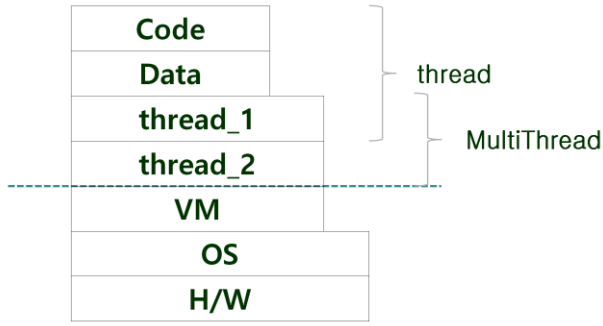
Thread

- Thread 구조
- Thread API
- Thread 프로그래밍
- Thread 상태도
- Thread 동작

14-1

Thread 구조

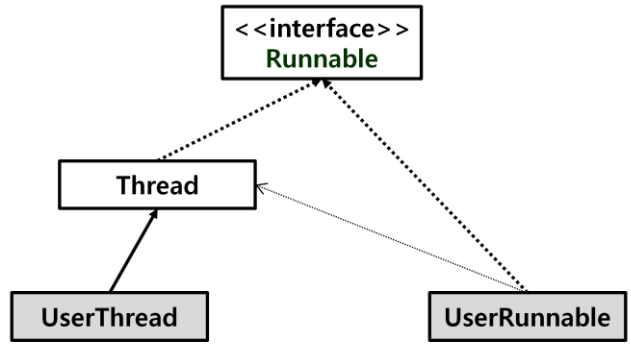
- MultiProcess
- MultiTasking
- MultiThread



14-2

Thread API

- Thread 객체를 작성하고 위해서는 Thread 클래스를 상속받거나 Runnable 인터페이스를 상속받아 run()를 재정의 한다.
- JVM에 MultiThread로 실행하기 위해서는 Thread 클래스의 start()를 호출해야 한다.



14-3

Thread 프로그래밍

- **Thread 클래스 상속**

```
class UserThread extends Thread{
   public void run() {
      …
    }
}
```

UserThread thread=new UserThread();
thread.start();

14-4

```
class UserThread extends Thread{
    int counter;
    public void run() {
        while(true){
            if(counter>100) break;
            System.out.println(currentThread().toString()+"counter : "+
counter++);
        }//while
    }//run
}
public class UserThreadTest {
    public static void main(String[] args) {
        UserThread t1=new UserThread();
        UserThread t2=new UserThread();
        //t1.run(); t2.run();     //Thread 아님
        t1.start();
        t2.start();  //start()-->VM ---> run()
    }
}
```

Thread 프로그래밍

- **Runnable 인터페이스 상속**

```
class UserRunnable implements Runnable{
   public void run() {
      …
    }
}
```

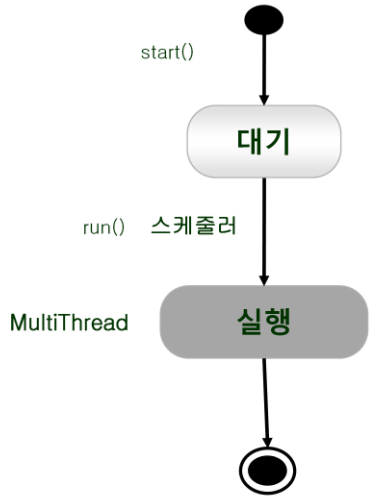
UserRunnable data=new UserRunnable();
Thread thread=new Thread(data);
thread.start();

14-5

```
class UserRunnable implements Runnable {
    int counter;
    public void run() {
        while(true){
            if(counter>100) break;
            System.out.println(Thread.currentThread().toString()+"counter : "+
counter++);
        }//while
    }//run
}
public class UserRunnableTest {
    public static void main(String[] args) {
        UserRunnable data=new UserRunnable();
        Thread t1=new Thread(data);
        Thread t2=new Thread(data)
        t1.start();
        t2.start();  //start()-->VM ---> run()
    }
}
```

