JAVA Thread

프로그램은 사람이 이해하는 코드를 작성.

느려도 꾸준하면 경기에서 이긴다.

작성자 : 홍효상
이메일 : hyomee@naver.com
소스 : https://github.com/hyomee/JAVA_EDU

Content

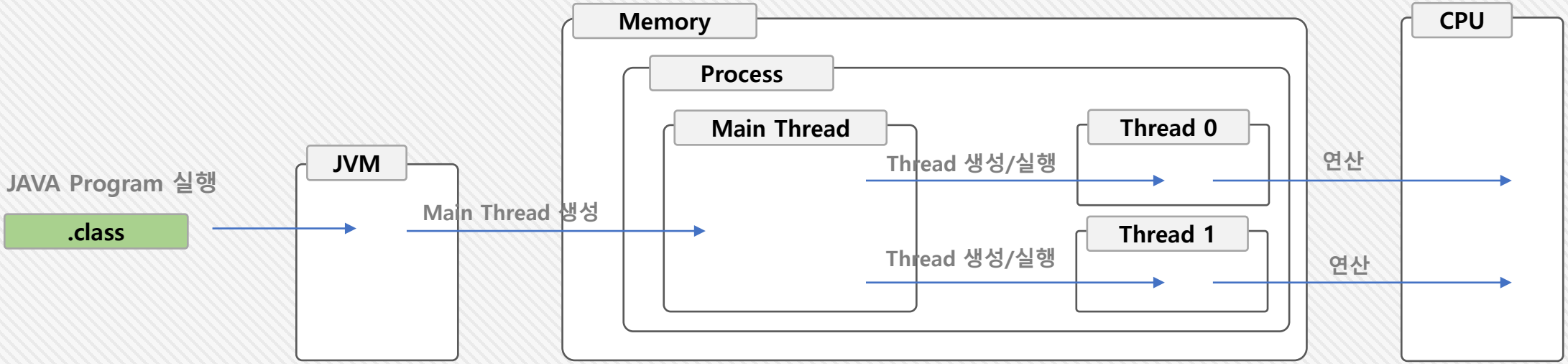
15. Thread

1. Thread 이란

" 프로세스는 최소 하나의 Thread를 가지고 있고 CPU를 사용 하는 최소 단위 "

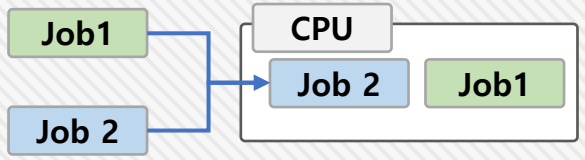
자바에서 Thread

- 자바로 작성된 프로그램이 시작 될 때 Main Thread 1개가 실행된다.
- 1개 이상의 Thread가 동시에 실행 되는 것을 Multi Thread Process라 한다.

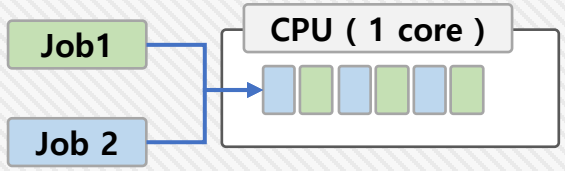


동시성과 병렬성

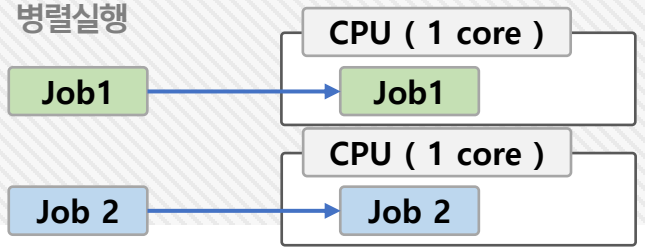
- 순차 실행



- 동시실행



- 병렬실행



1. Thread 생성 및 실행

01. Thread 상속 -> run 재정의

```
public class MythreadA extends Thread{
  @Override
  public void run() {
    System.out.println("MythreadA Start : " + currentThread().getId());
    try {
      Thread.sleep(1000);
    } catch (InterruptedException e) {
      e.printStackTrace();
    }
    System.out.println("MythreadA End : "+ currentThread().getId());
  }
}
```

```
MythreadA myThreadA = new MythreadA();
myThreadA.start();
```

