

## ■ 스레드 (Thread)

- 프로세스는 실행중인 프로그램을 의미한다.
- 스레드는 프로세스 내에서 별도의 실행흐름을 갖는 대상이다.
- 프로세스 내에서 둘 이상의 스레드를 생성하는 것이 가능하다.



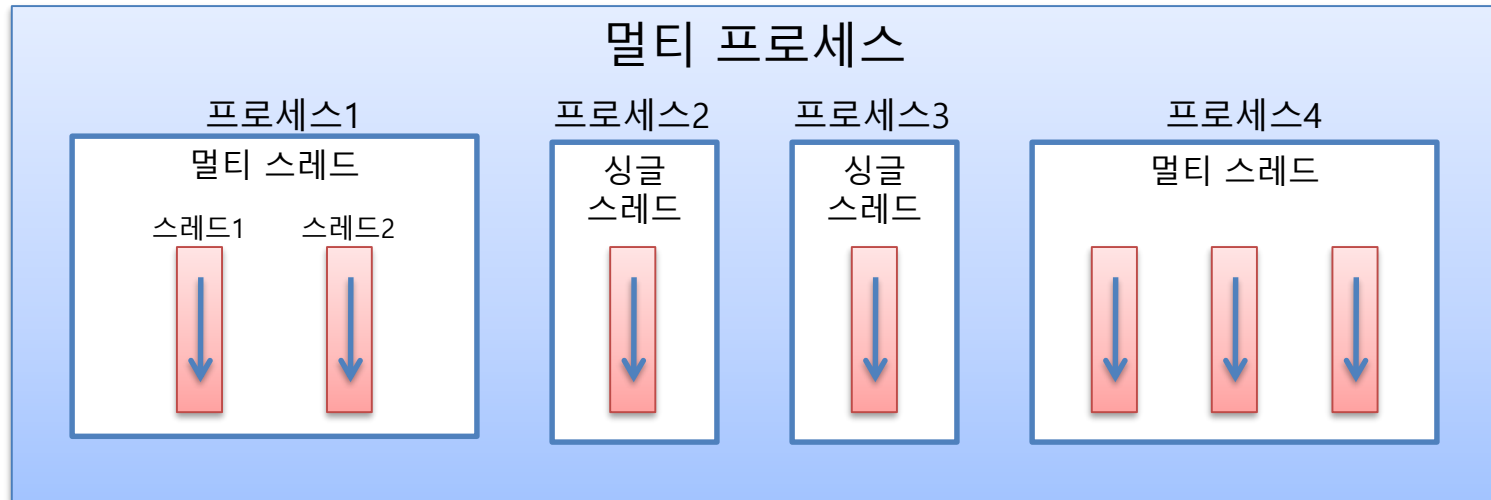
프로그램이 실행될 때 프로세스에 할당된 메모리, 이 자체를 단순히 프로세스라고 하기도 한다.

사실 스레드는 모든 일의 기본 단위이다. main 메소드를 호출하는 것도 프로세스 생성시 함께 생성되는 **main 스레드**를 통해서 이뤄진다.

