JAVA 프로그래밍

Chapter 15 멀티 스레드

목차

- 스레드의 개념
- 멀티 스레드

스레드의 개념

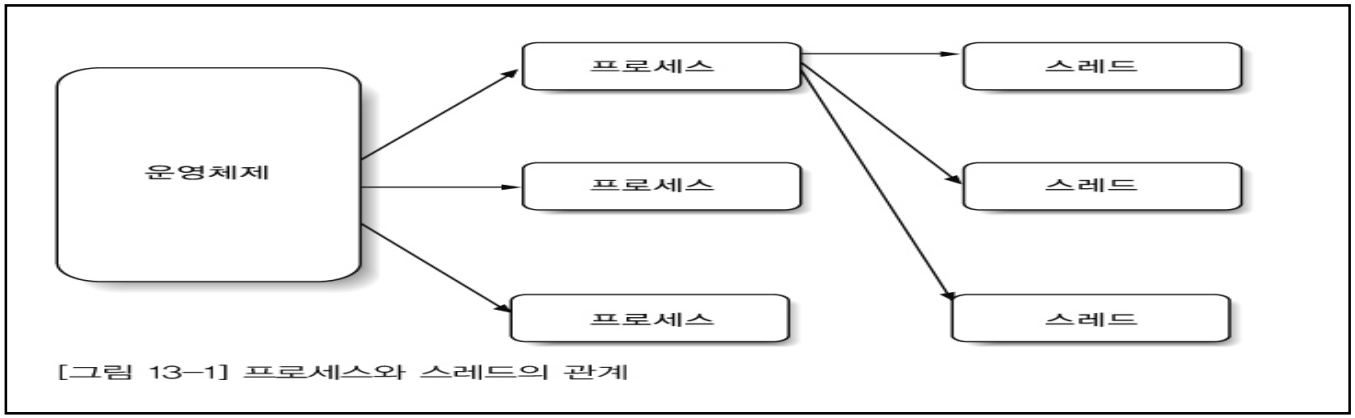
❖ 프로세스(process)란?

- 프로세스(process)란 단순히 실행 중인 프로그램(program)이라고 할 수 있습니다.
- 즉, 사용자가 작성한 프로그램이 운영체제에 의해 메모리 공간을 할당 받아 실행 중인 것을 말합니다.
- 이러한 프로세스는 프로그램에 사용되는 데이터와 메모리 등의 자원 그리고 스레드로 구성 됩니다.

스레드의 개념

❖ 스레드(thread)란?

- 스레드(thread)란 프로세스(process) 내에서 실제로 작업을 수행하는 주체를 의미합니다.
- 모든 프로세스에는 한 개 이상의 스레드가 존재하여 작업을 수행합니다.
- 또한, 두 개 이상의 스레드를 가지는 프로세스를 멀티스레드 프로세스(multi-threaded process)라고 합니다.



[그림 13-1] 프로세스와 스레드의 관계

스레드의 생성과 실행

- Thread 클래스를 상속하는 방법
  - start() : 스레드를 시작하는 메소드
  - run() : 실제 작업을 수행하는 메소드
  - sleep(time) : 주어진 시간 만큼 실행 흐름을 중지 시키는 메소드
- Runnable 인터페이스를 구현하는 방법
  - run()메소드를 반드시 재정의 해야 함.
  - 재정의 한 클래스를 Thread 클래스의 생성자 매개변수로 하여 Thread 객체를 생성해서 사용.
- 스레드의 종류
  - 독립 스레드
  - 데몬 스레드

스레드의 생성과 실행

```
class ThreadWithClass extends Thread {
  public void run() {
    System.out.println(getName()); // 현재 실행 중인 스레드의 이름을 반환함.
  }
}
class ThreadWithRunnable implements Runnable {
  public void run() {
    // 현재 실행 중인 스레드의 이름을 반환함.
    System.out.println(Thread.currentThread().getName());
  }
}
public class Thread01 {
  public static void main(String[] args){
    // Thread 클래스를 상속받는 방법
    ThreadWithClass thread1 = new ThreadWithClass();
    // Runnable 인터페이스를 구현하는 방법
    Thread thread2 = new Thread(new ThreadWithRunnable());

    thread1.start(); // 스레드의 실행
    thread2.start(); // 스레드의 실행
  }
}
```

Thread 클래스

❖ Thread 클래스의 생성자

[표 13-1] Thread 클래스의 주요 생성자

생성자	설명
Thread( )	가장 일반적인 형태의 생성자다. 이 생성자를 이용해서 Thread 객체를 생성하게 되면 Thread의 이름은 'Thread-' +n의 형태가 된다.
Thread(Runnable target)	Runnable 객체를 이용해서 Thread 객체를 생성할 수 있는 생성자다.
Thread(Runnable target, String name)	Runnable 객체를 이용해서 Thread 객체를 생성할 수 있는 생성자며, 스레드의 이름을 지정할 수 있는 생성자다.
Thread(String name)	스레드의 이름을 지정하면서 Thread 객체를 생성할 수 있는 생성자다.

