Synchronization I - synchronized Method

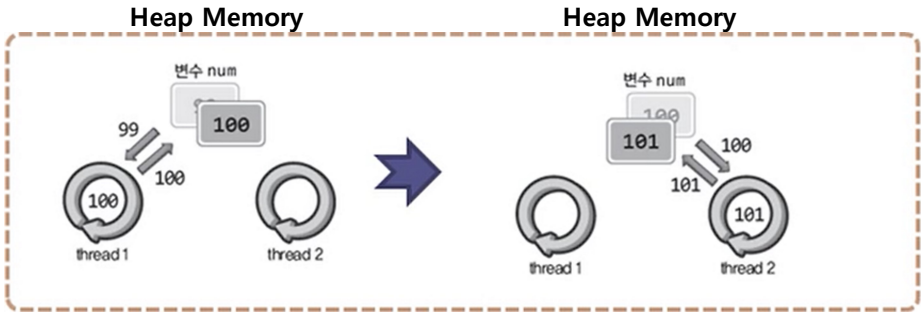
동기화 (Synchronization)

- Heap Memory 영역에 두 쓰레드가 동시접근 X
- 쓰레드의 Memory 영역 접근 순서를 컨트롤 하기
  (동기화는 어감 상 동시에 무언가를 하는 것 같이 느낌을 받지만,
  동기화는 순서를 정해서 메모리 공간에 접근하는 방법이다.)

- 쓰레드의 메모리 동시접근은 CPU의 Core 개수와는 상관이 없다.

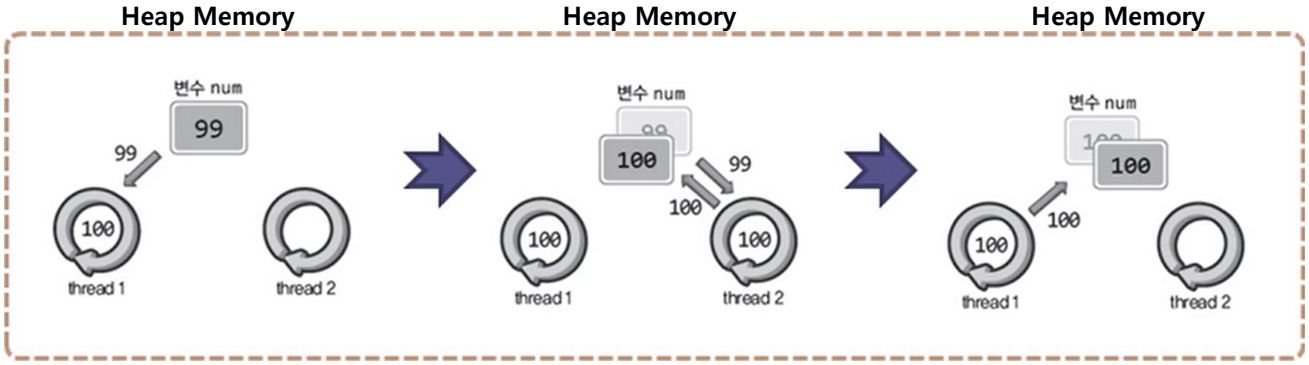
쓰레드의 메모리 접근방식과 그에 따른 문제점

i. 정상적인 결과를 보이는 연산의 예 (이상적인 Case)



- 쓰레드는 CPU를 통해서 메모리공간에 접근 → 메모리공간에 저장된 데이터를 꺼낸 후 →
  그 데이터로 연산 → 연산결과 데이터를 다시 메모리 공간에 저장.

ii. 비정상적인 연산을 보이는 연산의 예 (문제의 발생확률이 높은 Case)



- 실제로는 매우 고속으로 두 쓰레드가 공유한 CPU를 번갈아 쓴다. 따라서, 하나의 연산이 끝나야지만
  다른 쓰레드에게 CPU의 사용을 넘기는 것은 아니다.
- 위의 경우에는 Thread1은 99를 가져다가 100으로 증가 후 다시 메모리에 가져다 놓아야 연산이 끝나는데,
  하나의 연산 중간에도 CPU가 다른 쓰레드로 넘어갈 수 있다.

- 쓰레드의 동시접근 : 하나의 연산이 끝나고 다른 연산이 진행되어야 하는데, 서로 다른 연산들이
  연산 진행 과정 중에서 계속 CPU의 할당을 스위칭한다.

	연산시작	연산과정1	연산과정2	연산과정3	연산종료

쓰레드 1 연산	CPU사용	CPU사용		CPU사용	
쓰레드 2 연산			CPU사용		CPU사용

- 이 문제를 해결하기 위해서는 결국 하나의 연산이 실행되고 끝날 때까지 CPU의 사용을 보장해주어야 한다.

  (쓰레드가 하나의 작업을 완전히 끝낼 때 까지, CPU의 할당을 다른 쓰레드에게 넘기지 않는다! = 동기화!)

Thread-Safe 합니까?

Note that this implementation not synchronized

API 문서에는 해당 Class의 Instance, Method가 둘 이상의 쓰레드가 동시접근을 해도
문제가 발생하는지 않는지를 명시하고 있다.
따라서, 반드시 쓰레드 기반의 프로그래밍을 한다면 특정 Class의 사용에 앞서 쓰레드에 안전한지 확인해야 한다.

쓰레드의 동기화 기법1 - synchronized 기반 동기화 메소드

- 제일 간단한 방법이긴 하나 그에 따른 대가를 지불해야 한다 - 성능저하!
  (직접 실행시켜 보세요. 스카이레이크 Core i5인데도 불구하고 엄청 느립니다.....)

- 아래의 예제는 여러 쓰레드가 하나의 inc 인스턴스의 Method를 호출하고 있고,
  Method는 변수 num에 접근하고 있다.
  (Memory 공간의 동시접근이 발생할 수 있다.)

- 그러므로 synchronized 선언을 통해 여러 쓰레드가 Method를 동시에 실행하지 않도록 막는다!
  해당 Method를 호출한 순서대로 쓰레드가 작업을 하고 끝날 때까지, 다른 쓰레드는 Method의 실행을 기다린다.
  (이 때, 리소스 소모가 심하므로 성능이 대폭 저하)

```
public class Increment {
	private int num = 0;

	// synchronized 선언을 통해 여러 쓰레드가 Method를 동시에 실행하지 않도록 막는다!
	   동기화 처리, 쓰레드에 안전!
	public synchronized void increment(){ num++; }

	public int getNum(){ return num; }
}

public class IncThread extends Thread {
	Increment inc;

	public IncThread(Increment inc){
		this.inc = inc;
	}

	@Override
	public void run(){
```

```
          for(int i=0; i<10000; i++)
               for(int j=0; j<10000; j++)
                    inc.increment();
     }
}

public class ThreadSyncMethod {
     public static void main(String[] args) {
          Increment inc = new Increment();
          IncThread it1 = new IncThread(inc);
          IncThread it2 = new IncThread(inc);
          IncThread it3 = new IncThread(inc);

          it1.start();
          it2.start();
          it3.start();

          try{
               it1.join();
               it2.join();
               it3.join();
          }catch(InterruptedException e){ e.printStackTrace(); }

          System.out.println(inc.getNum());
     }
}
```