## ■ 스레드 (Thread)

### ● 멀티 태스킹 (Multi Tasking)

- 두 가지 이상의 작업을 동시에 처리하는 것



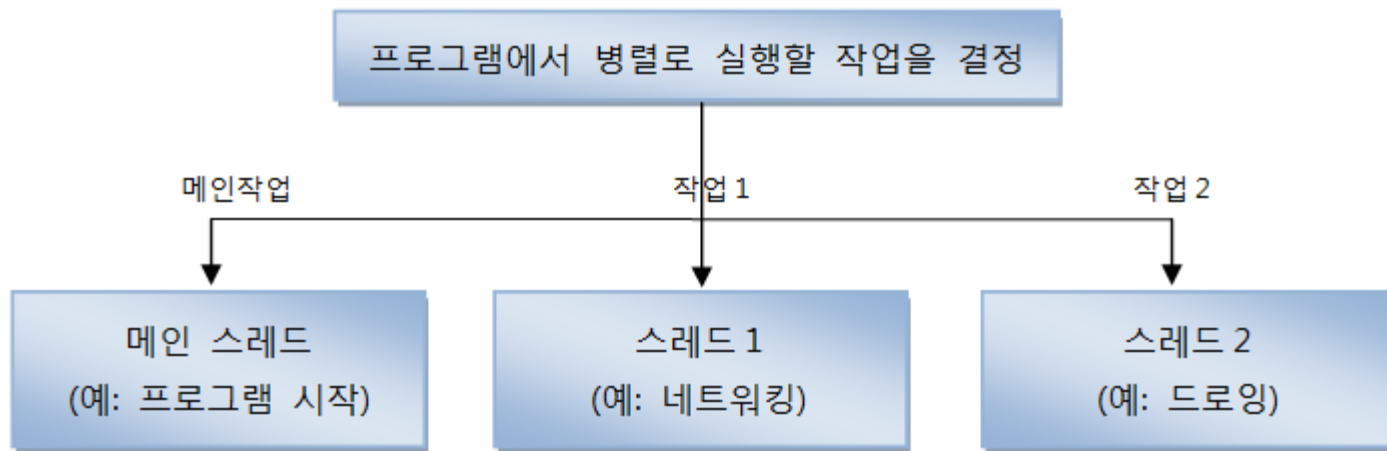
## ■ 스레드 (Thread)

### ● Main Thread

- 모든 자바 프로그램은 메인 메소드를 실행하며 시작
- 마지막까지 코드를 실행하거나 return을 만나면 종료

### ● Main에서 별도의 Thread를 하나 이상 만들면 Multi Thread

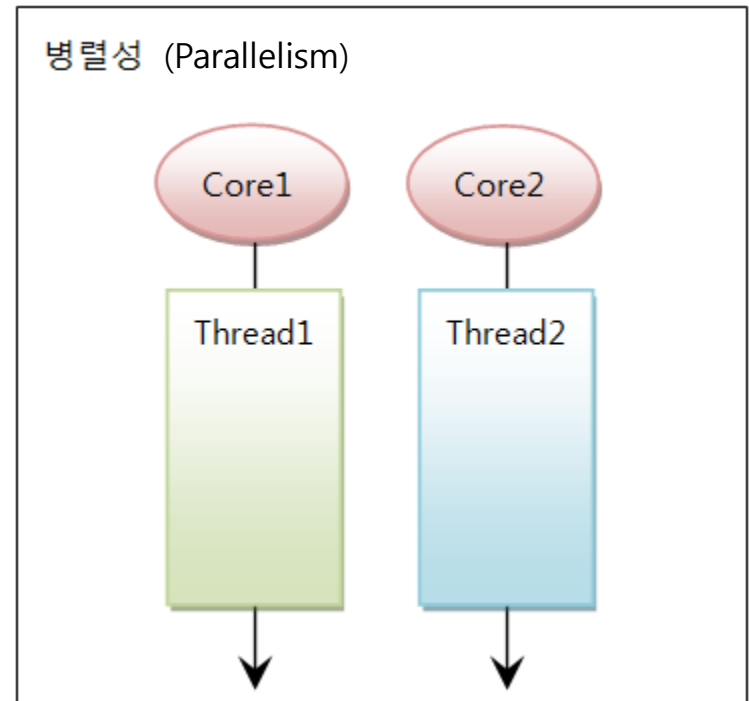
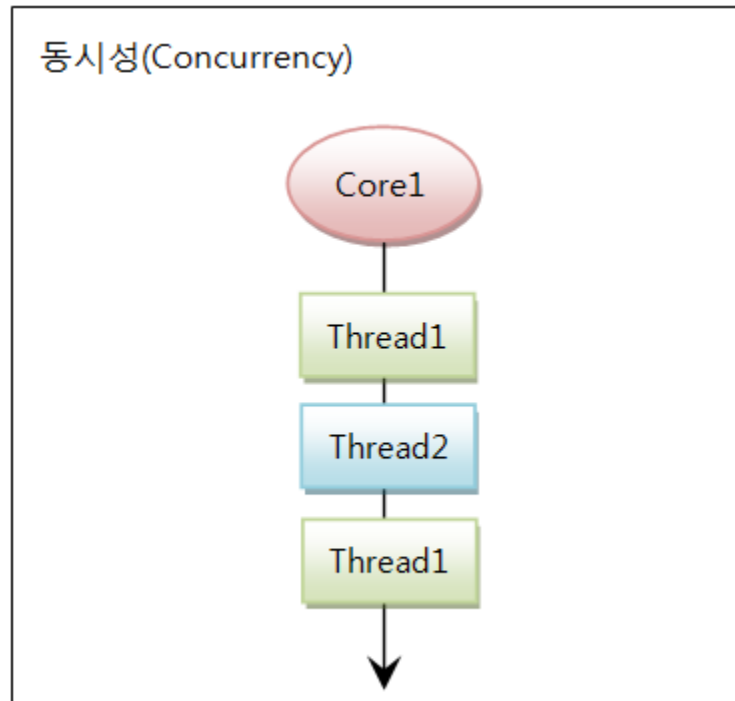
- Multi Thread의 경우 모든 스레드가 종료되어야 프로그램 종료