Thread 상태도

- Thread

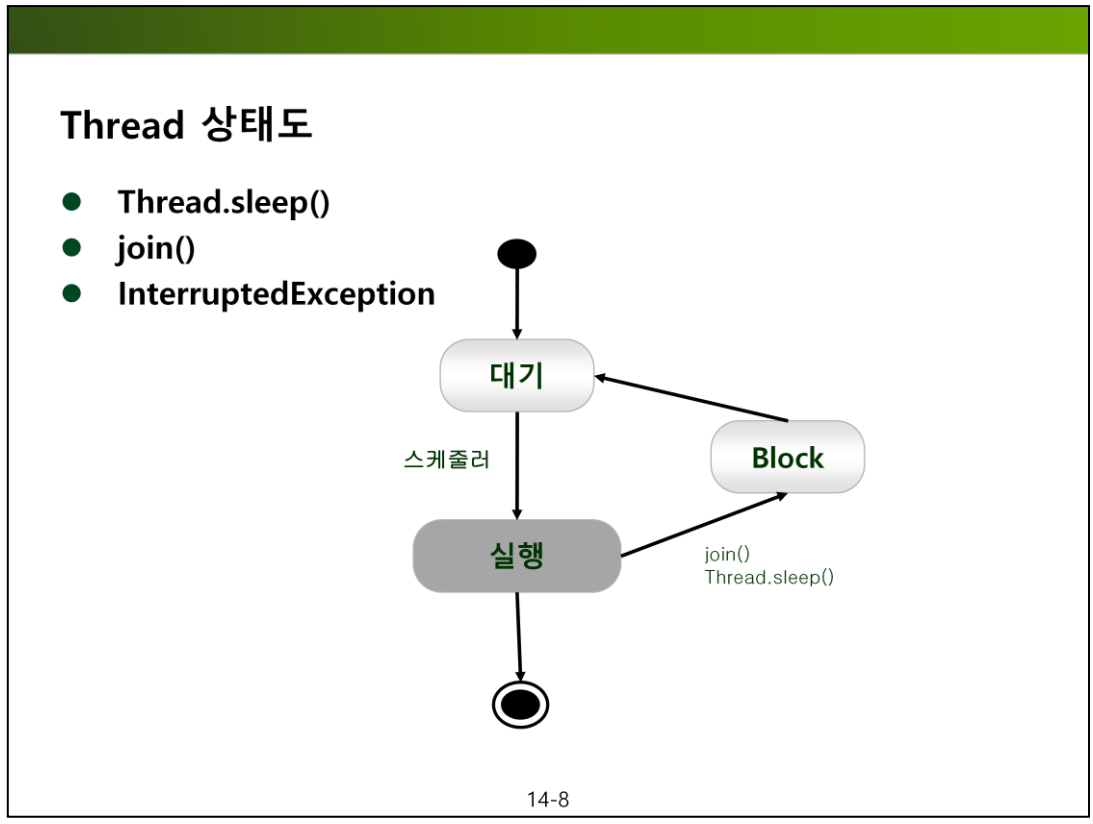


14-6

Thread 프로그래밍

- 주요 속성
  - Thread.MAX_PRIORITY // 10
  - Thread.MIN_PRIORITY // 1
  - Thread.NORM_PRIORITY //5

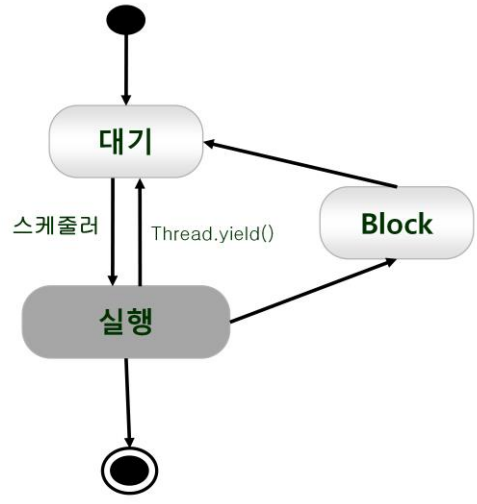
- 주요 메서드
  - start(), sleep(), join(), yield()
  - getName()
  - isAlive()
  - isDaemon()
  - interrupt()
  - currentThread()
  - getPriority()
  - setPriority(int newPriority)



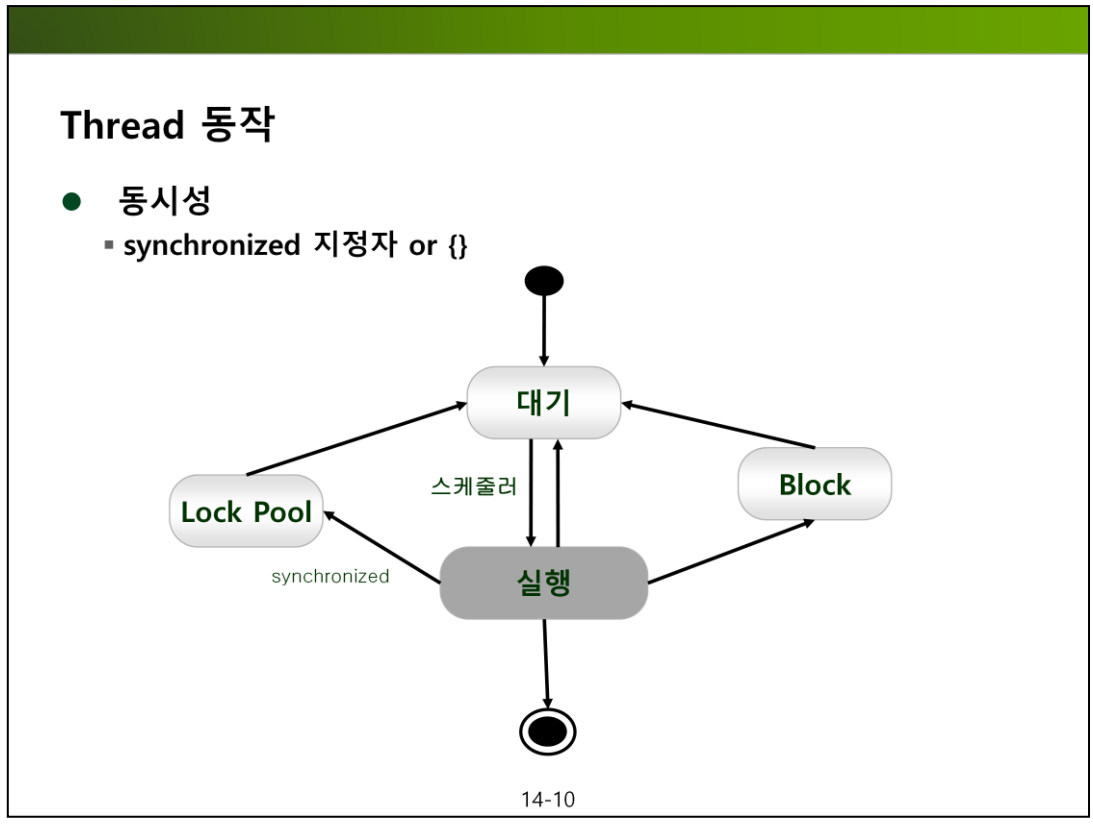

```
try {
    Thread.sleep(new Random().nextInt(2000));
} catch (InterruptedException e) {
     e.printStackTrace();
}
```