Thread 클래스

❖ Thread 클래스의 주요 메서드

[표 13-2] Thread 클래스의 주요 메서드

반환형	메서드	설명
static void	sleep(long millis)	millis에 지정된 시간만큼 대기한다.
String	getName( )	스레드의 이름을 반환한다.
void	setName(String name)	스레드의 이름을 반환한다.
	start( )	스레드를 시작시킨다.
int	getPriority( )	스레드의 우선순위를 반환한다.
void	setPriority(int newPriority)	스레드의 우선순위를 지정한다.
	join( )	현재 스레드는 join( ) 메서드를 호출한 스레드가 종료할 때까지 기다리게 된다.
static void	yield( )	수행중인 스레드 중 우선순위가 같은 다른 스레드에게 제어권을 넘긴다.
static Thread	currentThread( )	현재 수행되는 스레드 객체를 리턴한다.

Thread 상속으로 스레드 만들기

- Thread를 상속받는 방법(상속을 통해서 스레드를 만드는 방법)

```
public class SimpleThread extends Thread{
    public void run(){
        // 스레드 내의 작업
        for(int i=0; i<50; i++) System.out.print(i+"\t");
    }
}
```

- 스레드 생성 및 동작시키기

```
SimpleThread thrd = new SimpleThread();
thrd.start();
```

- Thread를 상속하는 두 개의 스레드를 생성한 후 실행시키기(Thread 상속)

```
public class SimpleThread Main{
    public static void main(String[] args){
        System.out.println("프로그램 시작");

        SimpleThread thrd1 = new SimpleThread();
        SimpleThread thrd2 = new SimpleThread();

        thrd1.start();
        thrd2.start();

        System.out.println("프로그램 종료");
    }
}
```

Runnable로 스레드 만들기

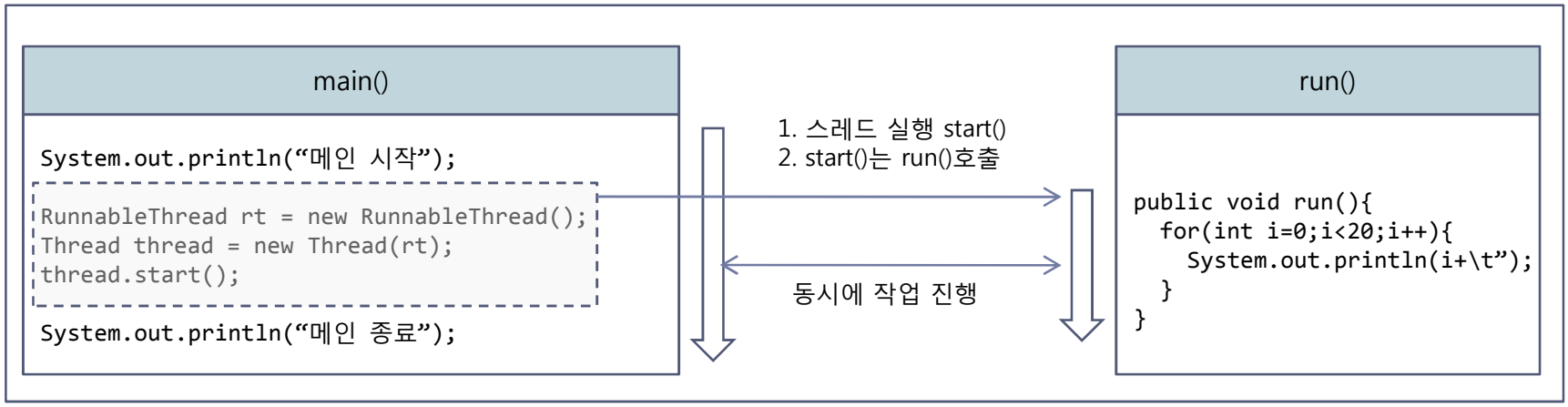
```
public interface Runnable{
  void run();
}
```

```
// Runnable 인터페이스의 구현
public class RunnableThread
  implements Runnable{
  public void run(){
    // 작업내용
    for(int i = 0; i < 20; i++){
      System.out.println(i + "\t");
    }
  }
}
```

```
// 스레드 실행
public class RunnableThreadMain {
  public static void main(String args[]){
    System.out.println("메인 시작");
    // Runnable 구현객체 생성
    RunnableThread rt = new RunnableThread();

    // Thread 객체 생성시 패러미터로 전달
    Thread thread = new Thread(rt);

    // 스레드 실행
    thread.start();
    System.out.println("메인 종료");
  }
}
```



스레드간의 커뮤니케이션(동기화)

❖ Critical section의 동기화 문제

- 동기화 : 공유자원을 상대로 순서대로 작업이 이루어지는 것을 동기화가 보장된다고 한다.
- Critical section : 스레드들이 공유하는 영역(데이터)
- 동기화(Synchronized) 블럭
- 동기화(Synchronized) 메소드

❖ 스레드의 제어를 위한 도구

- setPriority() : 스레드가 Run 상태에 들어갈 수 있는 우선권을 결정.
- sleep() : 일정시간 동안 작업을 멈추게 한다.
- wait() : 스레드를 대기상태(NotRunnable)로 보낸다.
- notify() : 대기상태에 있는 스레드를 Runnable 상태로 복귀시켜서 작업을 재개하게 한다.
- notifyAll() : 대기상태에 있는 모든 스레드를 Runnable 상태로 복귀시켜서 작업을 재개하게 한다.