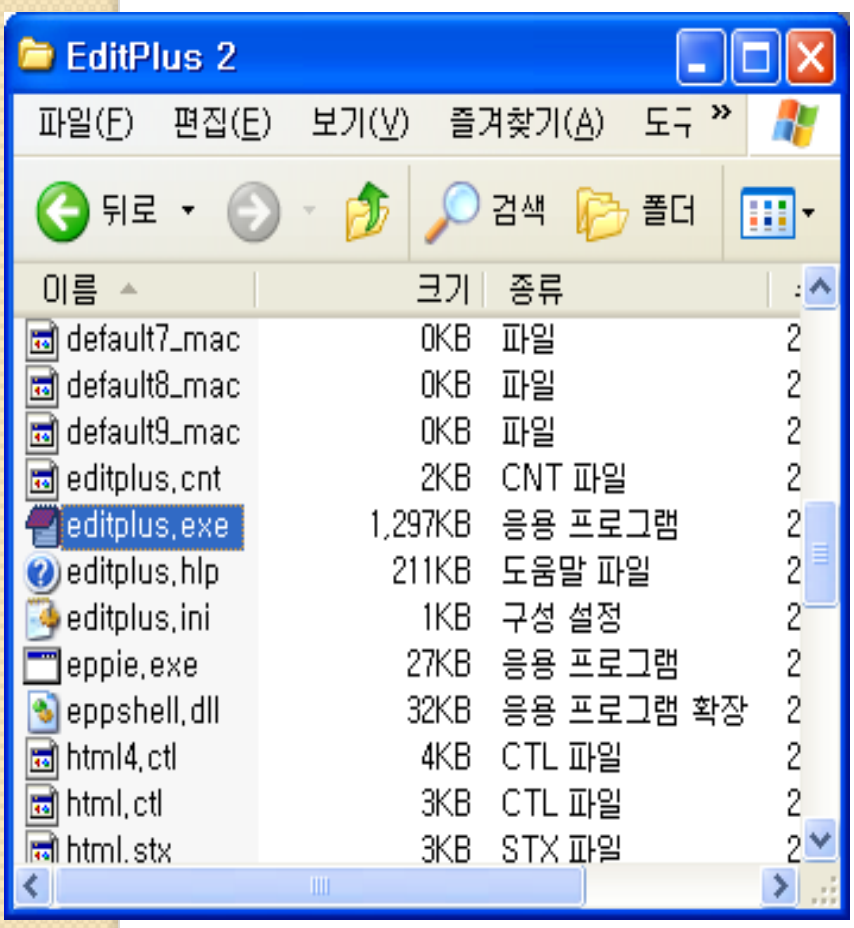
12. Thread

강사 강태광

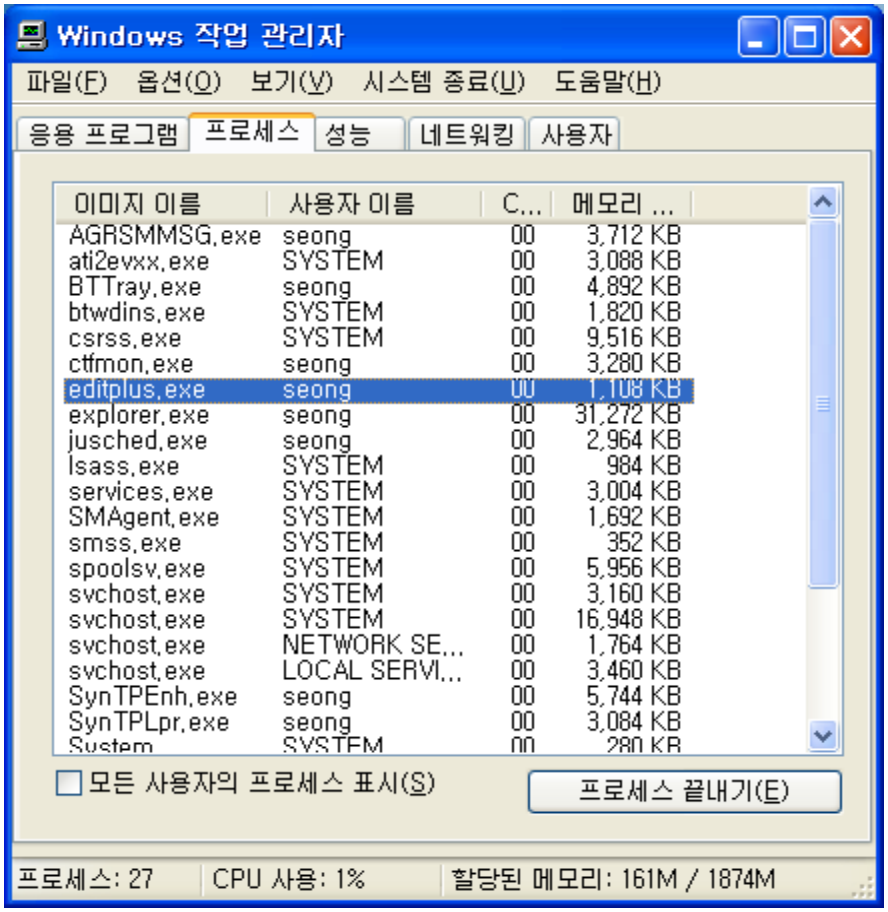
I] 프로세스와 쓰레드(process & thread)