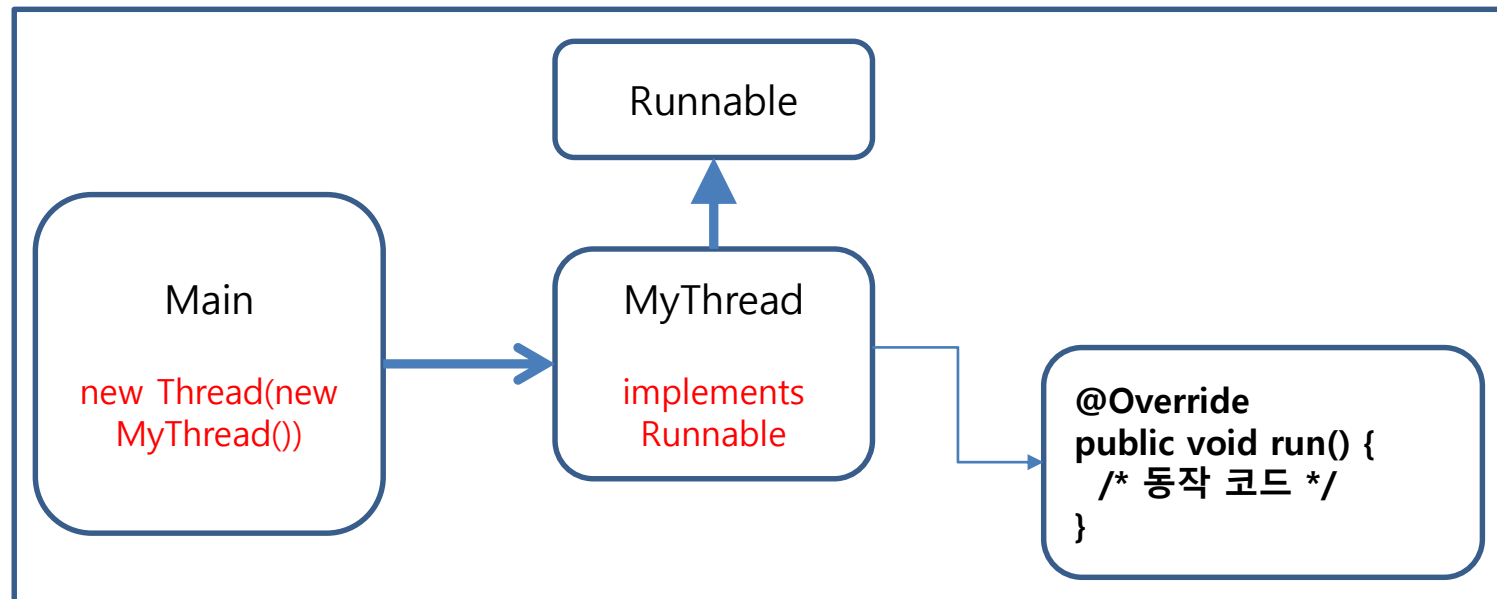
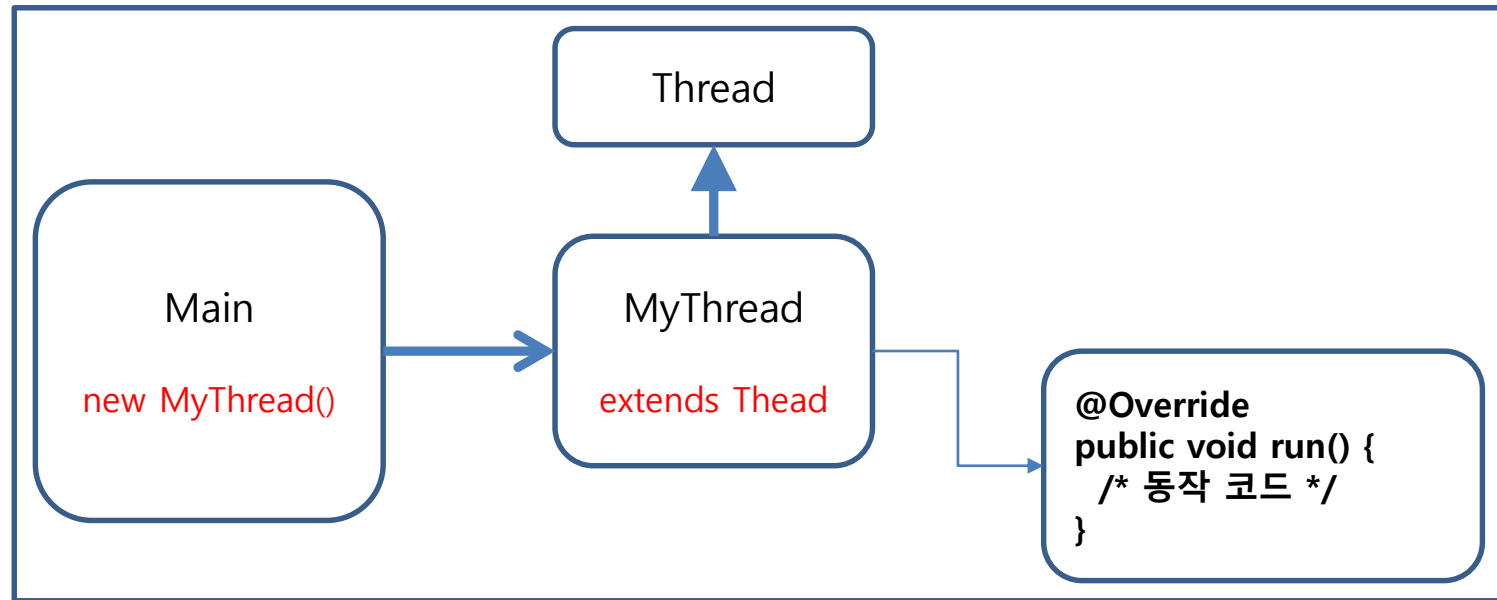
## ■ 스레드 (Thread)

