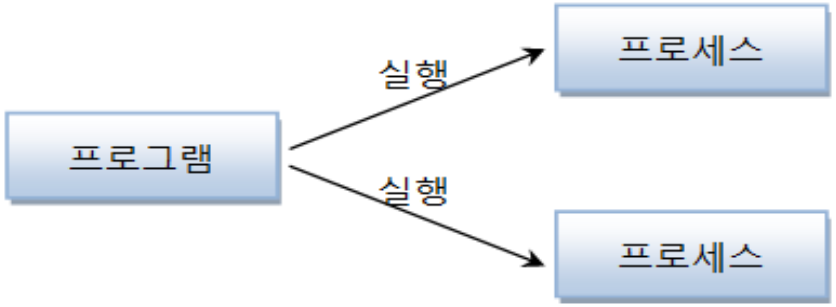
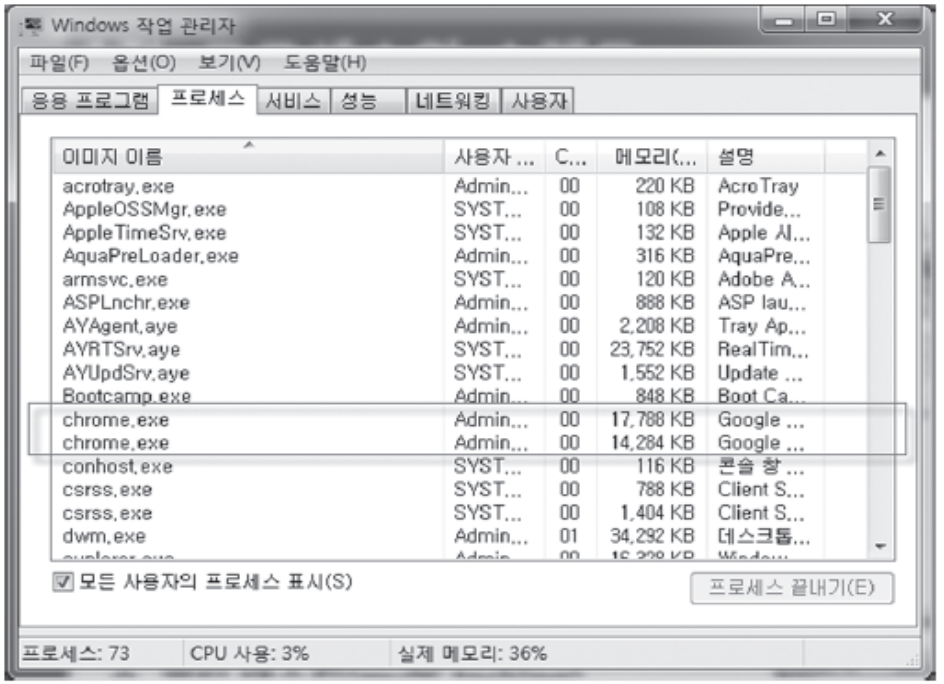
멀티 스레드

멀티 스레드

❖ 프로세스(process)

