

# 멀티스레드

최문환

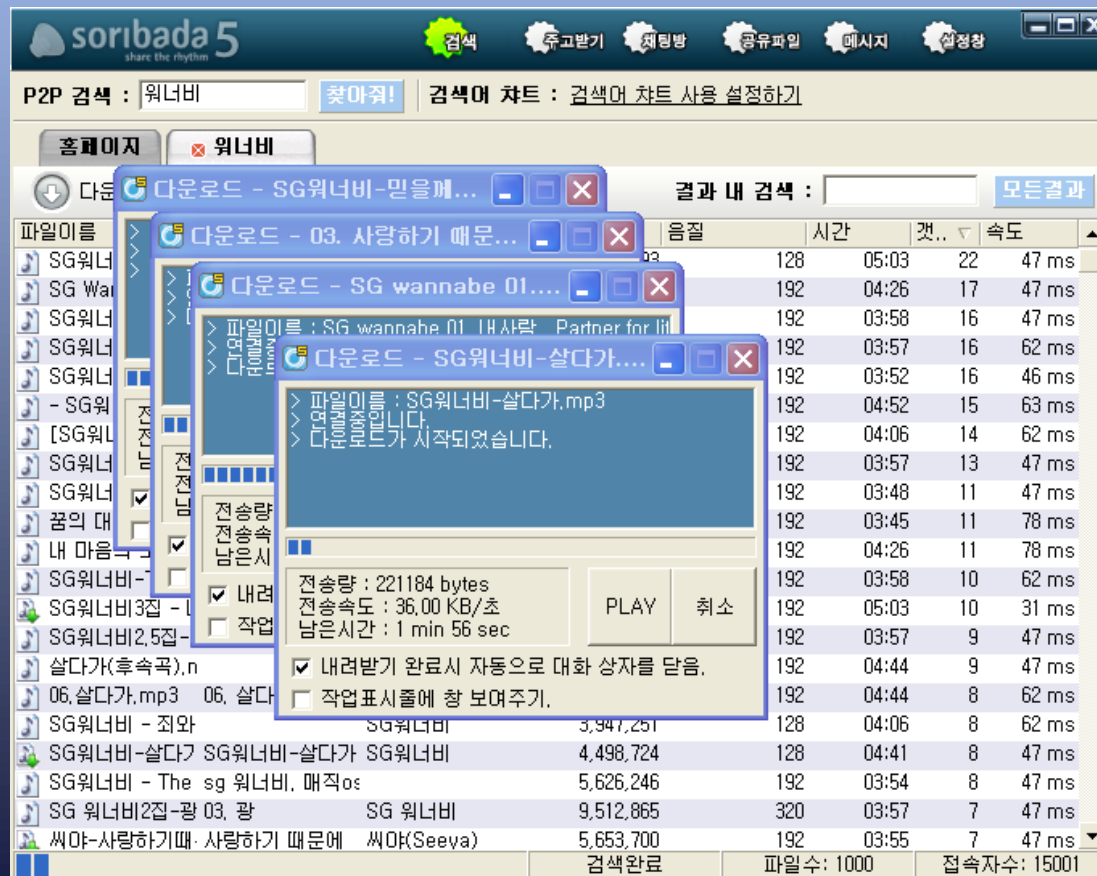


# 멀티스레드

1. 멀티 태스킹과 멀티 스레딩
2. Thread 클래스
3. Runnable 인터페이스
4. 스레드의 다양한 활용
5. 동기화

# 1. 멀티 태스킹과 멀티 스레딩

멀티 태스킹-하나의 CPU가 여러 개의 프로세스를 교대로 수행하는 것입니다.  
멀티 스레딩-하나의 CPU가 여러 개의 스레드를 교대로 수행하는 것입니다.