프로그램 —실행→ 프로세스

▶ 프로그램 : 실행 가능한 파일(HDD)



▶ 프로세스 : 실행 중인 프로그램(메모리)

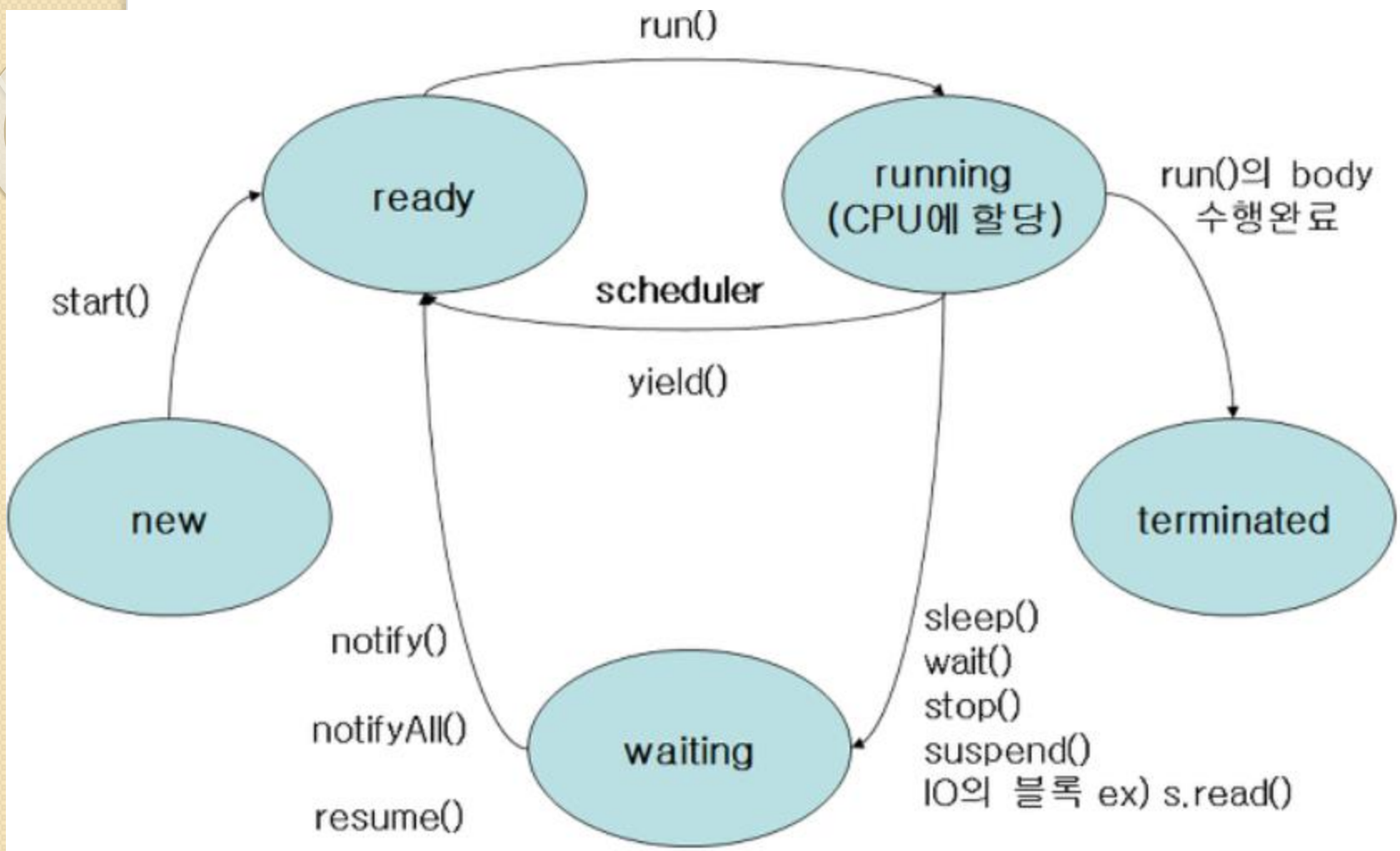


I] 프로세스와 쓰레드(process & thread)

▶ 스레드란
- 하나의 프로그램이 하나이상의 독립적 서브 Task로 분리되어 실행될때 각 서브 Task를 Thread
- 각 스레드는 별개로 실행하면서 CPU를 공동으로 사용
- 경량(lightweight) 프로세스
- 다중 스레드
  . 하나의 프로세스에서는 동시에 여러 쓰레드가 실행
  . 하나의 새로운 프로세스를 생성하는 것보다 하나의 새로운 쓰레드를 생성하는 것이 더 적은 비용이 든다

▶ 프로세스 : 실행 중인 프로그램, 자원(resources)과 쓰레드로 구성

2] 쓰레드(Thread)의 Life-Cycle



2] 멀티쓰레드의 장단점

- 멀티 쓰레드는 프로세스를 복사하는것이 아니라 Function 단위로 실행시키는 것이다.