- 실행 중인 하나의 프로그램
- 하나의 프로그램이 여러 프로세스로 만들어짐

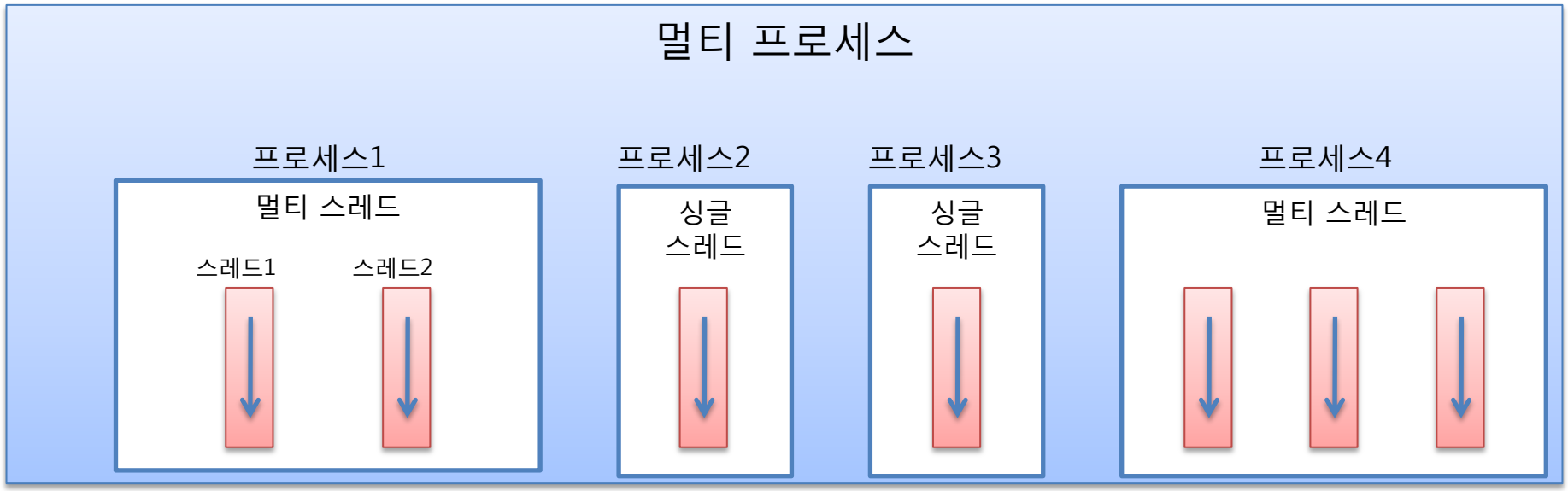




멀티 스레드

❖ 멀티 태스킹(multi tasking)

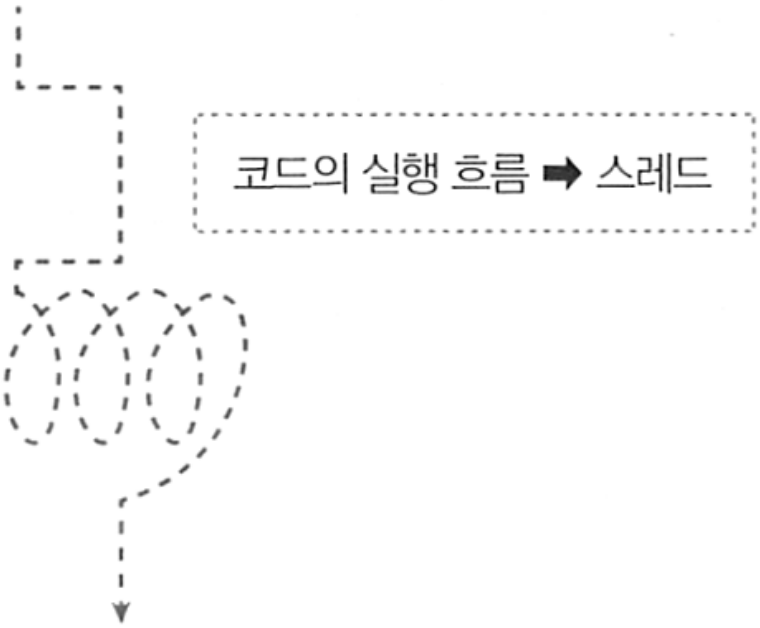
- 두 가지 이상의 작업을 동시에 처리하는 것
- 멀티 프로세스
  - 독립적으로 프로그램들을 실행하고 여러 가지 작업 처리
- 멀티 스레드
  - 한 개의 프로그램을 실행하고 내부적으로 여러 가지 작업 처리



