

프로세스 동기화

예제: 자바 쓰레드

- 맥 만들기

- `java.lang.Thread`
- 주요 메소드
 - `public void run()` // 새로운 맥이 흐르는 곳 (치환)
 - `void start()` // 쓰레드 시작 요청
 - `void join()` // 쓰레드가 마치기를 기다림
 - `static void sleep()` // 쓰레드 잠자기

- 자바를 이용해 멀티 쓰레드 프로그램을 구현해 볼 수 있다!

java.lang.Thread

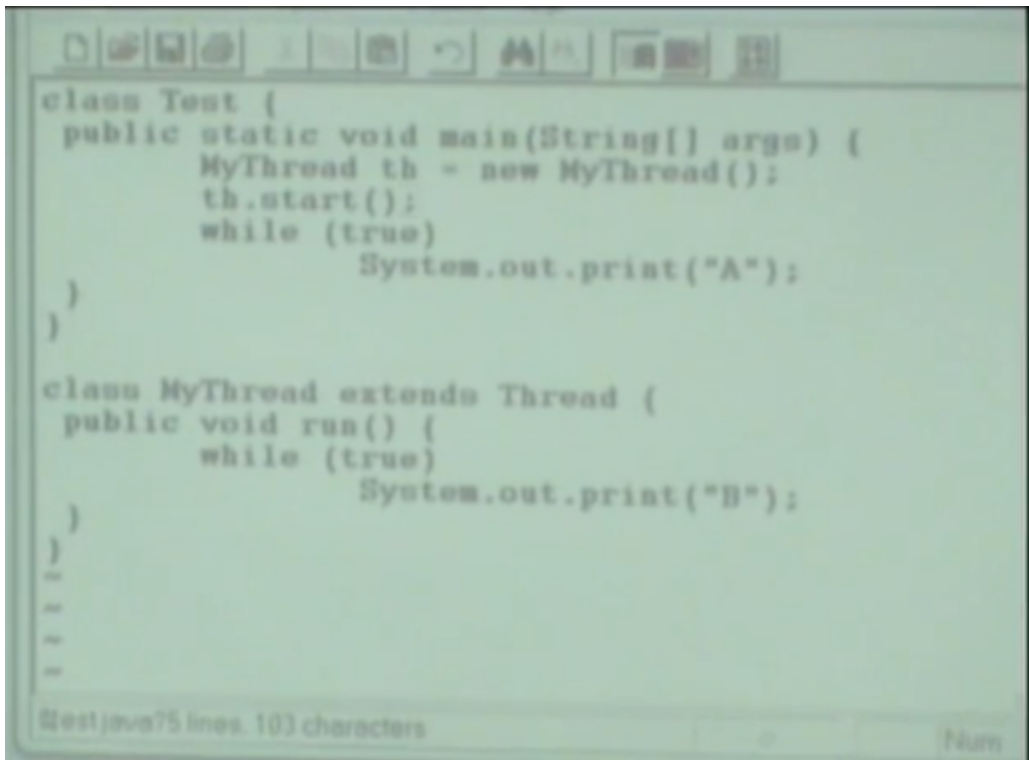
- `Thread.run()`

- 쓰레드가 시작되면 `run()` 메소드가 실행된다
⇒ `run()` 메소드를 치환한다.

```
class MyThread extends Thread {  
    public void run() {           // 치환 (override)  
        // 코드  
    }  
}
```

- 예제: 글자 A 와 B 를 동시에 화면에 출력하기

- 모든 프로그램은 처음부터 1개의 쓰레드는 갖고 있다 (main)
- 2개의 쓰레드: main + MyThread



```
class Test {
    public static void main(String[] args) {
        MyThread th = new MyThread();
        th.start();
        while (true)
            System.out.print("A");
    }
}

class MyThread extends Thread {
    public void run() {
        while (true)
            System.out.print("B");
    }
}
~
~
~
~

Test.java 75 lines, 103 characters
```

무한루프 두개를 각자 스레드에게 줄건데, 이렇게 적으면 A, B 가 동시에(화면엔 번갈아가며) 무한히 찍혀 나옴.

생각해보면 파이썬은 무한루프 걸리면강 거기서 뺏아서 하나의 철자만 출력이 될텐데? 이렇게 다중스레드 프로그램이라는것.

프로세스 동기화 (Process Synchronization)

일반적으로 프로세스가 데이터에 접근하거나 하는 경우를 생각해 보면, 데이터가 저장돼 있는 위치에서 데이터를 읽어와 연산을 하고, 연산 결과를 본래 위치에 저장하는 사이클이 있는데?

뭐 그냥 읽기만 할거면 아무 문제가 없겠지만, 연산 하고 수정 결과를 저장한다고 생각해보면 아주 찰나의 시간으로도 누가 먼저 연산하고 넣느냐에 따라서 충돌이 일어나거나 꼬여버릴 수 있음.