- 즉 어떤 Function에서 다른 Function을 호출하면서 그 밑의 루틴을 다시 실행하는 것이다.
- 이 것은 같은 프로세스이므로 멀티 프로세스에 비해서 부하를 덜준다.
- 그래서 경량프로세스란 말을 하는 것이다

장점	– 자원을 보다 **효율적**으로 사용할 수 있다. – 사용자에 대한 **응답성(responseness)**이 **향상**된다. – 작업이 분리되어 코드가 **간결**해 진다. **"여러 모로 좋다."**
단점	– 동기화(synchronization)에 주의해야 한다. – 교착상태(dead-lock)가 발생하지 않도록 주의해야 한다. – 각 쓰레드가 효율적으로 고르게 실행될 수 있게 해야 한다. **"프로그래밍할 때 고려해야 할 사항들이 많다."**

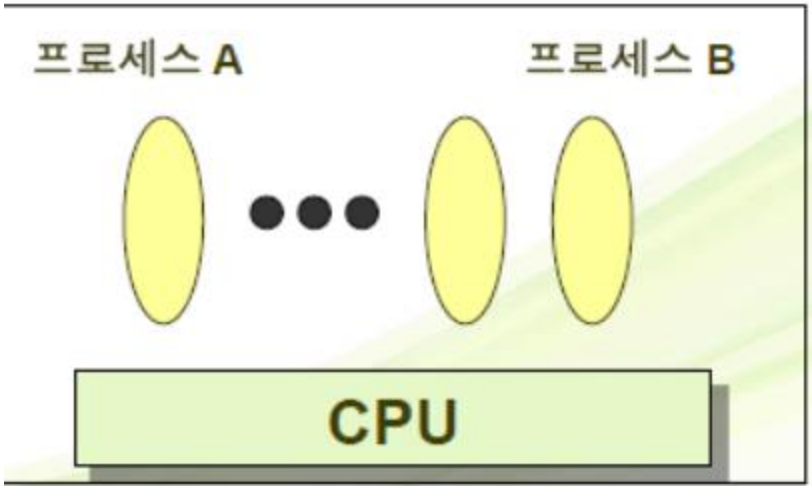
4] 멀티태스킹(Multi-Tasking)

▶멀티 태스킹
- 컴퓨터에서 각각의 일을 전담하는 해당 프로그램들**(여러 개의 프로세스 (Process))이 동시에 실행되는 것.**

- 멀티 태스킹을 위해 운영체제가 취하는 방식이 멀티 프로세스 시스템.