❖ 메인(main) 스레드

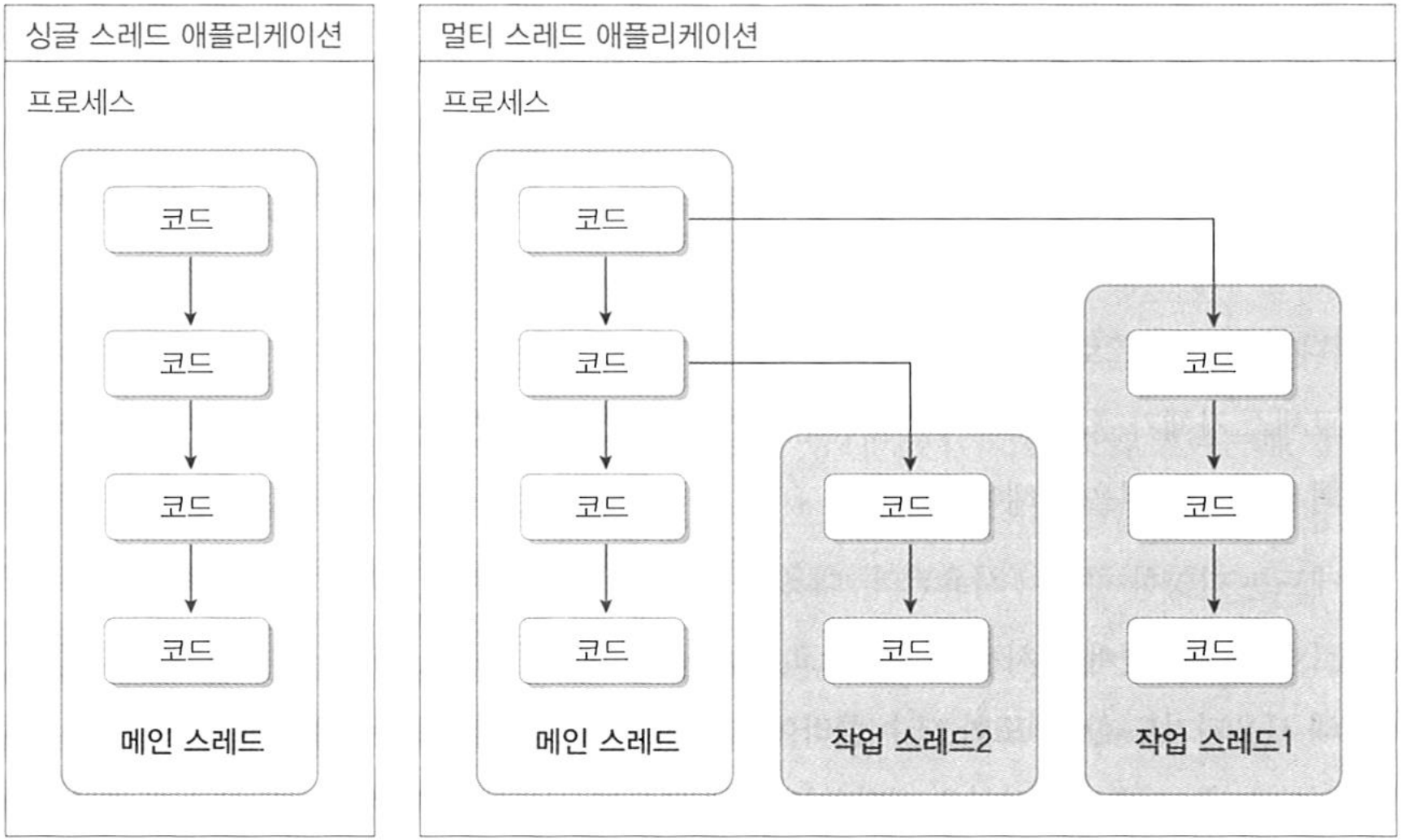
- 모든 자바 프로그램은 메인 스레드가 main() 메소드 실행하며 시작
- main() 메소드의 첫 코드부터 아래로 순차적으로 실행

```
public static void main(String[] args) {
    String data = null;
    if(…) {
    }
    while(...) {
    }
    System.out.println("…");
}
```