### ● 동시성과 병렬성

- 동시성 (Concurrency) : 하나의 코어에서 멀티 스레드가 번갈아 가며 실행
- 병렬성 (Parallelism) : 멀티 코어에서 개별 스레드를 동시에 실행



## ■ Thread 사용 (extends Thread / implements Runnable)



## ■ Thread 사용 – 1 (extends Thread)

```
public class MyThread1 extends Thread {  
    String name;  
  
    MyThread1(String name) {  
        this.name = name;  
    }  
  
    public void run() {  
        for(int i = 0; i < 10; i++) {  
            System.out.println(name + " : " + i);  
        }  
    }  
}
```

```
public class ThreadExam1 {  
    public static void main(String[] args) {  
        MyThread1 t1 = new MyThread1("first");  
        MyThread1 t2 = new MyThread1("second");  
  
        t1.start();  
        t2.start();  
    }  
}
```

```
first : 0  
second : 0  
second : 1  
first : 1  
second : 2  
first : 2  
second : 3  
first : 3  
second : 4  
first : 4  
second : 5  
first : 5  
second : 6  
second : 7  
second : 8  
second : 9  
first : 6  
first : 7  
first : 8  
first : 9
```

## ■ Thread 사용 – 2 (implements Runnable)

```
public class MyThread2 implements Runnable {  
    String name;  
  
    MyThread2(String name) {  
        this.name = name;  
    }  
  
    @Override  
    public void run() {  
        for(int i = 0; i < 10; i++) {  
            System.out.println(name + " : " + i);  
        }  
    }  
}
```

```
public class ThreadExam2 {  
    public static void main(String[] args) {  
        MyThread2 tt1 = new MyThread2("first");  
        MyThread2 tt2 =  
            new MyThread2("second");  
  
        Thread thread1 = new Thread(tt1);  
        Thread thread2 = new Thread(tt2);  
        thread1.start();  
        thread2.start();  
    }  
}
```

