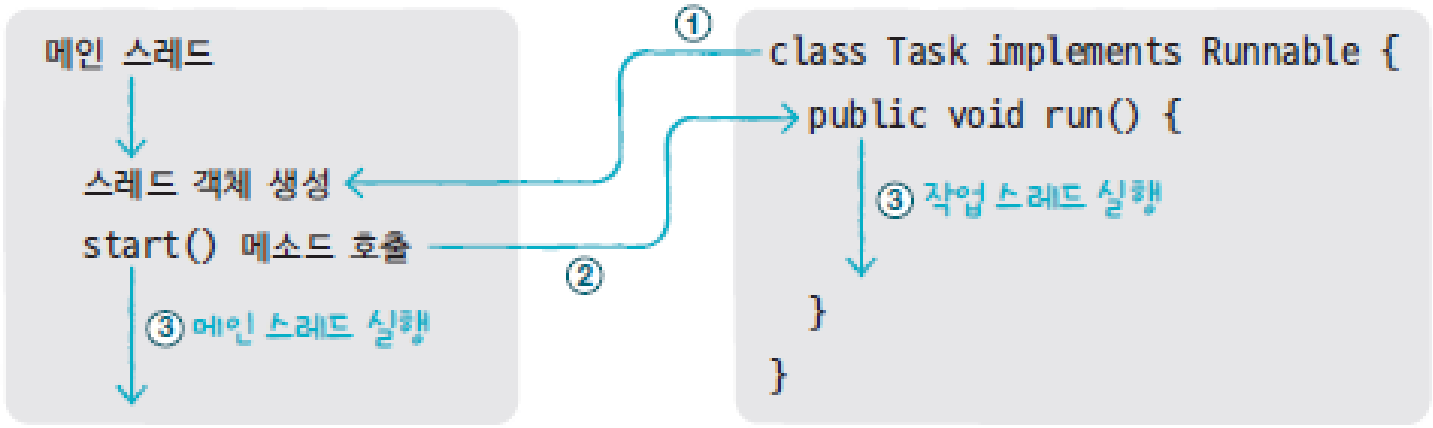
- Runnable 익명 객체를 매개값으로 사용하여 Thread 생성자 호출할 수도 있음

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

익명 구현 객체

- start() 메소드 호출하면 작업 스레드 실행

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



작업 스레드 생성과 실행

❖ Thread 하위 클래스로부터 생성

- Thread의 하위 클래스로 작업 스레드를 정의하면서 작업 내용을 포함
- 작업 스레드 클래스 정의하는 법

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

run() 메소드 재정의

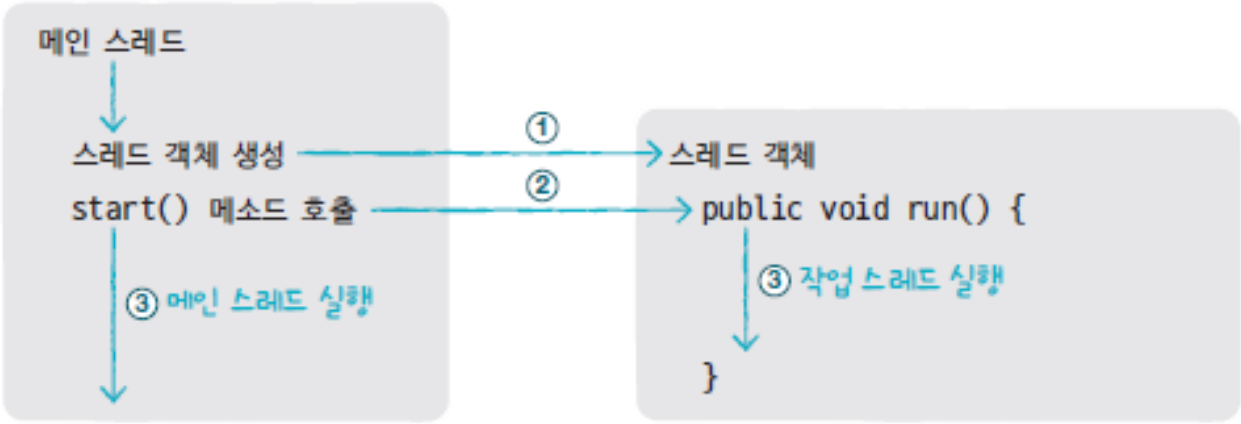
```
Thread thread = new Thread() {
  public void run() {
    스레드가 실행할 코드;
  }
};
```