멀티 스레드

❖ 메인(main) 스레드



멀티 스레드

❖ 메인(main) 스레드

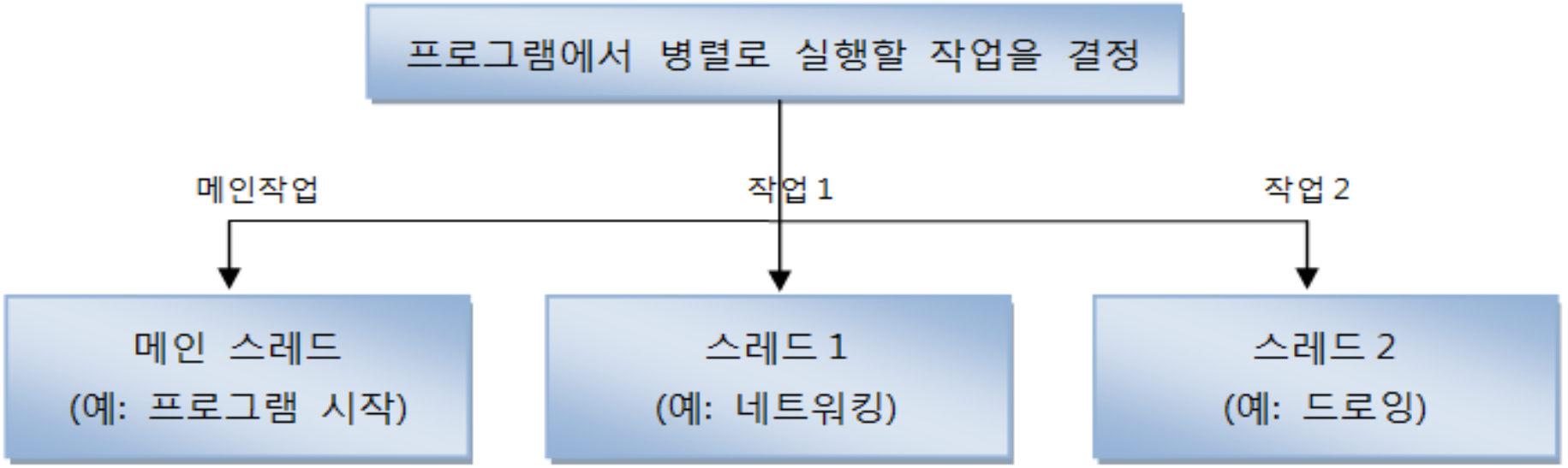
- 실행 종료 조건
  - 마지막 코드 실행
  - return 문을 만나면

- main 스레드는 작업 스레드들을 만들어 병렬로 코드들 실행
  - 멀티 스레드 생성해 멀티 태스킹 수행

- 프로세스의 종료
  - 싱글 스레드: 메인 스레드가 종료하면 프로세스도 종료
  - 멀티 스레드: 실행 중인 스레드가 하나라도 있다면, 프로세스 미종료

❖ 멀티 스레드로 실행하는 어플리케이션 개발

- 몇 개의 작업을 병렬로 실행할지 결정하는 것이 선행되어야



❖ 작업 스레드 생성 방법

- Thread 클래스로부터 직접 생성
- Runnable을 매개값으로 갖는 생성자 호출

```
Thread thread = new Thread(Runnable target);
```

```
class Task implements Runnable {
    public void run() {
        스레드가 실행할 코드;
    }
}
```

```
Runnable task= new Task();
Thread thread = new Thread(task);
```