# 자바에서 스레드 두 가지 형태

- (1) Thread 클래스를 상속 받는 방법
- (2) Runnable 인터페이스를 구현하는 방법

## 2. Thread 클래스

```
class ThreadName extends Thread {  
    public void run() {  
        while(true){  
  
            }//while  
        }//run  
    }//ThreadName 클래스
```

```
ThreadName t = new ThreadName();  
t.start( );
```

## 2. Thread 클래스

Thread의 생성자	설명
<code>public Thread()</code>	스레드 생성
<code>public Thread(String name)</code>	<code>name</code> 을 이름으로 갖는 새로운 스레드 생성
<code>public Thread(Runnable r1)</code>	<code>Runnable</code> 인터페이스의 <code>run</code> 메소드를 호출하는 스레드 생성
<code>public Thread(Runnable r1, String name)</code>	<code>Runnable</code> 인터페이스의 <code>run</code> 메소드를 호출하는 <code>name</code> 을 이름으로 갖는 스레드 생성

# <예제> Thread 클래스를 이용한 멀티스레드를 시 한 애플리케이션

```
001://멀티스레드를 지원하는 클래스를 만들려면 java.lang.Thread 클래스의 상속받은
002:class ThreadExam extends Thread {
003:    ThreadExam(String name){ //생성자에 스레드 이름이 전달인자로 넘어 오
004:        //ThreadExam의 상위 클래스인 Thread의 생성자( super)를 호출
005:        super(name); //전달인자로 준 값이 스레드의 이름이 됨
006:    }
007:
008:    public void run( ){
009:        for(int num=1; num<=5; num++){
010:            for(int k=1; k<1000000000; k++); //시간을 필요로 하는 작업을 위한 설정
011:            //스레드의 이름을 얻어 출력함
012:            System.out.println(getName() + " : " + num);
013:        }
014:    }
015:}
```

# <예제> Thread 클래스를 이용한 멀티스레드 사용한 애플리케이션

```
017: class ThreadTest01{
018:     public static void main (String args[]){
019:         //스레드 클래스를 인스턴스화하여 독립된 스레드를 생성
020:         ThreadExam t1=new ThreadExam("첫 번째 스레드");
021:         ThreadExam t2=new ThreadExam("두 번째 스레드");
022:         //start( ) 메소드를 호출하면 스레드 객체의
023:         t1.start(); //run( ) 메소드가 돌면서 스레드가 실행
024:         t2.start();
025:     }
026: }
```



## 2.1 main 메소드도 스레드다.

<예제> 메인 스레드의 이름 출력 [파일이름:ThreadTest00.java]

```
class ThreadTest00{  
    public static void main(String args[] ) {  
        System.out.println(Thread.currentThread( ).getName( ));  
    }  
}
```

### 3. Runnable 인터페이스

```
class RunnableExam implements Runnable {  
    public void run(){  
        while(true){  
  
            }//while  
        }//run  
    }  
}
```

```
RunnableExam R1 = new RunnableExam("첫 번째 스레드");  
Thread t1=new Thread(R1);  
t1.start();
```

## 5. 스레드의 다양한 활용

<예제> Runnable 인터페이스를 이용한 프레임에 글자 이동시키기

```
001:import java.awt.*;
002:import java.awt.event.*;
003://Runnable 인터페이스를 구현하는 클래스를 선언
004:class GraphicFrame extends Frame implements Runnable {
005:    int x=1;
006:    public GraphicFrame( ){
007:        setBackground(new Color(0,0,0));    //프레임의 배경색을 검정색으로 지정
008:        setSize(370,150);                    //프레임의 크기 지정
009:        setVisible(true);
010:        addWindowListener(new WindowAdapter( ){//[닫기]버튼이 눌리면 종료하도록
011:            public void windowClosing(WindowEvent e) {
012:                dispose();
013:                System.exit(0);
014:            }
015:        });
016:    }
```

## 5. 스레드의 다양한 활용

```
016: //스레드 객체가 수행해야 하는 작업을 run( ) 메소드 내부에 기술
017: public void run( ){
018:     while(true){
019:         try{
020:             Thread.sleep(100); //스레드를 일정시간 동안 대기 상태로 둔다.
021:         }catch(InterruptedException e){ }
022:         //프로그래머에 의해 애플릿이 다시 그려져야 할 필요가 있을 때
023:         repaint( ); //repaint( ) 메소드를 명시적으로 호출해야 paint 메소드가
024:         x+=5;
025:     }
026: }
027:
028: public void paint(Graphics g){
029:     Dimension d; //Dimension은 폭(width)과 높이(height)를 가지는 클래스
030:     d = getSize( ); //프레임 창의 크기를 구하여 Dimension 객체로 반환한다.
031:     g.setColor(Color.yellow);
032:     //글자를 출력한다. 출력되는 위치가 x 변수값이 run( ) 메소드에서 변경되므로
033:     g.drawString("클릭하세요 JAVA !!", x, d.height/2); //글자가 움직이게 된다
```