▶스케 줄링**(Scheduling)**

- 멀티프로세스 시스템에서 각 프로세스는 동시에 실행되는것 처럼 보이지만, **CPU가 하나이기 때문에 실행시간을 잘게 나누어 프로세스가 돌아가는데 각** 프로세스들이 돌아가면서 **CPU를 점유하는데 프로세스간 CPU점유작업을** 말함.



4] Thread 클래스

- JDK는 스레드를 지원하는 java.lang.Thread 클래스 제공
- 스레드를 생성하기 위해 사용
-4개의 생성자를 제공

- Thread()
- Thread(String s)
- Thread(Runnable r)
- Thread(Runnable r, String s)

4]Thread 클래스의 메서드

void sleep(long msec)	msec에 지정된 밀리초(milliseconds) 동안 대기
void sleep(long msec, int nsec) throws InterruptedException	msec에 지정된 밀리초+nsec에 지정된 나노초(nanoseconds) 동안 대기
String getName()	스레드의 이름을 반환
void setName(String s)	스레드의 이름을 s로 설정
void start()	스레드를 시작시킨다. run() 메소드를 호출
int getPriority()	스레드의 우선 순위를 반환
void setPriority(int p)	스레드의 우선 순위를 p값으로 설정
boolean isAlive()	스레드가 시작되었고 아직 끝나지 않았으면 true를 그렇지 않으면 false를 반환
void join()	스레드가 끝날 때까지 대기
void run()	스레드가 실행할 부분을 기술하는 메소드. 하위 클래스에서 오버라이딩 되어야 한다
void suspend()	스레드가 일시 정지된다. resume()에 의해 다시 시작될 수 있다.
void resume()	일시 정지된 스레드를 다시 시작시킨다

4] 스레드의 생성 및 이용

스레드를 생성하는 2가지 방법
1]Thread 클래스로부터 직접 상속받아 스레드를 생성
2]Runnable 인터페이스를 사용하는 방법
(현재의 클래스가 이미 다른 클래스로부터 상속 받고 있는 경우)

스레드 이용
1] Thread 클래스를 상속받은 뒤, 해당 스레드에서 지원하고 싶은 코드를 run()메서드에서 오버라이딩 해준다.
2] 해당 스레드를 생성한 후, 스레드 객체의 start()메서드를 호출한다.

```
class ThreadA extends Thread { // Thread 클래스로부터 상속
.....
public void run() {
.... // 상위 클래스인 Thread 클래스의 run() 메서드를 오버
.... //라 이딩하여 스레드가 수행하여야 하는 문장들을 기술
}
.....
```

```
ThreadA ta = new ThreadA();
ta.start();
// 스레드 객체를 생성하여 스레드를
// 시작시킨다
```

4] 스레드 extends 적용 예시

```
package ch12;
class A1 extends Thread {
	A1(String str) {
	   super(str);
	}
	public void run() {
		for (int i = 1; i <= 100; i ++ ) {
			System.out.print(getName() +i+ " ₩t");
			if ( i%10==0) System.out.println();
			try {
			Thread.sleep(50);
			} catch (InterruptedException e) {
			// TODO Auto-generated catch block
			//e.printStackTrace();
			}
		}
		super.run();
	}
}
public class Threadtest1 {
	public static void main(String[] args) {
		A1 a1 = new A1("subA");
		A1 a2 = new A1("subB");
		a1.run();
		a2.run();
		for (int i = 1; i <= 100; i ++ ) {
			System.out.print("Main " +i+ "₩t");
			if ( i%10==0) System.out.println();
		}
	}

}
```

