제 43강

스레드2

교재: p243~250

목차

복습

스레드란?

: 하나의 프로세스 안 에서 두 가지 이상의 일을 하도록 하는 것

<용어 정리>

- 프로세스(Process) : 실행 중인 프로그램
- 스레드(Thread) : 프로세스에서 작업을 수행하는 것
- 멀티 스레드 프로세스(Multi-Thread Process)
  : 두 가지 이상의 작업을 하는 프로세스

복습

[스레드 생성 방법]

- **자바 스레드로 작동할 작업이 무엇인지 코드로 작성**

1) Thread 클래스 상속
2) Runnable 인터페이스 구현

- **스레드 코드가 실행할 수 있도록 JVM 한테 요청**

1) 인스턴스 생성 후 start( ) 호출
2) 인스턴스 생성 후 Thread( ) 매개변수 생성자에 인자값 전달, start( ) 호출

복습

1) Thread 클래스 상속

```
class T1 extends Thread{
        public void run( ){
                        //작업할 내용
                }
}
```

```
Th1 t1 = new Th1( );
t1. start( );
```

2) Runnable 인터페이스 구현

```
class T2 implements Runnable{
        public void run( ){
                        //작업할 내용
                }
}
```

```
Th2 t2 = new Th2( )
Thread t = new Thread(t2);

t.start( );
```

1. 스레드의 우선순위

스레드는 시분할 방식으로 CPU의 시간을 분배하여 실행하지만
사용자가 직접 스레드의 우선순위를 지정해서 특정 스레드에 더 많은 실행시간 부여 가능

<우선순위 지정 메서드>

void setPriority(int newPriority)	우선 순위 지정
int getPriority( )	우선 순위 반환

- 우선순위는 1 부터 10까지 부여 가능
- 설정하지 않을 경우 기본적으로 5

1. 스레드의 우선순위

<실습> Exam-94.java

우선순위에 따른 스레드의 동작 실습

```
class MyThread3 implements Runnable{
    @Override
    public void run() {
        for(int i=0;i<5;i++) {
            System.out.println("<"+Thread.currentThread().getName()+">");
        }
        for(int i=0;i<1000;i++) {//시간지연
            }
        }
    }
```

```
public class Thread2 {
    public static void main(String[] args) {
        Runnable t =new MyThread3();
        Thread t1= new Thread(t,"thread1");
        t1.setPriority(1); //우선순위 1로 지정

        System.out.println("t1의 우선순위는: "+t1.getPriority());

        Thread t2= new Thread(t,"thread2");
        System.out.println("t2의 우선순위는: "+t2.getPriority());

        Thread t3= new Thread(t,"thread3");
        t3.setPriority(10); //우선순위 10으로 지정
        System.out.println("t3의 우선순위는: "+t3.getPriority());

        t1.start();
        t2.start();
        t3.start();

    }
}
```

<실행 결과>

```
t1의 우선순위는: 1
t2의 우선순위는: 5
t3의 우선순위는: 10
<thread1>
<thread3>
<thread2>
<thread3>
<thread3>
<thread1>
<thread3>
<thread2>
<thread2>
<thread3>
<thread1>
<thread2>
<thread1>
<thread2>
<thread1>
```