(기차표 예매 문제 = 나도 예약했는데 알고보니 같은 자리 주인이 하나 더 생겨버렸다?)

=> 동기화를 통해 해결하자!

프로세스 동기화

- Process Synchronization

- cf. Thread synchronization

- Processes

- Independent vs. Cooperating
- Cooperating process: one that *can affect or be affected by* other processes executed in the system
- 프로세스간 통신: 전자우편, 파일 전송
- 프로세스간 자원 공유: 메모리 상의 자료들, 데이터베이스 등
- 명절 기차표 예약, 대학 온라인 수강신청, 실시간 주식거래

사실 현대 운영체제는 프로세스 단위로 스위칭 되는게 아니고 프로세스 안에 쓰레드 단위로 스위칭 되니까, **쓰레드 동기화**가 더 맞는 표현일 수 있음.

일단은 근데, 프로세스 단위라고 생각해 보면 프로세스에는 두가지 타입이 있음.

- 1 Independent = 다른 프로세스가 하는 일이란 아무 관련이 없음. 그냥 지 혼자 자기 할 거 함.
- 2 Cooperating = 다른 프로세스랑 협력해서 뭘 해야 함. (다른 프로세스에 영향을 주거나 받는 프로세스.)

둘중 2 타입이 더 많음.

DB(Common Resource)생각해보면, (기차표의 예제) => 명절에 기차표 예매할때를 생각해보면 엄청 어려움.

왜냐? 하나의 서버에 수많은 사용자 프로세스들이 (손님들의 프로그램) DB(실제 그 철도회사가 어떤 시간에 기차표를 편성해 뒀는지와 티켓 가격의 정보 등)에 사용(접근)하려고 하기 때문에.

그럼 이 손님들의 프로세스는 2 타입이라고 볼 수 있는데, 한명이 뭐 성공하면 다른쪽은 실패하거나 이런식으로 영향이 있기 때문임.

프로세스 동기화

- Process Synchronization

- Concurrent access to shared data may result in data inconsistency
- *Orderly execution of cooperating processes so that data consistency is maintained*

- Example: BankAccount Problem (은행계좌문제)

- 부모님은 은행계좌에 입금; 자녀는 출금
- 입금(deposit)과 출금(withdraw)은 독립적으로 일어난다.

동시에 공유자원에 대해 접근하게 되면 데이터 일관성을 해칠 수 있게 된다.

그러므로, 프로세스 동기화는 서로 영향을 주고 받는 프로세스들 간에 순서를 지켜 실행 되도록 하 게끔 하여 데이터 일관성을 유지시키는 작업이다.

- 은행 계좌문제

둘이 우연히 같은 시점에 입출금을 하게 된다면? 문제가 생길 수 있으니까.

<pre>class Test { public static void main(String[] args) throws InterruptedException { BankAccount b = new BankAccount(); Parent p = new Parent(b); Child c = new Child(b); p.start(); c.start(); p.join(); c.join(); System.out.println("Wnbalance = " + b.getBalance()); } }</pre>	<pre>class BankAccount { int balance; void deposit(int amount) { balance = balance + amount; } void withdraw(int amount) { balance = balance - amount; } int getBalance() { return balance; } }</pre>
<pre>class Parent extends Thread { BankAccount b; Parent(BankAccount b) { this.b = b; } public void run() { for (int i=0; i<100; i++) b.deposit(1000); } }</pre>	<pre>class Child extends Thread { BankAccount b; Child(BankAccount b) { this.b = b; } public void run() { for (int i=0; i<100; i++) b.withdraw(1000); } }</pre>

- 1 우선 BankAccount 클래스를 하나 짚는다. 은행 계좌의 속성은 잔액이고, 은행 업무는 3개로 나눌 수 있음. 출금, 입금, 잔액 확인
- 2 부모, 자식 쓰레드 만들. 부모는 입금만 하고 애는 출금만 함 (100번 1000원씩)

3 main 에서 각자 인스턴스들 생성해주고 start() ! + 어차피 parent child 모두 파라미터로 은행계좌 b 인스턴스 애를 공유함.

그럼 이 코드에서 프로세스 동기화 처리는 대체 어딴냐? join() 이거임.

입금이나 출금 끝날때까지 기다려라!

그래서 이거 컴파일 하고, 실행해보면 깔끔하게 0 나옴. (1000입금 출금 막 일어나다가)

근데 좀 신기한건 뭐 찰나의 차이같은게 있는건지 막 입금 주르륵 일어나다가 출금 주르륵 일어나도 이게 꼭 번갈아 가는건 아닌데?

=> 운영체제가 자체적으로 스위칭 하는 텀 같은게 있어서 그럼