## ■ Thread 상태 제어

- 주어진 시간 동안 일시 정지 (sleep)

```
try {  
    Thread.sleep(1000);  
} catch (InterruptedException e) {  
    // interrupt() 메소드가 호출되면 실행  
}
```

- 밀리 세컨드(1/1000) 단위로 지정



## ■ Thread 사용 – 3 (sleep)

```
public class MyThread3 extends Thread {  
    @Override  
    public void run() {  
        for(int i = 1; i <= 10; i++) {  
            try {  
                System.out.println(  
                    i*10 + "퍼센트 완료");  
                sleep(1000);  
            } catch (InterruptedException e) {  
                e.printStackTrace();  
            }  
        }  
    }  
}
```

```
public class ThreadExam3 {  
    public static void main(String[] args) {  
        MyThread3 t = new MyThread3();  
        t.start();  
    }  
}
```

## ■ Thread 사용 – 4 (sleep)

```
public class MyThread4 implements Runnable {  
    @Override  
    public void run() {  
        for(int i = 1; i <= 10; i++) {  
            try {  
                System.out.println(  
                    i*10 + "퍼센트 완료");  
                Thread.sleep(1000);  
            } catch (InterruptedException e) {  
                e.printStackTrace();  
            }  
        }  
    }  
}
```

```
public class ThreadExam4 {  
    public static void main(String[] args) {  
        MyThread4 t = new MyThread4();  
        new Thread(t).start();  
    }  
}
```

## ■ Thread 미사용 – 5 (sleep)

```
import java.awt.Toolkit;

public class BeepPrint1 {
    public static void main(String[] args) {
        Toolkit toolkit = Toolkit.getDefaultToolkit();
        for(int i = 0; i < 5; i++) {
            toolkit.beep();
            try {
                Thread.sleep(1000);
            } catch(Exception e) {}
        }

        for(int i = 0; i < 5; i++) {
            System.out.println("실행");
            try {
                Thread.sleep(500);
            } catch(Exception e) {}
        }
    }
}
```

비프음 5번 동작 후

실행  
실행  
실행  
실행  
실행

## ■ Thread 사용 – 6 (sleep)

```
import java.awt.Toolkit;

public class BeepPrint2 {
    public static void main(String[] args) {
        Toolkit toolkit = Toolkit.getDefaultToolkit();
        new Thread(new Runnable() {
            @Override
            public void run() {
                for (int i = 0; i < 5; i++) {
                    toolkit.beep();
                    try {
                        Thread.sleep(1000);
                    } catch (Exception e) {}
                }
            }
        }).start();

        for (int i = 0; i < 5; i++) {
            System.out.println("실행");
            try {
                Thread.sleep(1000);
            } catch (Exception e) {}
        }
    }
}
```

비프음 5번 동작과 같이

실행  
실행  
실행  
실행  
실행

## ■ 클릭된 위치에 이미지 생성 및 이동 – Thread 미사용

```
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class Balloon {
    public static void makeBalloon(JPanel panel, int x, int y) {
        /* 이미지 생성 */
        ImageIcon icon = new ImageIcon("image/balloon.png");

        /* 이미지를 보여주기 위해서 JLabel 사용 */
        JLabel label = new JLabel(icon);

        /* 이미지 크기 지정 */
        label.setSize(icon.getIconWidth(), icon.getIconHeight());

        /* 이미지를 가지고 있는 JLabel을 JPanel로 추가 */
        panel.add(label);
    }
}
```

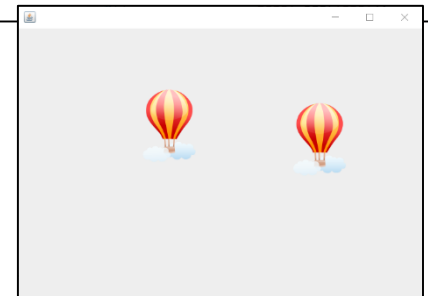
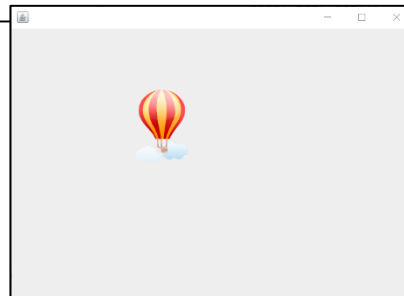
## ■ 클릭된 위치에 이미지 생성 및 이동 – Thread 미사용

```
for(int i = 0; i < 10; i++) {  
    /* 이미지가 보여질 위치 지정 */  
    label.setLocation(x, y);  
    try {  
        /* 그래픽 표현 상태 변경 후 새로 고침을 해야 화면에 반영 */  
        panel.repaint();  
        Thread.sleep(200);  
    } catch (InterruptedException e1) {  
        e1.printStackTrace();  
    }  
    /* Y축 위치 변경 (위로 이동) */  
    y -= 25;  
}
```