새로운 Thread 관리 그리고 실행

02. Thread 상속 -> run 재정의

```
public class MyThreadRunnable implements Runnable{
  @Override
  public void run() {
    System.out.println("MyThreadRunable Start : " + Thread.currentThread().getId());
    try {
      Thread.sleep(1000);
    } catch (InterruptedException e) {
      e.printStackTrace();
    }
    System.out.println("MyThreadRunable End : "+ Thread.currentThread().getId());
  }
}
```

```
// Runable을 상속 받은 객체
MyThreadRunnable myThreadRunnable = new MyThreadRunnable();
myThreadRunnable.run();

// Runable를 상속 받으면 start가 없어서 Thread 객체 생성 후 start
Runnable myRunnable = new MyThreadRunnable();
Thread myThread = new Thread(myRunnable);
myThread.start();
```

1. Thread 생성 및 실행

03. InnerClass 사용

```
Thread myInnerThread = new Thread(new Runnable() {
  @Override
  public void run() {
    System.out.println("myInnerThread Start : " + Thread.currentThread().getId());
    try {
      Thread.sleep(1000);
    } catch (InterruptedException e) {
      e.printStackTrace();
    }
    System.out.println("myInnerThread End : "+ Thread.currentThread().getId());
  }
});
myInnerThread.start();
```

```
myInnerThread.setPriority(10);
myInnerThread.setDaemon(true);
myInnerThread.start();
```

Thread 시작 전에 설정 하여야 함

04. Thread 속성

- isDaemon(): 데몬, 일반 인지 확인
  - main thread 가 종료 되면 자동 종료 됨 : 데몬 으로 설정 (setDaemon(true))
- setPriority : 우선 순위 지정 1 ~ 10, 10 가장 높음, 5: 지정 하지 않으면 default
- getId() : Thread id 얻음
- getName : Thread Name 얻음
  - 설정 하지 않으면 자동 부여
  - setName() 으로 이름 설정 함

```
static void ThreadView( Thread t, Class obj, String startEnd) {
  Timestamp timestamp = new Timestamp(System.currentTimeMillis());
  System.out.println(String.format("[%s][%s][%s] %s\t" +
                   "Thread Name\t : %S\n" +
                   "-- Thread Id\t : %S\n" +
                   "-- Thread Priority\t : %S\n",
          timestamp.toString(),
          obj.getName().toString(),
          t.isDaemon()?"데몬":"일반",
          startEnd,
          t.getName(),
          t.getId(),
          t.getPriority()));
}
```

1. Thread 동기화

" 하나의 작업이 끝나고 다음 작업 수행 "

동기화

- 동기화(synchronized): 하나의 작업이 끝나고 다음 작업 수행
- 비동기화(asynchronized) : 하나의 작업 결과 관계 없이 다음 작업 수행

SyncTest

```
void add() {
  int num = this.num;
  try {
    Thread.sleep(1000);
  } catch (InterruptedException e) {
    e.printStackTrace();
  }
  this.num = num + 1;
}
```

```
class MyThread extends Thread {
  SyncTest syncTest;
  MyThread(SyncTest syncTest) {
    this.syncTest = syncTest;
  }

  @Override
  public void run() {
    this.syncTest.add();
    System.out.println(Thread.currentThread().getName()
            + ".." + this.syncTest.num);
  }
}
```

공유 객체

```
public class syncMain {
  public static void main(String... args) {
    SyncTest syncTest = new SyncTest();
    MyThread myA = new MyThread(syncTest);
    myA.start();

    MyThread myB = new MyThread(syncTest);
    myB.start();
    System.out.println(myA.currentThread().getName()
            + ".." + myA.getState());
  }
}
```