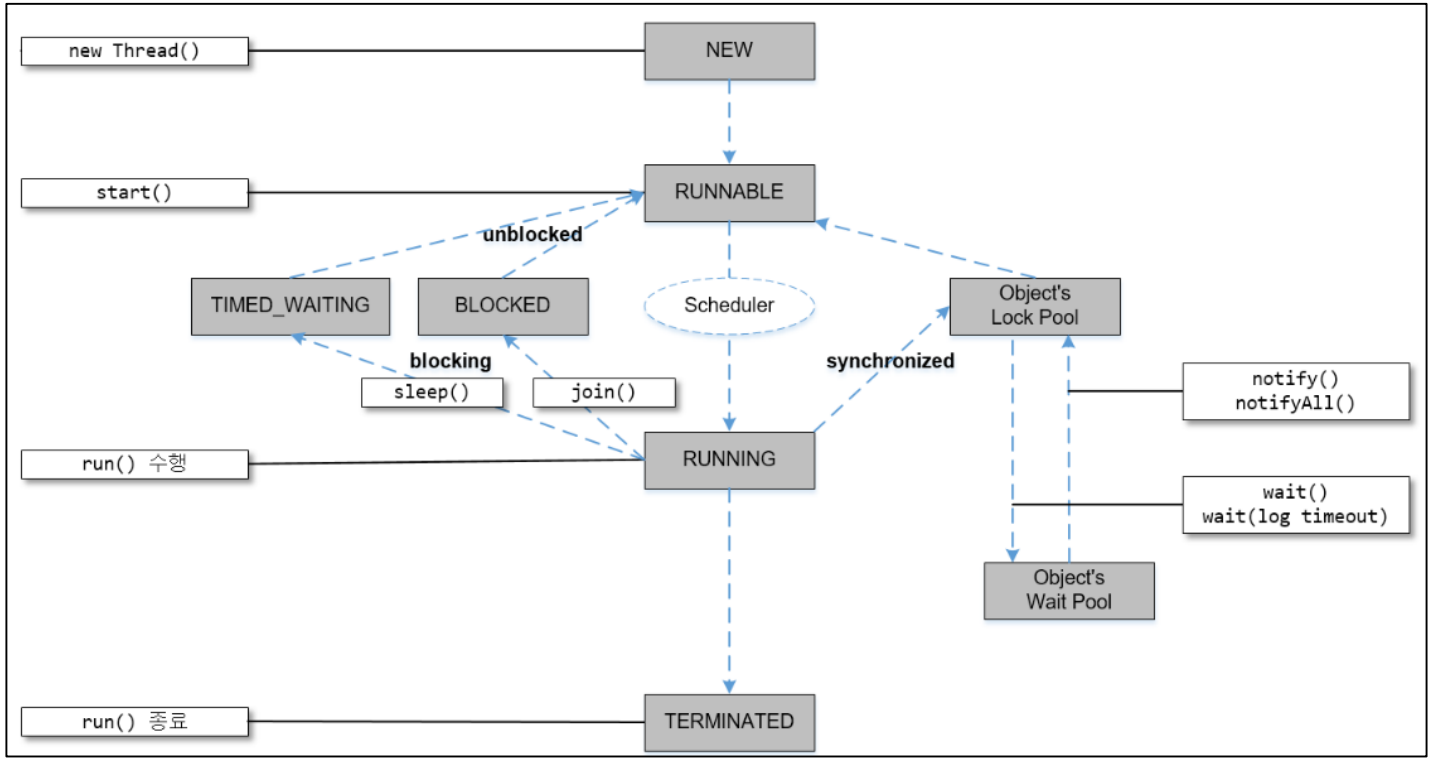
2. 스레드의 라이프사이클

스레드는 현재 상태에 따라 네 가지 상태로 분류할 수 있으며,
상태가 변화하는 주기를 Life Cycle 이라고 한다.

< 스레드 상태 >

new	스레드가 키워드 new를 통해서 인스턴스 화된 상태 Runnable이 될 수 있는 상태이며 아직 대기열에 올라가지 못한 상태
Runnable	start( ) 메서드가 호출되면 new 상태에서 Runnable 상태가 된다. Runnable 상태가 되면 실행할 수 있는 상태로 대기하게 되며 스케줄러에 의해 선택되면 run( )메서드를 바로 수행
Blocked	실행 중인 스레드가 sleep( ), join( ) 메서드를 호출하게 되면, Blocked 상태가 된다. Blocked 상태가 되면 스케줄러에 의해서 선택받을 수 없다.
Dead	run( )메서드의 실행을 모두 완료하게 되면 스레드는 Dead 상태가 된다. 할당받은 메모리와 정보 모두 삭제된다.

2. 스레드의 라이프사이클



출처.https://codedragon.tistory.com/3526

2. 스레드의 라이프사이클

[1] sleep( )

: 스레드를 지정한 시간 동안 block 상태로 만든다
시간은 1000분의 1초 까지 지정할 수 있으며, 지정된 시간이 지나면 다시 runnable (실행 가능한)상태로 돌아간다.