익명 자식 객체

작업 스레드 생성과 실행

- 작업 스레드 객체 생성 후 strat() 메소드 호출하면 run() 메소드가 실행

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



작업 스레드 생성과 실행

- 메인 스레드는 'main' 이름 가지며, 우리가 직접 생성한 스레드는 자동적으로 'Thread-n' 이름 설정됨
  - setName() 메소드로 이름 변경 가능

```
thread.setName("스레드 이름");
```

  - getName() 메소드로 스레드 이름 알 수 있음

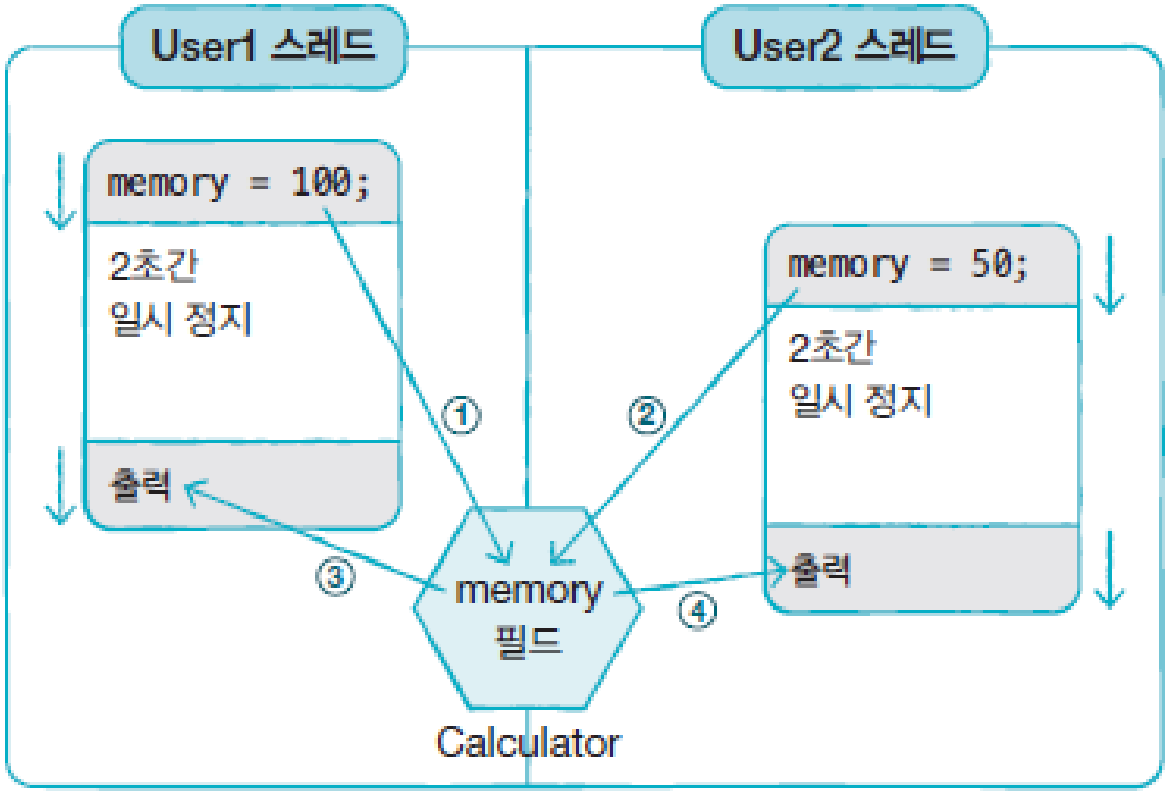
```
thread.getName();
```

  - currentThread() 메소드로 현재 스레드의 참조 얻을 수 있음

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

동기화 메소드

- 공유 객체를 사용할 때 주의할 점
  - 멀티 스레드 프로그램에서 스레드들이 객체 공유해서 작업해야 하는 경우 의도했던 것과 다른 결과 나올 수 있음

