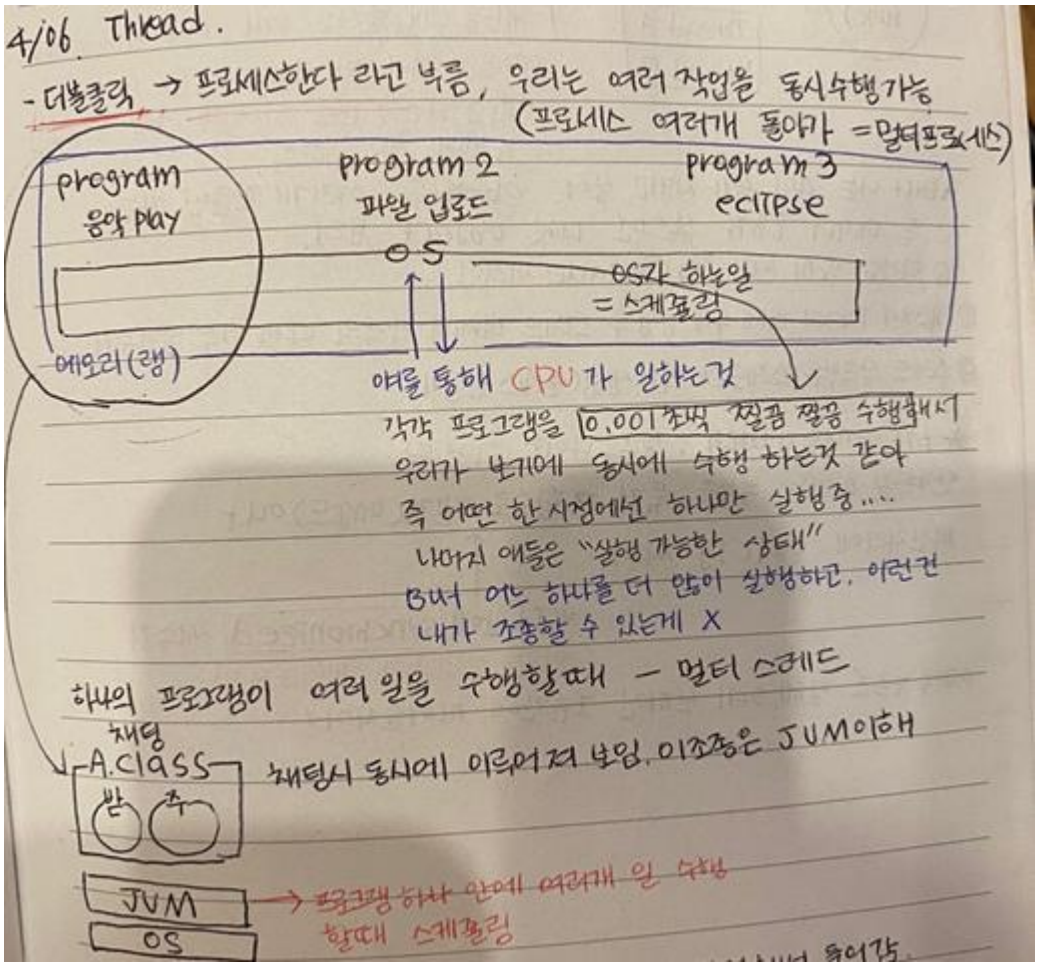
[17]javaThread



[서예린 필기용 노트]

OS 를 통해 CPU가 일한다 각각 프로그램을 완전하게 동시에 실행하는 것이 아니라, 각각 프로그램을 0.001초씩 찔끔찔끔 수행해서 우리가 보기에 동시에 수행하는 것 같아 보이게 하는 것 ! 즉 어떤 한 시점에선 하나만 실행중인 것 . 실행하지 않는 나머지 부분들은 "실행 가능한 상태"로 남아 있는 것 하지만 어느 하나를 더 많이 실행하게 해라 라는 것들은 내가 조절이 불가능함