## ■ 클릭된 위치에 이미지 생성 및 이동 – Thread 미사용

```
public static void main(String[] args) {  
    JFrame j = new JFrame();  
    JPanel panel = new JPanel();  
    panel.addMouseListener(new MouseAdapter() {  
        @Override  
        public void mousePressed(MouseEvent e) {  
            makeBalloon(panel, e.getX(), e.getY());  
        }  
    });  
    j.add(panel);  
    j.setSize(700, 500);  
    j.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
    j.setVisible(true);  
}
```

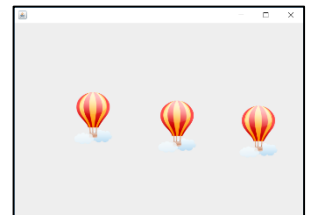
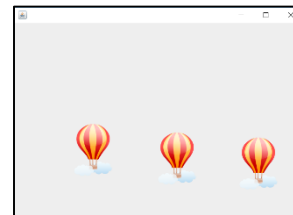
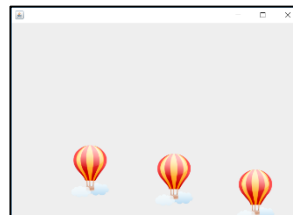
클릭하여 이미지 생성 및 이동 완료 후 다음 동작 가능



## ■ 클릭된 위치에 이미지 생성 및 이동 – Thread 사용

```
public static void makeBalloon(JPanel panel, int x, int y) {  
    /* Thread 사용 */  
    new Thread(new Runnable() {  
        @Override  
        public void run() {  
            /* 내부 클래스에서 지역변수 사용 */  
            int y2 = y;  
            ImageIcon icon = new ImageIcon("image/balloon.png");  
            JLabel label = new JLabel(icon);  
            label.setSize(icon.getIconWidth(), icon.getIconHeight());  
            panel.add(label);  
            for(int i = 0; i < 10; i++) {  
                label.setLocation(x, y2);  
                try {  
                    panel.repaint();  
                    Thread.sleep(200);  
                } catch (InterruptedException e1) {  
                    e1.printStackTrace();  
                }  
                y2 -= 25;  
            }  
        }  
    }).start();  
}
```