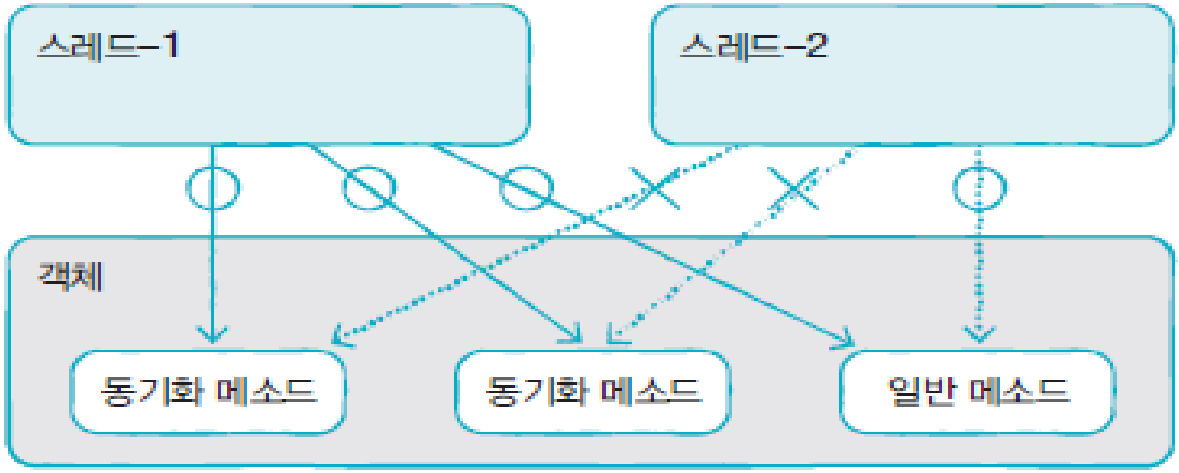


동기화 메소드

❖ 동기화 메소드

- 스레드가 사용 중인 객체를 다른 스레드가 변경할 수 없게 하려면 스레드 작업 끝날 때까지 객체에 잠금 걸어야 함
- 임계 영역 (critical section) : 단 하나의 스레드만 실행할 수 있는 코드 영역
- 동기화 (synchronized) 메소드 : 스레드가 객체 내부의 동기화 메소드 실행하면 즉시 객체에 잠금 걸림

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



키워드로 끝내는 핵심 포인트

- 프로세스 : 애플리케이션 실행하면 운영체제로부터 실행에 필요한 메모리 할당받아 애플리케이션 실행됨.

- 멀티 스레드 : 하나의 프로세스 내에 동시 실행하는 스레드가 2개 이상인 경우

- 메인 스레드 : 자바의 모든 어플리케이션은 메인 스레드가 main() 메소드 실행하면서 시작. main() 메소드의 첫 코드부터 아래로 순차 실행하고, main() 메소드의 마지막 코드 실행하거나 return 문 만나면 실행이 종료

- 작업 스레드 : 메인 작업 이외에 병렬 작업의 수만큼 생성하는 스레드. 객체로서 생성되기 때문에 클래스 필요. Thread 클래스를 직접 객체화해서 생성할 수도 있고, Thread 클래스를 상속해서 하위 클래스 만들어 생성할 수도 있음

- 동기화 메소드 : 멀티 스레드 프로그램에서 단 하나의 스레드만 실행할 수 있는 코드 영역을 임계 영역이라 함. 이를 지정하기 위해 동기화 메소드가 제공됨. 스레드가 객체 내부의 동기화 메소드 실행하면 즉시 객체에 잠금 걸어 다른 스레드가 동기화 메소드 실행하지 못하게 함

12-2. 스레드 제어

혼자 공부하는 자바 (신용권 저)