***하나의 프로그램이 여러 일을 수행하는 것을 멀티 스레드라고 부름**

(멀티스레드의 장점)
자원을 보다 효율적으로 사용할 수 있다.
사용자에 대한 응답성이 향상된다.
작업이 분리되어 코드가 간결해 진다.
(멀티스레드의 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)

(1) Runable 인터페이스 구현을 통한 Target과 Thread

```
// "안녕하세요 10번"하는 target정의
public class TargetEx01 implements Runnable{
        @Override
        public void run() {
                for(int i=0 ;i<10 ; i++) {
                        System.out.println("안녕하세요");
                        try { // 현재 작업을 500밀리세컨(0.5초)동안 대기상태로
                                Thread.sleep(500);
                        } catch (InterruptedException e) {
                                System.out.println(e.getMessage());
                        }
                }//for
        }//run
}

// "반갑습니다" 10번 하는 target정의
public class TargetEx02 implements Runnable{
    @Override
    public void run() {
            for(int i=0 ; i<10 ; i++) {
                    System.out.println("반갑습니다");
                    try {
                            Thread.sleep(500);
                    } catch (InterruptedException e) {
                            System.out.println(e.getMessage());
                    }
            }
    }
}

public class TargetExMain {
    public static void main(String[] args) {
            Runnable target1 = new TargetEx01();
            Runnable target2 = new TargetEx02();
            Thread threadA = new Thread(target1, "A"); // target1의 run()작업을
하는 "A"
            Thread threadB = new Thread(target2, "B"); // target2의 run()작업을
하는 "B"
            threadA.start(); // 쓰레드 실행 시작
            threadB.start(); // 쓰레드 실행 시작
            for(int i=0 ; i<10 ; i++) {
                    System.out.println("나는 main 쓰레드");
                    try {
                            Thread.sleep(500);
```

```
                } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
        }
    }
}
```

(2) Thread 클래스 상속을 통한 Thread

```
// "안녕하세요 10번"하는 target정의
public class TargetEx01 extends Thread{
    @Override
    public void run() {
        for(int i=0 ; i<10 ; i++) {
                System.out.println("안녕하세요");
                try {
                        Thread.sleep(500);
                } catch (InterruptedException e) { }
        }
    }
}
// "반갑습니다 10번"하는 target정의
public class TargetEx02 extends Thread{
    @Override
    public void run() {
        for(int i=0 ; i<10 ; i++) {
                System.out.println("반갑습니다.");
                try {
                        Thread.sleep(500);
                } catch (InterruptedException e) { }
        }
    }
}
public class TargetExMain {
    public static void main(String[] args) {
        Thread threadA = new TargetEx01(); // 쓰레드 생성과 동시에 run()이
이미 정의됨
        threadA.setName("A");
        Thread threadB = new TargetEx02(); // 쓰레드 생성과 동시에 run()이
이미 정의됨
        threadB.setName("B");
        threadA.start();
        threadB.start();
        for(int i=0 ; i<10 ; i++) {

    System.out.println(Thread.currentThread().getName()+"쓰레드");
                try {
                        Thread.sleep(500);
                } catch (InterruptedException e) { }
        }
    }
}
```

(3) Runable 인터페이스 구현을 통한 Target과 Thread (target공유)

```
public class ThreadEx implements Runnable{
    @Override
    public void run() {
            System.out.println(Thread.currentThread().getName());
            System.out.println("ThreadEx");
            for(int i = 0 ; i<10 ; i++) {

    System.out.println(Thread.currentThread().getName()+"쓰레드의 i = "+i);
                    try {
                            Thread.sleep(500);
                    } catch (InterruptedException e) {}
            }
    }
}
public class ThreadExTestMain {
    public static void main(String[] args) {
            Runnable target = new ThreadEx();
            Thread threadA = new Thread(target, "A");
            Thread threadB = new Thread(target, "B");
            threadA.start();
            threadB.start();
            System.out.println(Thread.currentThread().getName());
            System.out.println("main함수 끝");
    }
}
```