<실습> Exam-95.java

sleep( )의 block 효과를
이용한 카운트 다운

<실행 결과>

```
카운트 다운 5초
5
4
3
2
1
0
종료
```

```java
public class Sleep2 {
    public static void main(String[] args) {
            SleepThread t= new SleepThread();
            t.start();
    }
}

class SleepThread extends Thread {
    public void run() {
        System.out.println("카운트 다운 5초");
        for(int i=5;i>=0;i--) {
            System.out.println(i);
            if(i!=0) {
                try{
                    Thread.sleep(1000); //1초동안 스레드 block
                }
                catch(InterruptedException e) {
                    System.out.println(e.toString());
                }
            }
        }
        System.out.println("종료");
    }
}
```

2. 스레드의 라이프사이클

[2] yield( )

: 자신의 시간을 양보하는 메서드.
스레드가 작업을 수행하던 중 yield( )를 만나면, 자신에게 할당된 시간을 다음 차례 스레드에게 양도

<실습> Exam-96.java

★를 찍는 스레드가
yield( ) 메서드를 호출 시 결과

```
public class Yield1 {
    public static void main(String[] args) {
        MyThread6 s1= new MyThread6();
        MyThread7 s2= new MyThread7();

        Thread t1=new Thread(s1);
        Thread t2=new Thread(s2);

        t1.start();
        t2
        .start();
    }
}
class MyThread6 implements Runnable{
    @Override
    public void run() {
        for(int i=0;i<30;i++) {
            System.out.print("★");
            Thread.yield(); //자신에게 주어진 시간 양보
        }
    }
}
class MyThread7 implements Runnable{
    @Override
    public void run() {
        for(int i=0;i<30;i++) {
            System.out.print("☆");
        }
    }

}
```

<실행 결과>

★★★☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆★★★★★★★★★★★★★★★★★★★★★★★★★★

2. 스레드의 라이프사이클

[3] join( )

: 특정한 스레드가 작업을 먼저 수행할 때 사용
실행 순서를 지켜야 하는 스레드들을 제어한다.

<실습> Exam-97.java

join( )메서드를 알아보는 코드

```
class MyThread8 implements Runnable {
    @Override
    public void run() {
        for (int i = 0; i < 10; i++) {
            System.out.println("t1 : " + i);
        }
        System.out.println("<<t1 완료>>");
    }
}

class MyThread9 implements Runnable {
    @Override
    public void run() {
        for (int i = 0; i < 10; i++) {
            System.out.println("t2 : " + i);
        }
        System.out.println("<<t2 완료>>");
    }
}
```

```
public class Join1 {
    public static void main(String[] args) {
        MyThread8 s1 = new MyThread8();
        MyThread9 s2 = new MyThread9();

        Thread t1 = new Thread(s1);
        Thread t2 = new Thread(s2);
        t1.start();

        try {
            t1.join();// t1를 제외한 다른 스레드 (lock), t1이 완료될때까지 기다림
        } catch (InterruptedException e) {
            System.out.println(e.toString());
        }
        t2.start();
        try {
            t2.join();// t2를 제외한 다른 스레드 (lock), t2이 완료될때까지 기다림
        } catch (InterruptedException e) {
            System.out.println(e.toString());
        }
        for(int i=0;i<10;i++) {
            System.out.println("main : "+i);
        }
    }
}
```

<실행 결과>

```
t1 : 0
t1 : 1
t1 : 2
t1 : 3
t1 : 4
t1 : 5
t1 : 6
t1 : 7
t1 : 8
t1 : 9
<<t1 완료>>
t2 : 0
t2 : 1
t2 : 2
t2 : 3
t2 : 4
t2 : 5
t2 : 6
t2 : 7
t2 : 8
t2 : 9
<<t2 완료>>
main : 0
main : 1
main : 2
main : 3
main : 4
main : 5
main : 6
main : 7
main : 8
main : 9
```