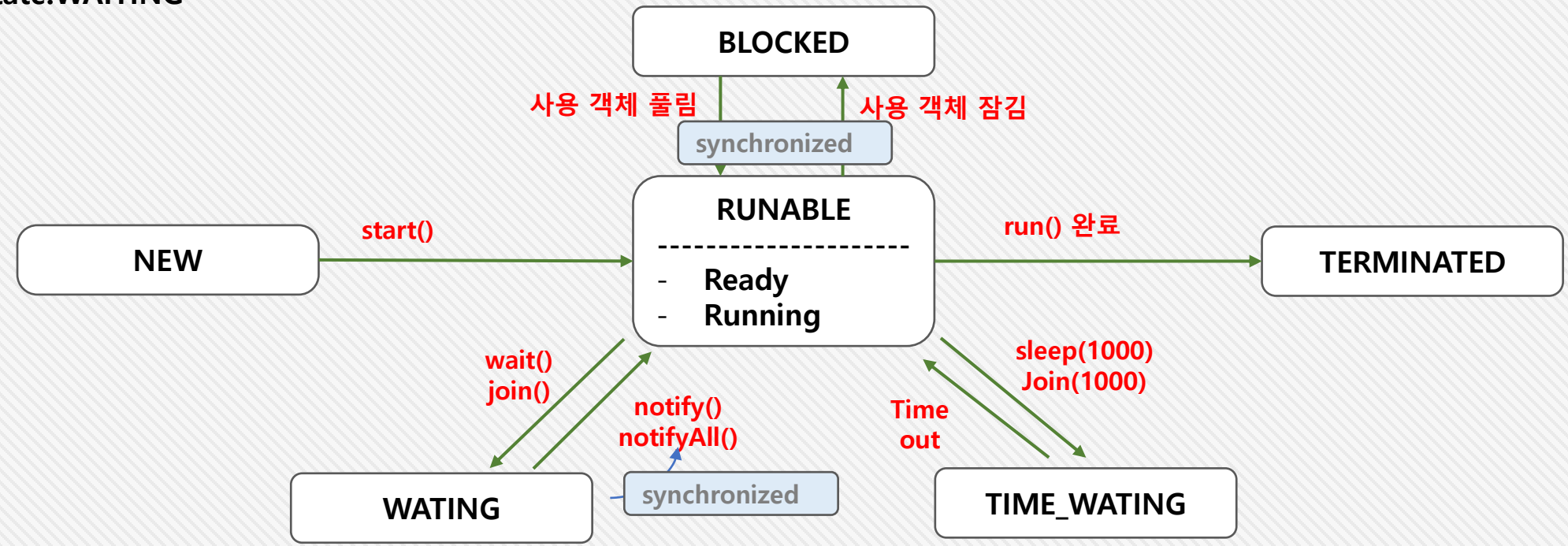
```
main..RUNNABLE
Thread-1..2
Thread-0..2
```

- Thread 1 이 실행 중인 상태에서 Thread 2가 수행 되어 공유 객체의 데이터에 변함이 없다
  -> Thread 1 실행이 끝나 날 때 가지 대기 .

1. Thread 상태

java.lang.Thread.State

- ThreadState.BLOCKED
- ThreadState.NEW
- ThreadState.RUNNABLE
- ThreadState.TERMINATED
- ThreadState.TIMED_WAITING
- ThreadState.WAITING



1. Thread 상태


```java
class BasicThread implements Runnable {
  @Override
  public void run() {
    Thread thread = Thread.currentThread();
    try{

      System.out.println("5초 동안 BasicThread Sleep");
      thread.sleep(5000);
      synchronized (thread) {
        thread.wait();
      }
    }catch(InterruptedException iException){
      iException.printStackTrace();
    }
  }
}
```

```java
public class StateMain {
  public static void main(String... args) {
    // BasicThread 생성
    Thread threadInstance=new Thread(new BasicThread());
    threadInstance.start();
    System.out.println("BasicThread State: "
            + threadInstance.getState());
    try {
      boolean keepRunning=true;
      int count=1;
      while(keepRunning) {
        Thread.sleep(2000);
        System.out.println(count*2+ " 초 경과 - BasicThread State: "
                + threadInstance.getState());
        count++;
        if(count==4){
          //  6 초 경과
          synchronized(threadInstance) {
            threadInstance.notify();
          }
        }
        if(Thread.State.TERMINATED == threadInstance.getState()){
          keepRunning = false;
        }
      }
    }catch(InterruptedException iException){
      iException.printStackTrace();
    }
  }
}
```

```
5초 동안 BasicThread Sleep
BasicThread State: RUNNABLE
2 초 경과 - BasicThread State: TIMED_WAITING
4 초 경과 - BasicThread State: TIMED_WAITING
6 초 경과 - BasicThread State: WAITING
8 초 경과 - BasicThread State: TERMINATED
```