## 5. 스레드의 다양한 활용

```
034:
035:  if(x > d.width) //출력될 x 좌표값이 프레임 창 영역 밖(오른쪽 끝)으로 벗어나면
036:      x=0;//다시 처음(왼쪽 끝)부터 다시 시작한다.
037:  }
038:}
039:
040:public class ThreadTest05{
041:  public static void main (String args[]){
042:    //Frame을 상속받고 또한 Runnable 인터페이스를 구현한 인스턴스 생성
043:    GraphicFrame f = new GraphicFrame( );
044:    //스레드를 생성할 때 Runnable 인터페이스를 매개변수로 넘겨줌
045:    Thread t1=new Thread( f );
046:    t1.start();
047:  }
048:}
```

# 7. 동기화

- ★ 메소드의 동기화 방법

```
public synchronized void Method(){  
    //임계영역 처리 구문  
}
```

- ★ 특정 블록의 동기화 방법

```
public void Method(){  
    synchronized(동기화할 객체 또는 동기화할 클래스명){  
        //임계영역 처리 구문  
    }  
}
```

# <예제> 특정 은행 계좌에서 입금, 출금, 잔액 조회를 하는 클래스

```
001: class Atm { //Atm 계좌
002: // 은행 계좌 잔액
003: private int money ;
004: public Atm(int m){ // 생성자
005:     money = m; //계좌 개설시 입금한 금액
006: }
007:
008: synchronized void deposit(int amount, String name){ // 입금
009:     money += amount ;
010:     System.out.println(name+": 입금 금액 : "+ amount);
011: }
012:
013: synchronized void withdraw(int amount, String name){ // 출금
014:     if( (money - amount) > 0 ) { // 출금 가능하면
015:         money -= amount ;
016:         System.out.println(name+": 출금 금액 : "+ amount);
017:     }
```

# <예제> 특정 은행 계좌에서 입금, 출금, 잔액 조회를 하는 클래스

```
018:     else{
019:         System.out.println(name +": 출금 못함 (잔액이 부족)");
020:     }
021: }
022:
023: public void getmoney( ){
024:     System.out.println("    계좌 잔액은:"+ money) ;
025: }
026: }
027:
028: class AtmUser extends Thread { // Atm 사용자
029:     boolean flag = false ; // 입금/인출
030:     Atm obj ;
031:     public AtmUser( Atm obj, String name){
032:         super(name);
033:         this.obj = obj ;
034:     }
```



# <예제> 특정 은행 계좌에서 입금, 출금, 잔액 조회를 하는 클래스

```
035: public void run(){
036:     for(int i = 0 ; i < 5 ; i++){
037:         try{
038:             sleep(500);
039:         }catch(InterruptedException e){}
040:
041:         if(flag){
042:             obj.deposit((int)(Math.random()*10+2)*100, getName());
043:             obj.getmoney();
044:         }
045:         else{
046:             obj.withdraw((int)(Math.random()*10+2)*100, getName());
047:             obj.getmoney();
048:         }
049:         flag = !flag ;
050:     }
050: }
051: }
```

# <예제> 특정 은행 계좌에서 입금, 출금, 잔액 조회를 하는 클래스

```
052: class ThreadTest08 {  
053:     public static void main(String [] args){  
054:         Atm obj = new Atm(1000); //Atm 계좌 개설 하여 1000원 입금  
055:         AtmUser user1 = new AtmUser(obj,"성윤정");//3명이 동일 계좌에서 입 출금함  
056:         AtmUser user2 = new AtmUser(obj,"전수빈");  
057:         AtmUser user3 = new AtmUser(obj,"전원지");  
058:  
059:         user1.start();  
060:         user2.start();  
061:         user3.start();  
062:     }  
063: }
```