❖ 작업 스레드 생성 방법

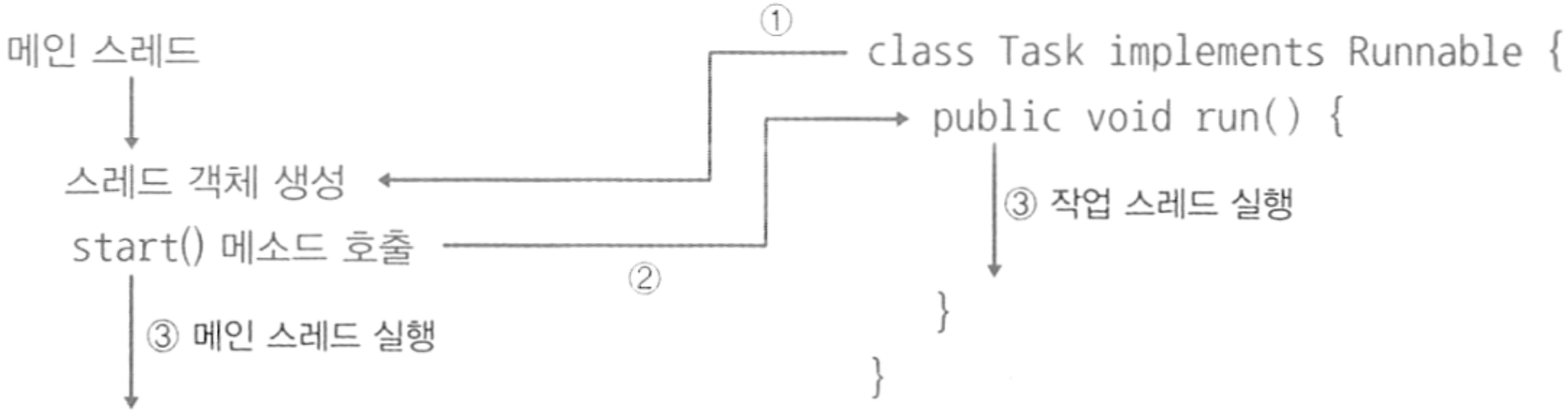
- Thread 클래스로부터 직접 생성
  - 익명 클래스 정의를 통한 스레드 생성

```
Thread thread = new Thread(new Runnable() {
    public void run() {
        스레드가 실행할 코드;
    }
});
```

익명 구현 객체

❖ 작업 스레드 생성 방법

- 스레드의 실행
  - `Thread.start( );`



❖ BeepPrintExample1.java

```
import java.awt.Toolkit;

public class BeepPrintExample1 {
    public static void main(String[] args) {
        Toolkit toolkit = Toolkit.getDefaultToolkit();

        for(int i=0; i<5; i++) {
            toolkit.beep();
            try { Thread.sleep(500); } catch(Exception e) {}
        }

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

❖ BeepTask.java

```
import java.awt.Toolkit;

public class BeepTask implements Runnable {
    public void run() {
        Toolkit toolkit = Toolkit.getDefaultToolkit();
        for(int i=0; i<5; i++) {
            toolkit.beep();
            try { Thread.sleep(500); } catch(Exception e) {}
        }
    }
}
```

❖ BeepPrintExample2.java

```
import java.awt.Toolkit;

public class BeepPrintExample2 {
    public static void main(String[] args) {
        //how1
        Runnable beepTask = new BeepTask();
        Thread thread = new Thread(beepTask);

        //how2 : Runnable 익명 객체 이용
        /*Thread thread = new Thread(new Runnable() {
            @Override
            public void run() {
                Toolkit toolkit = Toolkit.getDefaultToolkit();
                for(int i=0; i<5; i++) {
                    toolkit.beep();
                    try { Thread.sleep(500); } catch(Exception e) {}
                }
            }
        });*/
```

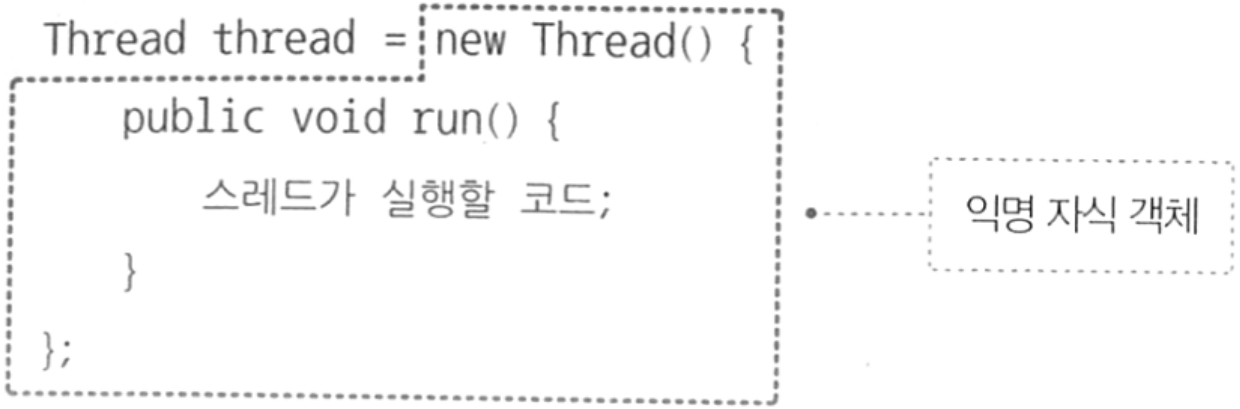
❖ BeepPrintExample2.java