5] Thread-Runnable 인터페이스

1.현재의 클래스가 이미 다른 클래스로부터 상속을 받고 있다면 Runnable 인터페이스를 사용하여 스레드를 생성할 수 있다
2. Runnable 인터페이스는 JDK에 의해 제공되는 라이브러리 인터페이스이다
3. Runnable 인터페이스에는 run() 메소드만 정의되어 있다

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

▶ Runnable 인터페이스 이용

1] 이미 Applet 클래스로부터 상속을 받고 있으므로 인터페이스 이용

```
class RunnableB extends Applet implements Runnable {
.....
        public void run() {
        ..... // Runnable 인터페이스에 정의된 run() 메소드를
        ..... // 오버라이딩하여 스레드가 수행할 문장들을 기술한다
        }
.....
}
```

5] Thread-Runnable 인터페이스 이용

1.Runnable 인터페이스를 이용하여 스레드를 생성하는 방법

1) RunnableB rb = new RunnableB(); // 객체 rb 생성
Thread tb = new Thread(rb);
// rb를 매개변수로 하여 스레드 객체 tb를 생성
tb.start(); // 스레드 시작

2) RunnableB rb = new RunnableB();
new Thread(rb).start(); // 스레드 객체를 생성하여 바로 시작

5] Thread-Runnable 인터페이스 이용 적용 예시

```
package ch12;

class RunnableExam01  implements Runnable {
	public void run( ){
		int a=0;
		System.out.println(">> run 메소드 진입 <<");
		while(true){
			try{
				Thread.sleep(200); //0.3초간 대기
			}catch(InterruptedException e){
				e.printStackTrace();
			}
			System.out.println(Thread.currentThread( ).getName( )+" : " + ++a);
			if(a>=5) break;
		}
		System.out.println(">> run 메소드 종료 <<");
	}
}

public class Exam_01 {
	public static void main(String[] args) {
		RunnableExam01  r1=new RunnableExam01();
		RunnableExam01  r2=new RunnableExam01();
		RunnableExam01  r3=new RunnableExam01();
		Thread t1=new Thread(r1, "첫 번째 스레드");
		Thread t2=new Thread(r2, "두 번째 스레드");
		Thread t3=new Thread(r3, "세 번째 스레드");
		t1.start();
		t2.start();
		t3.start();
	}
}
```

5] 스레드의 제어 I

▶ 스레드의 상태

1) 실행(Run): run()메소드 내의코드를 실행하고 있는 상태
2) 대기: 실행을 기다리고 있는 상태
- 대기(Waiting): 뒤에서 배울 동기화(Synchronized)에 의한 대기상태
- 슬립(**Sleeping): CPU의 점유를 중지하고, 아무것도 안하고 있는 상태**
- 일시멈춤(**Suspended): 일시중지 상태**
- 지연((**Blocked): 입출력 메서드 등의 호출로 인해서, 메서드의 종료가** 일어날 때까지 스레드가 대기하고 있는 상태
3) 준비(Ready): 실행상태에 들어가기 위해, 준비하고 있는 상태.
스레드 스케줄러에 의해 선택된 스레드가 실행 상태에 들어가게 된다.
4) 종료(Done): run() 메서드나 stop()메서드에 의해 스레드의 실행이 완료된 상태

5] 스레드의 제어2

▶메서드 호출에 의한 스레드 제어

1) start() 메서드: 스레드는 준비상태에 들어간다
2) run()메서드: 스레드가 살아있는 동안 계속해서 실행하는 코드를 갖고 있으며, 스레드가 실행상태에 있는동안 수행된다.
3) yield()메서드: 다른 스레드에게 실행상태를 양보하고, 자신은 대기상태로 바꾸는 스레드
4) sleeping()메서드: 실행상태에서 슬립상태로 들어감.
   - 보통 슬립메서드는 만분의 일초 단위의 슬립 시간을 지정하는데, 이 시간동안에 스레드는 아무동작도 하지 않으며, 시간이 지나면, 스레드는 준비상태로 들어가게 된다.