목차

시작하기 전에

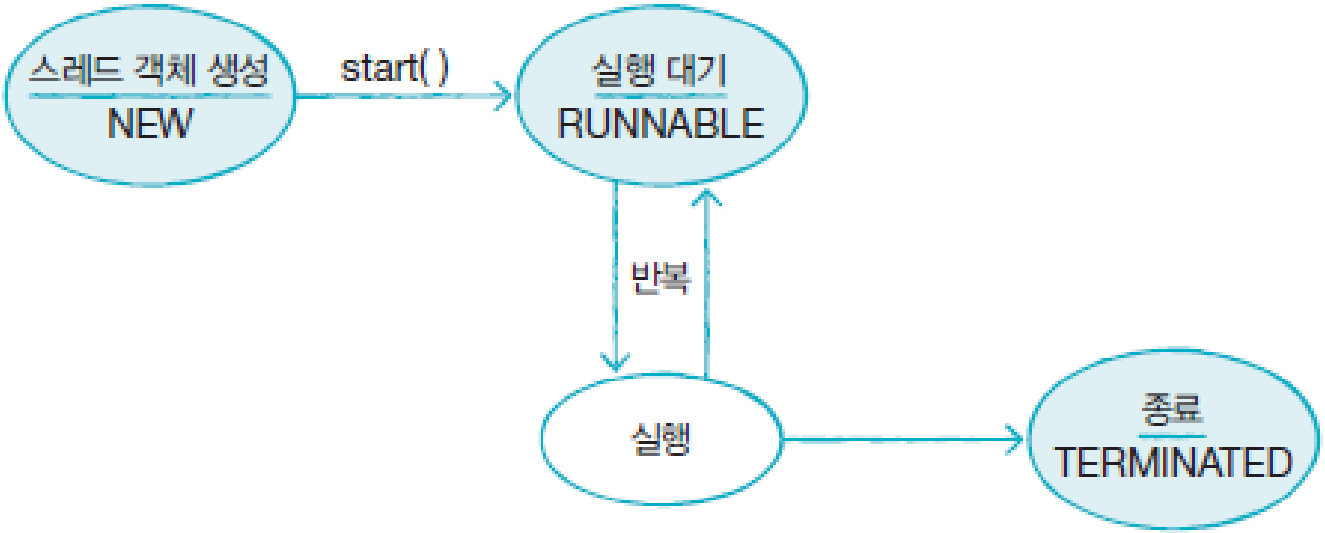
[핵심 키워드] : 스레드 상태, 일시정지, 안전한 종료, 데몬 스레드

[핵심 포인트]

스레드를 생성하고 시작하면 다양한 상태를 가지게 된다. 이러한 스레드 상태는 자동으로 변경될 수도 있고 코드에 의해 변경될 수도 있다. 스레드의 상태를 변경해 제어하는 방법에 대해 알아본다.

시작하기 전에

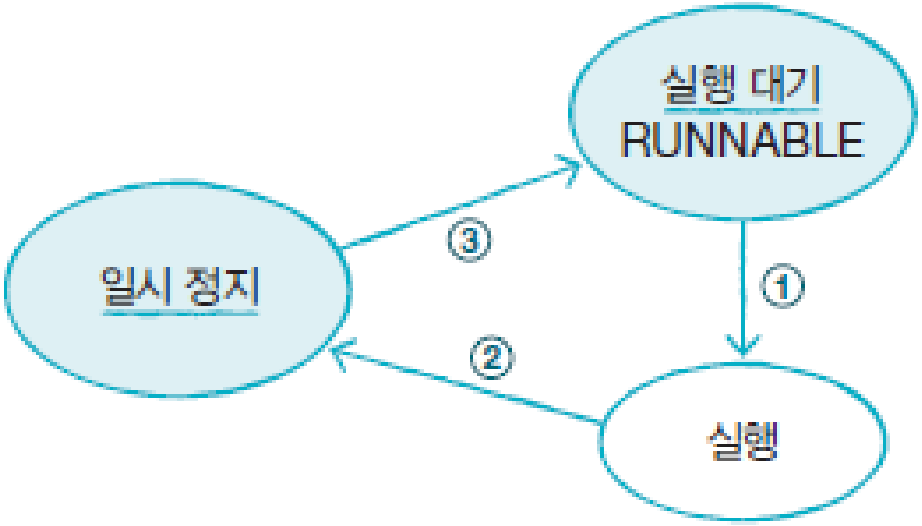
- 스레드 객체 생성하고 strat() 메소드를 호출하면 바로 실행되는 것이 아니라 실행 대기 상태가 됨
  - 실행 상태 스레드는 run() 메소드를 모두 실행하기 전 다시 실행 대기 상태로 돌아갈 수 있음
  - 실행 대기 상태에 있는 다른 스레드가 선택되어 실행 상태가 되기도 함
  - 실행 상태에서 run() 메소드의 내용이 모두 실행되면 스레드 실행이 멈추고 종료 상태가 됨



스레드 상태

❖ 실행 (running) 상태
- 실행 대기 상태의 스레드 중에서 운영체제가 하나를 선택하여 CPU가 run() 메소드를 실행하도록 했을 때
- Run() 메소드 모두 실행하기 전에 다시 실행 대기 상태로 돌아갈 수 있음