```
        thread.start();

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

❖ 작업 스레드 생성 방법

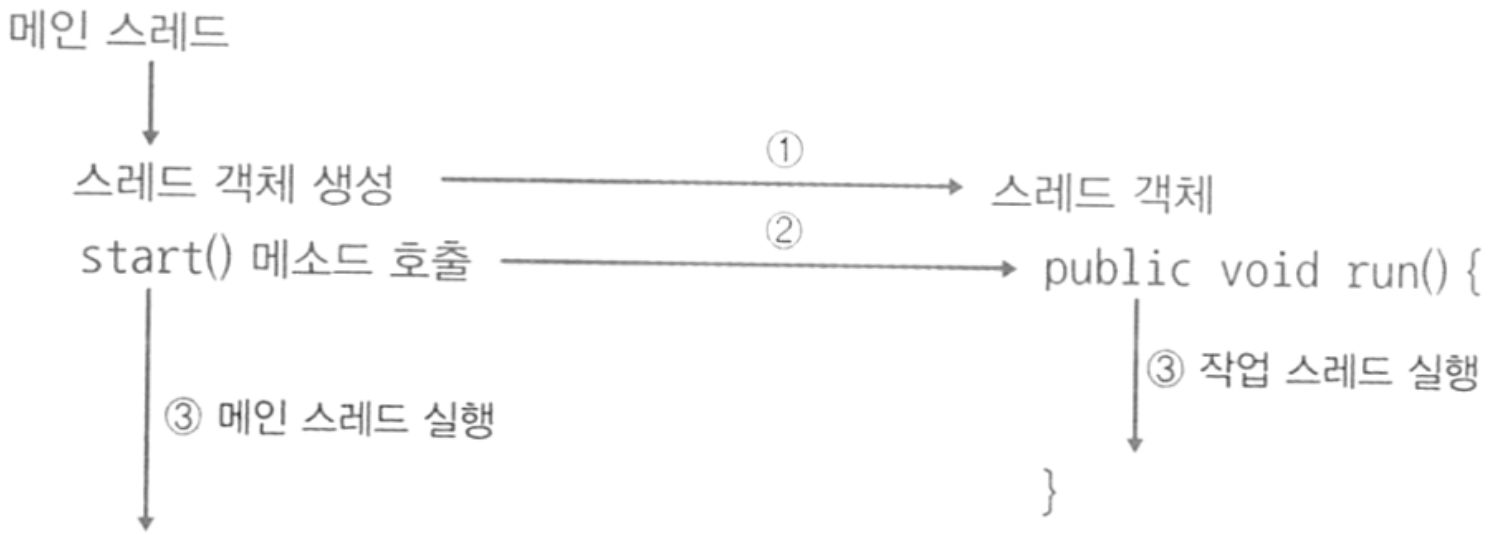
- Thread 하위 클래스로부터 생성
  - Thread 클래스 상속 후 run 메소드 재정의 해 스레드가 실행할 코드 작성

```
public class WorkerThread extends Thread {
    @Override
    public void run() {
        // 스레드가 실행할 코드
    }
}

Thread thread = new WorkerThread()
```