5] 스레드의 제어3

▶메서드 호출에 의한 스레드 제어

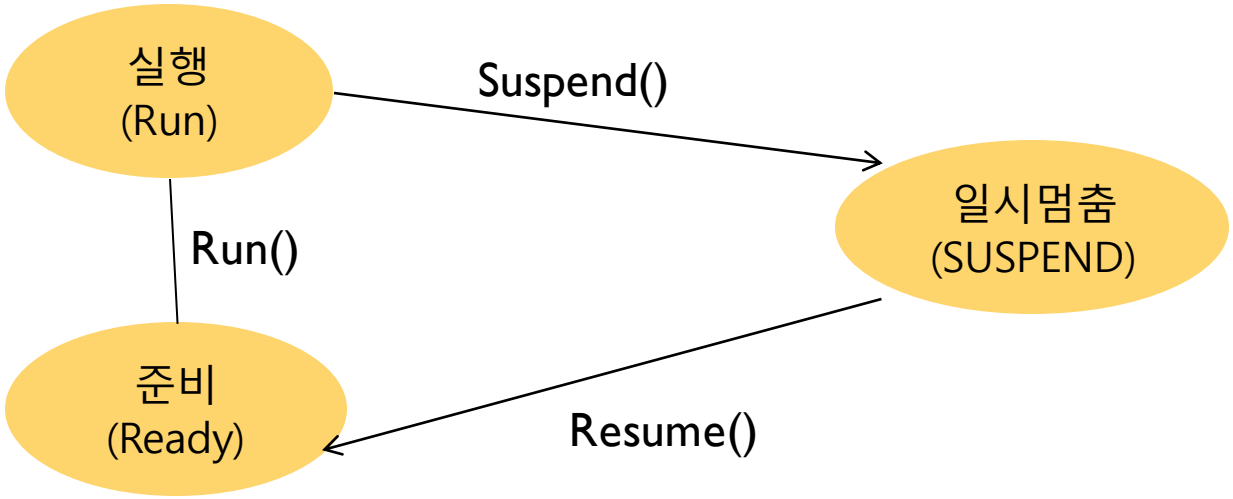
1) Suspend()와 resume()메서드: 스레드의 실행을 잠시 중단하고, 다시 실행시키는 제어를 하는데 사용된다.

① Suspend()메서드: 일시 멈춤상태로 들어가게 된다.

② resume()메서드 : 스레드가 준비상태로 들어가서 스레드 스케줄러의 선택을 기다리게 된다.

2) stop()메서드: 실행상태에서 강제로 종료 상태 스레드 바꾼다.
이때 스레드는 실행이 중지되고, 종료된 스레드가 된다.

- 스레드가 종료된 후에는 해당 스레드를 절대로 다시 실행시킬수 없다.
다시 실행시키려면, 스레드를 새로 생성하는 수밖에는 없다.