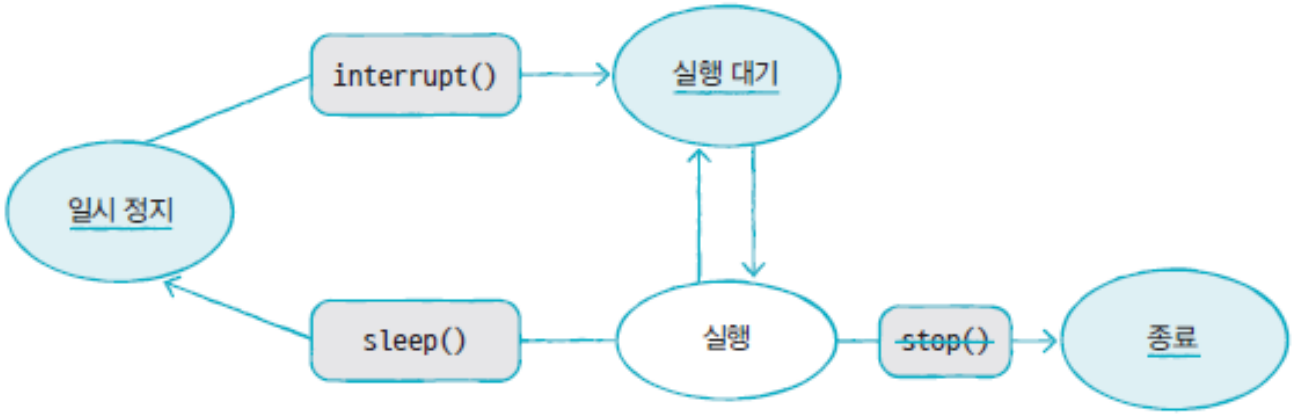
❖ 종료 (terminated) 상태
- 실행 상태에서 run() 메소드 종료되면 더 이상 실행할 코드 없기 때문에 스레드 실행이 정지됨



스레드 상태 제어

❖ 스레드 상태 제어

- 실행 중인 스레드의 상태를 변경
- 스레드 상태 변화에 필요한 메소드를 정확히 파악해야



메소드	설명
interrupt()	일시 정지 상태의 스레드에서 InterruptedException을 발생시켜, 예외 처리 코드(catch)에서 실행 대기 상태로 가거나 종료 상태로 갈 수 있도록 합니다.
sleep(long millis)	주어진 시간 동안 스레드를 일시 정지 상태로 만듭니다. 주어진 시간이 지나면 자동적으로 실행 대기 상태가 됩니다.
stop()	스레드를 즉시 종료합니다. 불안전한 종료를 유발하므로 사용하지 않는 것이 좋습니다.

스레드 상태 제어

- 주어진 시간 동안 일시 정지
  - Thread 클래스의 정적 메소드 sleep() 사용

```
try {
  Thread.sleep(1000);
} catch(InterruptedException e) {
  //interrupt() 메소드가 호출되면 실행
}
```

스레드 상태 제어

❖ 스레드의 안전한 종료

- 실행 중인 스레드를 즉시 종료해야 하는 경우

- stop 플래그를 이용하는 방법

```
public class XXXThread extends Thread {
  private boolean stop;  //stop 플래그 필드

  public void run() {
    while( !stop ) {  <-- stop이 true가 되면 run()이 종료
      스레드가 반복 실행하는 코드;
    }
    //스레드가 사용한 자원 정리
  }
}
```

키워드로 끝내는 핵심 포인트

- 스레드 상태 : 스레드를 생성하고 시작하면 스레드는 다양한 상태를 가지게 되며, 이는 자동으로 혹은 코드에 의해 변경될 수 있다

- 일시 정지 : 실행 중인 스레드를 일정 시간 멈추게 하려는 경우 Thread 클래스의 정적 메소드인 sleep()을 사용. Thread.sleep() 메소드를 호출한 스레드는 주어진 시간 동안 일시정지 상태가 되고 다시 실행 대기 상태로 돌아감

- 안전한 종료 : 스레드를 안전하게 종료하기 위해 stop 플래그나 interrupt() 메소드를 이용할 수 있다.

- 데몬 스레드 : 주 스레드의 작업을 돕는 보조적 역할 하는 스레드. 주 스레드가 종료되면 자동 종료된다.

Thank You!

혼자 공부하는 자바 (신용권 저)

한빛미디어 Hanbit Media, Inc.