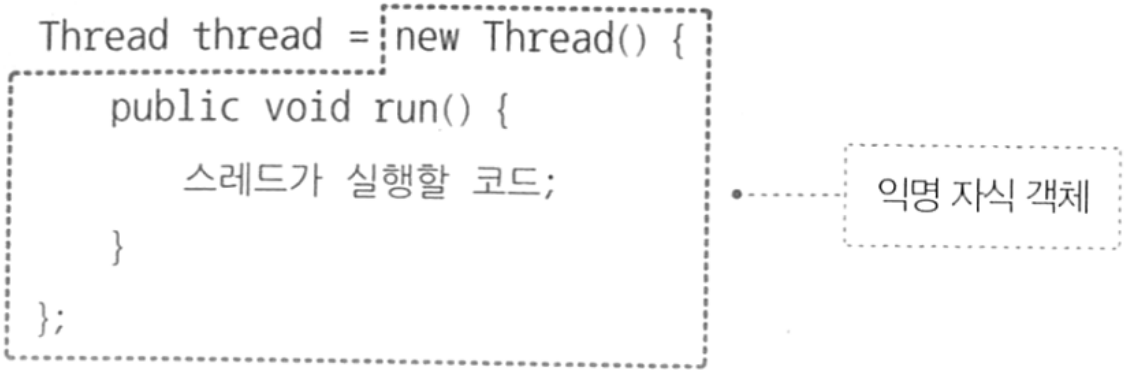
❖ 작업 스레드 생성 방법

- Thread 실행
  - Thread.start( );



❖ 작업 스레드 생성 방법

- Thread 하위 클래스로부터 생성




```
Thread thread = new Thread() {
    @Override
    public void run() {
        Toolkit toolkit = Toolkit.getDefaultToolkit();
        for(int i=0; i<5; i++) {
            toolkit.beep();
            try { Thread.sleep(500); } catch(Exception e) {}
        }
    }
};
```

❖ BeepThread.java

```
import java.awt.Toolkit;

public class BeepThread extends Thread {
    @Override
    public void run() {
        Toolkit toolkit = Toolkit.getDefaultToolkit();
        for(int i=0; i<5; i++) {
            toolkit.beep();
            try { Thread.sleep(500); } catch(Exception e) {}
        }
    }
}
```

❖ BeepPrintExample3.java

```
import java.awt.Toolkit;

public class BeepPrintExample3 {
    public static void main(String[] args) {
        //how1
        Thread thread = new BeepThread();

        thread.start();

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

❖ BeepPrintExample3.java

```
import java.awt.Toolkit;

public class BeepPrintExample3 {
    public static void main(String[] args) {
        // how 2
        Thread thread = new Thread() {
            @Override
            public void run() {
                Toolkit toolkit = Toolkit.getDefaultToolkit();
                for(int i=0; i<5; i++) {
                    toolkit.beep();
                    try { Thread.sleep(500); } catch(Exception e) {}
                }
            }
        };
        thread.start();

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

❖ 스레드의 이름

- 메인 스레드 이름: main
- 작업 스레드 이름 (자동 설정) : Thread-n

```
thread.getName();
```

- 작업 스레드 이름 변경

```
thread.setName("스레드 이름");
```

- 코드 실행하는 현재 스레드 객체의 참조 얻기

```
Thread thread = Thread.currentThread();
```

❖ ThreadA.java

```
public class ThreadA extends Thread {
    public ThreadA() {
        setName("ThreadA");
    }

    public void run() {
        for (int i = 0; i < 2; i++) {
            System.out.println(getName() + "가 출력한 내용");
        }
    }
}
```