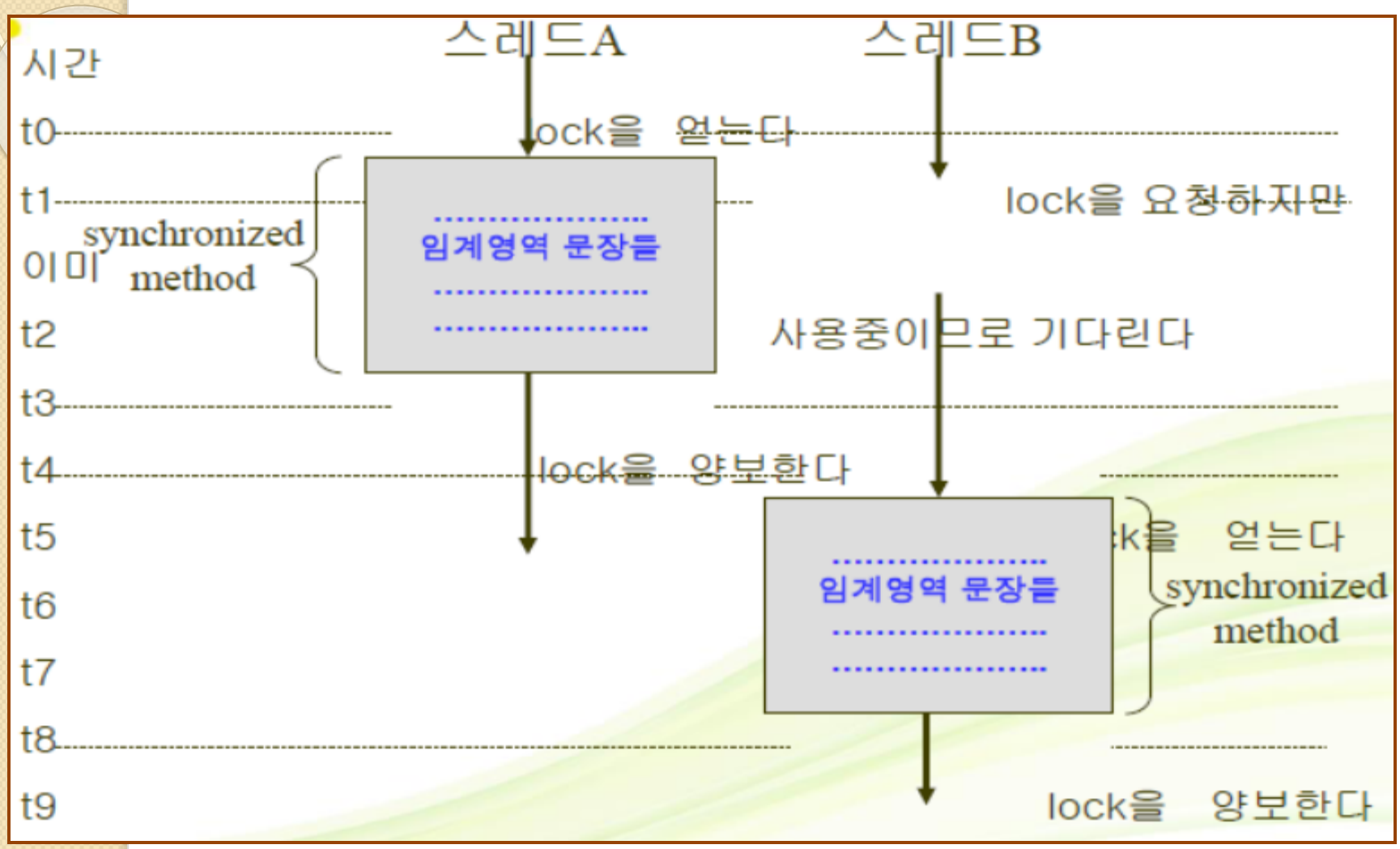
5] 동기화(Synchronization)

1. 동기화(Synchronization)
 - 하나의 자원을 여러 태스크가 사용하려 할 때에, 한 시점에서 하나의 태스크만이 사용할 수 있도록 하는 것
- 대부분의 응용 프로그램에서 다수개의 스레드가 공유할 수 있는 부분이 요구된다
- 공유부분은 상호 배타적으로 사용되어야 한다
- 임계영역(critical section)
① 상호배타적으로 사용되는 공유부분
② 자바는 한 순간에 하나의 스레드만 실행할 수 있는 synchronized method 제공
③ 한 스레드가 synchronized method를 수행 중이면 다른 스레드는 대기한다

동기화 예 - 좌석예매 시스템

네트워크를 이용하여 여러매표소에서 좌석이 예매될 때, 매표소들간에 좌석 예매시스템의 좌석 정보를 동시에 접근할 수 있다면, 같은 좌석을 여러 사람에게 중복되게 예매할 수도 있다.
따라서, 이렇게 여러 태스크들이 하나의 정보에 접근 할때에는 , 한번에 하나씩 의 태스크만이 접근 할 수 있도록 해야 한다.

6] synchronized method 수행



6] synchronized method – 적용 예시

```
package ch12;

public class Account {
   int total;
   Account (int total) {
   this.total = total;
   }

   synchronized void deposit(int amt , String name) {
   total += amt;
   System.out.println(name + " 예금 : " + amt);
   }
   // synchronized 사용시 동시 접근 막는 다.
   synchronized void withdraw(int amt, String name) {
   if (amt <= total) {
   total -= amt;
   System.out.println(name + " 출금 : " + amt);
   getTotal();
   } else {
   System.out.println(name +"의 출금 요청" + amt+"잔액부족 저금좀 해 --> 현재 잔액:" + total);
   }

   }
   void getTotal() {
   System.out.println("잔액 : " + total);
   }
}
```