Thread Ⅰ - extends Thread, implements Runnable

쓰레드의 이해와 생성

Java Program
↑↓
JVM
↑↓
OS

Process

- **OS로 부터 메모리공간을 할당받은 현재 실행 중인 프로그램 (ing)**
- **OS 관점에서 메모리 공간을 할당받아 프로그램 실행흐름을 이루는 하나의 독립적인 프로그램을 의미**

- OS 관점에서 JVM도 하나의 Process이다.
- JVM 위에서 동작하는 프로그램도 Process이다.

Thread

- **하나의 Process 내에서의 독립적인 프로그램 실행흐름**
- **Process는 Thread를 담는 그릇이다.**

- **Main Thread : Process 내에 기본적으로 만들어지는 Thread**
  JVM이 Java Program을 실행시키면 Process의 생성과 동시에 그 안에 하나의 Thread가 생성
  바로 이 Thread가 실행하는 Method가 main Method이고, 이 Thread가 Main Thread이다.

- **Java Process의 Memory 구조**

Method Area	Method Code들이 존재
Stack Area	**Method의 호출을 위한 Memory 공간**
Heap Area	모든 Instance는 Heap에 생성

  - **Main Thread의 실행을 위해서는 Stack 영역이 필요하다.**
  - **Stack 하나가 존재한다는 것은 하나의 실행흐름이 존재하는 것**
    (Thread가 호출하는 Method를 순서대로 Stack에 쌓고, 처리 후 Stack이 호출된 Method들의 메모리공간을 반환한다.)
  - **Thread 간에 Stack 영역은 공유가 불가능하고, 각각의 Thread 별로 Stack이 독립으로 존재해야 한다.**
  - **Thread 간에 Heap 영역은 공유가 가능하다. (참조변수로 Instance의 참조가 가능하므로)**

Method Area

↑↓ ↑↓ ↑↓
Thread A Stack **Thread B Stack** **Thread C Stack**
↑↓ ↑↓ ↑↓

Heap Area

쓰레드 생성방법 Ⅰ - extends Thread

- **Thread Class를 상속받은 Thread 용 Class를 디자인해서 Thread가 할 일을 정의.**
- **이러한 Class를 Thread Class라고 한다.**

```
public class ShowThread extends Thread {
    String threadName;

    public ShowThread(String name){
```

```
        threadName = name;
    }

    // run() Method는 Thread의 Main Method이다. Thread의 시작지점
    @Override
    public void run(){
        for(int i=0; i<100; i++){
            System.out.println("안녕하세요. "+threadName+"입니다. ");
            try{
                Thread.sleep(100);
            }catch(InterruptedException e){
                e.printStackTrace();
            }
        }
    }
}

public class ThreadMain {
    public static void main(String[] args) {
        // 1.Thread Instance 생성
        ShowThread thread1 = new ShowThread("예쁜 쓰레드");
        ShowThread thread2 = new ShowThread("멋진 쓰레드");
```

start( ) Method가 호출이 되면 JVM이 Process 내에 Thread를 생성하고, Thread에 Stack 영역이 할당되고 Stack 영역에 run( ) Method를 호출해준다.

```
        // 2.Thread Instance의 start() Method를 호출하여 Thread 시작
        thread1.start();
        thread2.start();
    }
}
```

쓰레드의 생성을 보인 첫번째 예제의 문제점 1

Q. 쓰레드의 인스턴스를 생성하고 나서, start( ) Method를 호출하면 run( ) Method가 실행되는데, run( ) Method를 직접 호출하면 안되나?

A. run( ) Method를 직접 호출하는 것도 불가능한 일은 아니지만 단! 이 경우에는 단순 Method의 호출에 지나지 않는다.
JVM은 start( ) Method를 통해 메모리 공간(Stack)의 할당과 쓰레드의 실행을 위한 기반을 마련한 다음에 run( ) Method를 대신 호출해준다.
이는 개발자가 직접 main Method를 호출하지 않는 것과 같은 이치이다.

Q. CPU는 하나인데, 어떻게 둘 이상의 쓰레드가 동시에 실행 가능한가?

A. 모든 쓰레드는 CPU를 공유하고 CPU의 자원을 나누어 쓴다. (다수의 Process를 매우 빠른 속도로 스위칭하며 동시 실행)

Q. main Method가 종료되어도 쓰레드는 계속 실행할까?

A. main Method가 종료되면 Main Thread가 종료되는데, 하나의 쓰레드가 종료되더라도, 다른 Thread에 영향을 미치지 않는다.
그러므로 Main Thread 종료시점에서 다른 Thread가 있었다면 Process는 종료되지 않고, 다른 Thread의 실행을 돕는다.
(Process 안의 모든 Thread가 종료되어야 Process가 종료된다, 그리고 run( ) Method가 종료되어야 Thread가 종료된다.)

Q. 쓰레드는 실행흐름인데, 정확히 쓰레드는 무엇인가?

A. Thread Class를 설계하고 Thread Instance의 run( ) 메소드를 실행하는데, Instance가 쓰레드인가?
쓰레드는 JVM에 의해서 실행흐름을 위해 준비되는 모든 것들의 총칭(Stack, Resource, 쓰레드 관련 작업들..)이다.
Thread Instance는 쓰레드를 대표하는 것이다.

쓰레드 생성방법 II - implements Runnable

- Java는 다중 상속을 허용하지 않는다.
- 그러므로 상속이 된 Class를 대상으로 Thread Class로 정의하기 위해 Runnable Interface를 Implements 시켜준다.

```
class Sum {
     int num;
     public Sum(){ num=0; }
     public void addNum(int n){ num += n; }
     public int getNum(){ return num; }
}

class AdderThread extends Sum implements Runnable {
     int start, end;
     int threadNo;

     public AdderThread (int no, int s, int e){
          start = s;
          end = e;
          threadNo = no;
     }

     @Override
     public void run() {
          for(int i=start; i<=end; i++){
               addNum(i);

               try {
                    Thread.sleep(500);
               } catch (InterruptedException e) {
                    e.printStackTrace();
               }

               System.out.println("Thread No : "+threadNo+" num : "+num);
          }
     }
}

class RunnableMain {
     public static void main(String[] args) {
```

쓰레드의 두 가지 조건
i. Thread Instance의 생성
ii. run() Method의 호출

```
          AdderThread at1 = new AdderThread(1, 1, 50); // Thread Instance는 아니다.
          AdderThread at2 = new AdderThread(2, 51, 100);

          Thread thread1 = new Thread(at1);
          // 인자로 at1을 전달했으므로 at1의 run() Method를 호출할 여건이 마련되었다.
          // 본질적으로 extends Thread나 implements Runnable의 동작차이는 없다.

          Thread thread2 = new Thread(at2);
          tr1.start();
          tr2.start();

          try{
               thread1.join();
               thread2.join();
          }catch(InterruptedException e){
               e.printStackTrace();
          }

          System.out.println("1~100까지의 합 : "+(at1.getNum()+at2.getNum()));
     }
}
```