(4) Thread 클래스 상속을 통한 thread

```
// ThreadEx threadA = new ThreadEx();
// threaA.setName("A")
// => ThreadEx threadA = new ThreadEx("A");
public class ThreadEx extends Thread{
    public ThreadEx() {}
    public ThreadEx(String name) {
            super(name); // 쓰레드 이름을 명명
    }//생성자
    @Override
    public void run() {
            System.out.println(Thread.currentThread().getName());
            System.out.println("ThreadEx");
            for(int i = 0 ; i<10 ; i++) {

    System.out.println(Thread.currentThread().getName()+"쓰레드의 i = "+i);
                    try {
                            Thread.sleep(500);
                    } catch (InterruptedException e) {}
            }// for
    }
}//class
public class ThreadExTestMain {
    public static void main(String[] args) {
            ThreadEx threadA = new ThreadEx("A");
            ThreadEx threadB = new ThreadEx();
```

```
            threadB.setName("B");
            threadA.start();
            threadB.start();
            System.out.println(Thread.currentThread().getName());
            System.out.println("main 함수 끝");
    }
}
```

(5) 객체1개, 스레드n개

```
public class ThreadEx implements Runnable{
    private int num = 0; //값 공유
    @Override
    public void run() {
            for(int i=0 ; i<10 ; i++) {
                    if(Thread.currentThread().getName().equals("A")) {
                            System.out.println("~ ~ A 수행중 ~ ~");
                            ++num;
                    }
                    System.out.println("ThreadName :
"+Thread.currentThread().getName()+"\t num : "+num);
                    try {
                            Thread.sleep(1500);
                    } catch (InterruptedException e) { }
            }
    }
    public int getNum() {
            return num;
    }
}

public class ExObject1ThreadN {
    public static void main(String[] args) {
            ThreadEx threadEx = new ThreadEx();
            Thread threadA = new Thread(threadEx, "A");
            Thread threadB = new Thread(threadEx, "B");
            threadA.start();
            threadB.start();
            System.out.println("main 함수 쓰레드 :
"+Thread.currentThread().getName());
            try {
                    Thread.sleep(2000);
            } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
            }
            System.out.println("DONE. 이 시점의 num : "+threadEx.getNum());
    }
}
```

(6) 객체n개, 스레드n개

```
public class ThreadEx01 implements Runnable{
    private int num = 0; // 공유변수
    @Override
    public void run() {
```

```
            for(int i = 0 ; i <10 ; i++) {
                    if(Thread.currentThread().getName().equals("A")) {
                            System.out.println("~ ~ A 쓰레드 수행 중 ~ ~");
                            num++;
                    }
                    System.out.println(Thread.currentThread().getName()+"의 num
= "+num);
                    try {
                            Thread.sleep(500);
                    } catch (InterruptedException e) { }
            }//for
    }// run()
}
public class ThreadEx01testMain {
    public static void main(String[] args) {
            Runnable target01 = new ThreadEx01();
            Runnable target02 = new ThreadEx01();
            Thread threadA = new Thread(target01, "A");
            Thread threadB = new Thread(target02, "B");
            threadA.start();
            threadB.start();
            System.out.println("main 함수");
    }
}
public class ThreadEx02 extends Thread{
    private int num = 0;
    @Override
    public void run() {
            for(int i = 0 ; i <10 ; i++) {
                    if(Thread.currentThread().getName().equals("A")) {
                            System.out.println("~ ~ A 쓰레드 수행 중 ~ ~");
                            num++;
                    }
                    System.out.println(Thread.currentThread().getName()+"의 num
= "+num);
                    try {
                            Thread.sleep(500);
                    } catch (InterruptedException e) { }
            }//for
    }//run()
    public int getNum() {
            return num;
    }
}//class
//ThreadEx02 extends Thread
public class ThreadEx02testMain {
    public static void main(String[] args) {
            ThreadEx02 threadA = new ThreadEx02();
            threadA.setName("A");
            ThreadEx02 threadB = new ThreadEx02();
            threadB.setName("B");
            threadA.start();
            threadB.start();
            try {
                    Thread.sleep(7000);
            } catch (InterruptedException e) { }
```

```
            System.out.println("main함수 종료 전 : A의 num =
"+threadA.getNum());
            System.out.println("main함수 종료 전 : B의 num =
"+threadB.getNum());
    }
}
```