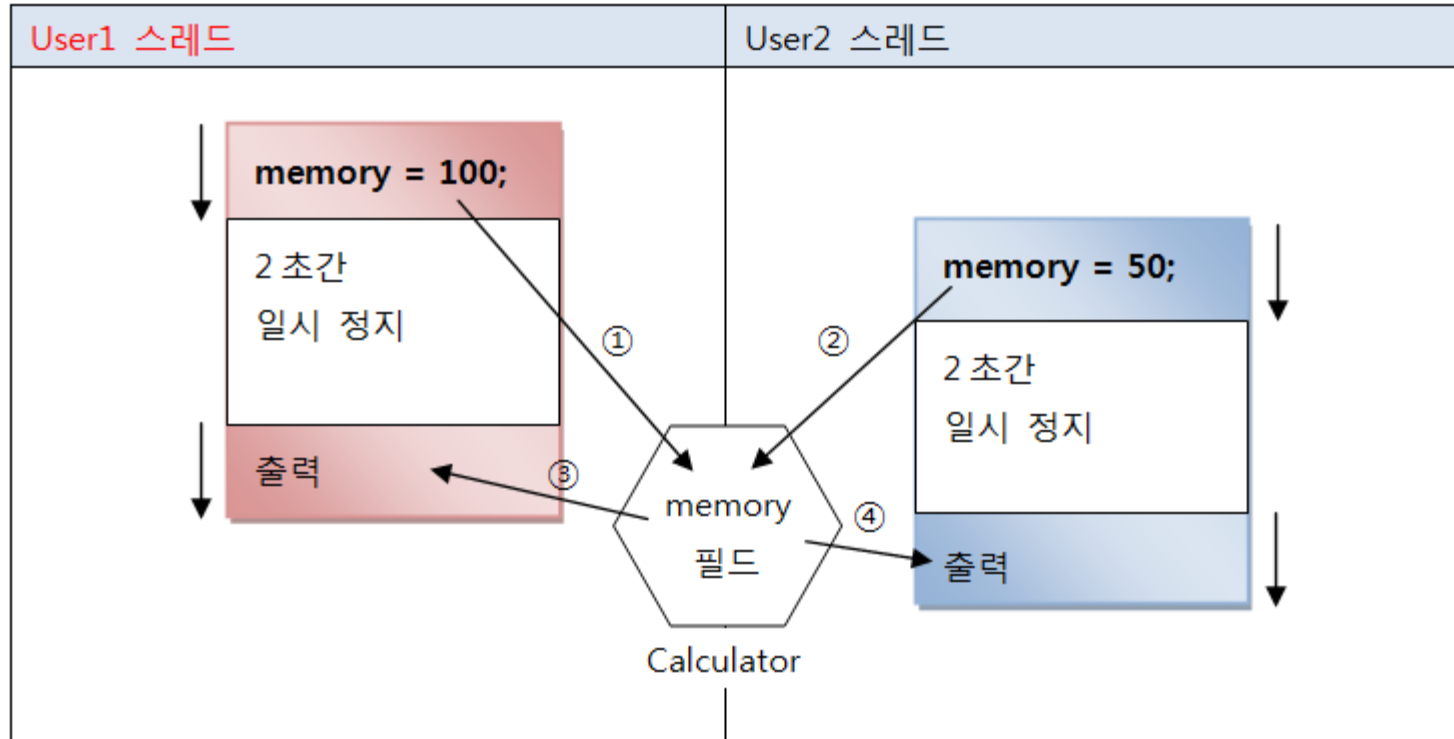
이전 작업과 상관없이 다음 동작 실행





## ■ Thread 동기화

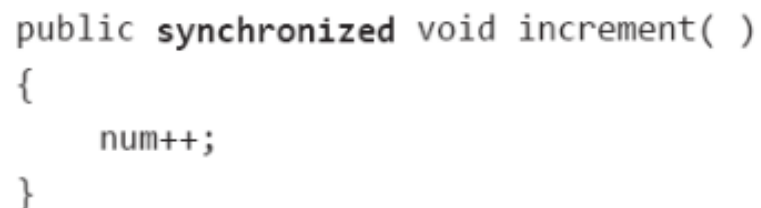
- Thread는 하나의 자원(객체)를 공유하는 경우 주의



## ■ 동기화 기법

```
class Increment
{
    int num=0;
    public synchronized void increment(){ num++; }
    public int getNum() { return num; }
}

class IncThread extends Thread
{
    Increment inc;
    public IncThread(Increment inc)
    {
        this.inc=inc;
    }
    public void run()
    {
        for(int i=0; i<10000; i++)
            for(int j=0; j<10000; j++)
                inc.increment();
    }
}
```



```
public synchronized void increment( )
{
    num++;
}
```

**동기화 메소드의 선언!**

**synchronized 선언으로 인해서 increment 메소드는 쓰레드에 안전한 함수가 된다.**

## ■ 동기화 미적용 - 1

```
public class ValueClass {  
    private int number;  
    public void addNumber(int number) {  
        this.number += number;  
    }  
    public int getNumber() {  
        return number;  
    }  
}
```

```
public class ThreadExam5 {  
    public static void main(String[] args) {  
        ValueClass vc = new ValueClass();  
        new MyThread5(vc, 1, 500).start();  
        new MyThread5(vc, 501, 1000).start();  
    }  
}
```

```
public class MyThread5 extends Thread {  
    ValueClass vc;  
    int start;  
    int end;  
  
    MyThread5(ValueClass vc, int start, int end) {  
        this.vc = vc;  
        this.start = start;  
        this.end = end;  
    }  
  
    public void run() {  
        for(int i = start; i <= end; i++) {  
            vc.addNumber(i);  
        }  
        System.out.println(vc.getNumber());  
    }  
}
```