Thread 동작

- Thread.yield()



14-9

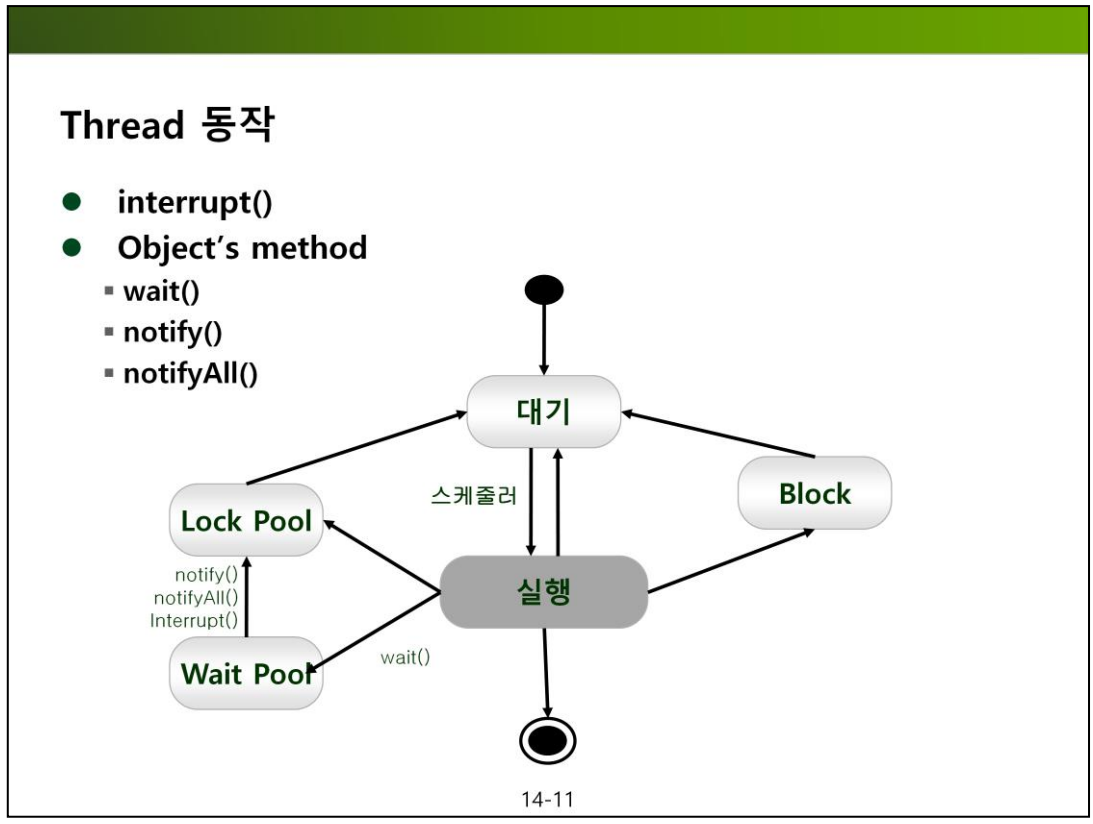


• 메서드 지정자(modifier)

```
public synchronized void methodA(){
   data--;
}
```

• synchronized 블럭

```
public void methodA(){
   synchronized(this){
       data--;
   }
}
```



• wait()

```
public synchronized void getTicket(){
   if(ticket==0) {
        try{
          wait();
        }catch(InterruptedException i){}
    }
   ticket--;
}
```

•notify()

```
public synchronized void addTicket(){
   ticket++;
    notify();
…
}
```

LAB

14-12