## ■ 동기화 적용 - 1

```
public class ValueClass {  
    private int number;  
    public synchronized void addNumber(  
        int number) {  
        this.number += number;  
    }  
    public int getNumber() {  
        return number;  
    }  
}
```

```
public class ThreadExam5 {  
    public static void main(String[] args) {  
        ValueClass vc = new ValueClass();  
        new MyThread5(vc, 1, 500).start();  
        new MyThread5(vc, 501, 1000).start();  
    }  
}
```

```
public class MyThread5 extends Thread {  
    ValueClass vc;  
    int start;  
    int end;  
  
    MyThread5(ValueClass vc, int start, int end) {  
        this.vc = vc;  
        this.start = start;  
        this.end = end;  
    }  
  
    public void run() {  
        for(int i = start; i <= end; i++) {  
            vc.addNumber(i);  
        }  
        System.out.println(vc.getNumber());  
    }  
}
```

## ■ 동기화 미적용 - 2

```
public class Counter {  
    private int value = 0;  
    public void increment() {  
        value++;  
    }  
    public int getValue() {  
        return this.value;  
    }  
}
```

```
public class ThreadExam6 {  
    public static void main(String[] args) {  
        Counter c = new Counter();  
        new MyThread6(c, 1).start();  
        new MyThread6(c, 2).start();  
        new MyThread6(c, 3).start();  
        new MyThread6(c, 4).start();  
    }  
}
```

```
public class MyThread6 extends Thread {  
    private Counter counter;  
    private int number;  
  
    MyThread6(Counter counter, int number) {  
        this.counter = counter;  
        this.number = number;  
    }  
  
    public void run() {  
        int i = 0;  
        while(i < 1000) {  
            counter.increment();  
  
            try {  
                int random = (int) Math.random();  
                sleep(random);  
            } catch (InterruptedException e) {  
                e.printStackTrace();  
            }  
            i++;  
        }  
        System.out.println(  
            number + " " + counter.getValue());  
    }  
}
```

## ■ 동기화 적용 - 2

```
public class Counter {  
    private int value = 0;  
    public synchronized void increment() {  
        value++;  
    }  
    public int getValue() {  
        return this.value;  
    }  
}
```

```
public class ThreadExam6 {  
    public static void main(String[] args) {  
        Counter c = new Counter();  
        new MyThread6(c, 1).start();  
        new MyThread6(c, 2).start();  
        new MyThread6(c, 3).start();  
        new MyThread6(c, 4).start();  
    }  
}
```

```
public class MyThread6 extends Thread {  
    private Counter counter;  
    private int number;  
  
    MyThread6(Counter counter, int number) {  
        this.counter = counter;  
        this.number = number;  
    }  
  
    public void run() {  
        int i = 0;  
        while(i < 1000) {  
            counter.increment();  
            try {  
                int random = (int) Math.random();  
                sleep(random);  
            } catch (InterruptedException e) {  
                e.printStackTrace();  
            }  
            i++;  
        }  
        System.out.println(  
            number + " " + counter.getValue());  
    }  
}
```

## ■ 동기화 적용 – 3 (1 / 2)

```
public class WashRoom {  
    public void synchronized useRoom(  
        String name) {  
        System.out.println(name + " 입장");  
        try {  
            System.out.println(name + " 사용중");  
            int random =  
                new Random().nextInt(3) + 1;  
            Thread.sleep(random * 1000);  
        } catch (InterruptedException e) {  
            e.printStackTrace();  
        }  
        System.out.println(name + " 퇴장");  
    }  
}
```

```
public class FamilyThread extends Thread {  
    private WashRoom room;  
    private String name;  
  
    public FamilyThread(String name, WashRoom  
        room) {  
        this.name = name;  
        this.room = room;  
    }  
  
    @Override  
    public void run() {  
        room.useRoom(name);  
    }  
}
```

## ■ 동기화 적용 – 3 (2 / 2)

```
public class FamilyLauncher {  
    public static void main(String[] args) {  
        WashRoom room = new WashRoom();  
        FamilyThread father = new FamilyThread("father", room);  
        FamilyThread mother = new FamilyThread("mother", room);  
        FamilyThread syster = new FamilyThread("syster", room);  
        FamilyThread brother = new FamilyThread("brother", room);  
  
        father.start();  
        mother.start();  
        syster.start();  
        brother.start();  
    }  
}
```