❖ ThreadB.java

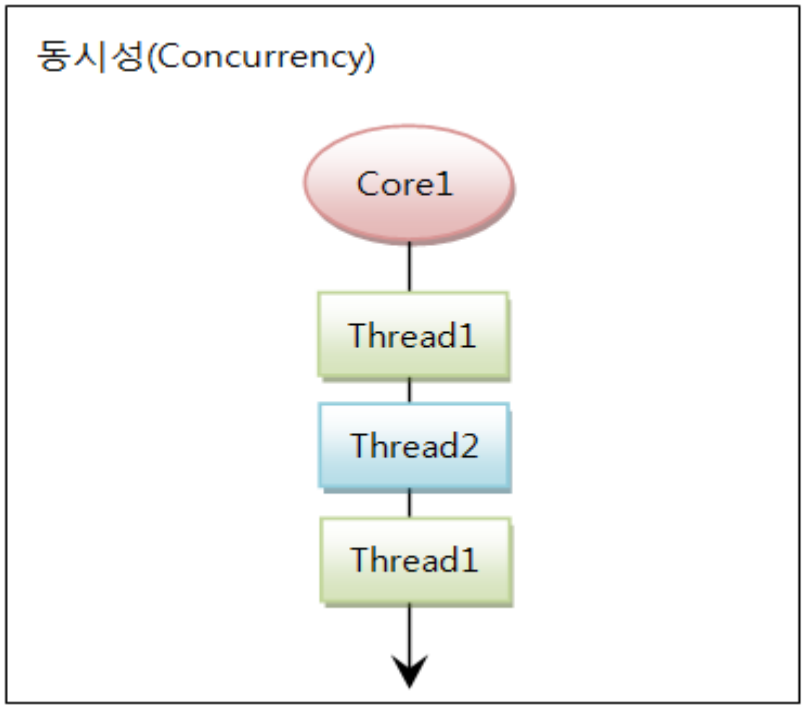
```
public class ThreadB extends Thread {
    public void run() {
        for(int i=0; i<2; i++) {
            System.out.println(getName() + "가 출력한 내용");
        }
    }
}
```

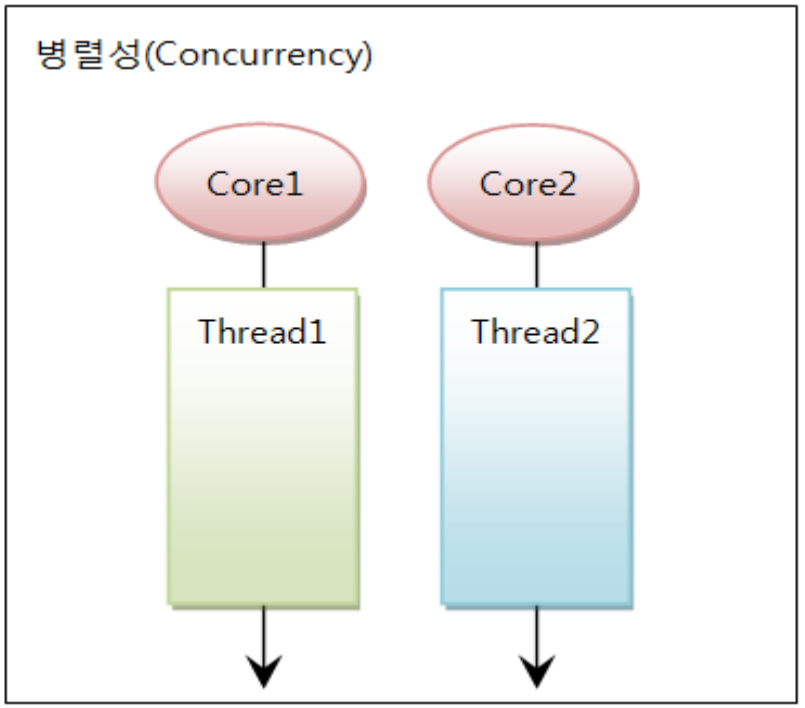
❖ ThreadNameExample.java

```
public class ThreadNameExample {
    public static void main(String[] args) {
        Thread mainThread = Thread.currentThread();
        System.out.println("프로그램 시작 스레드 이름: " +
                                                mainThread.getName());

        ThreadA threadA = new ThreadA();
        System.out.println("작업 스레드 이름: " + threadA.getName());
        threadA.start();

        ThreadB threadB = new ThreadB();
        System.out.println("작업 스레드 이름: " + threadB.getName());
        threadB.start();
    }
}
```

❖ 동시성과 병렬성

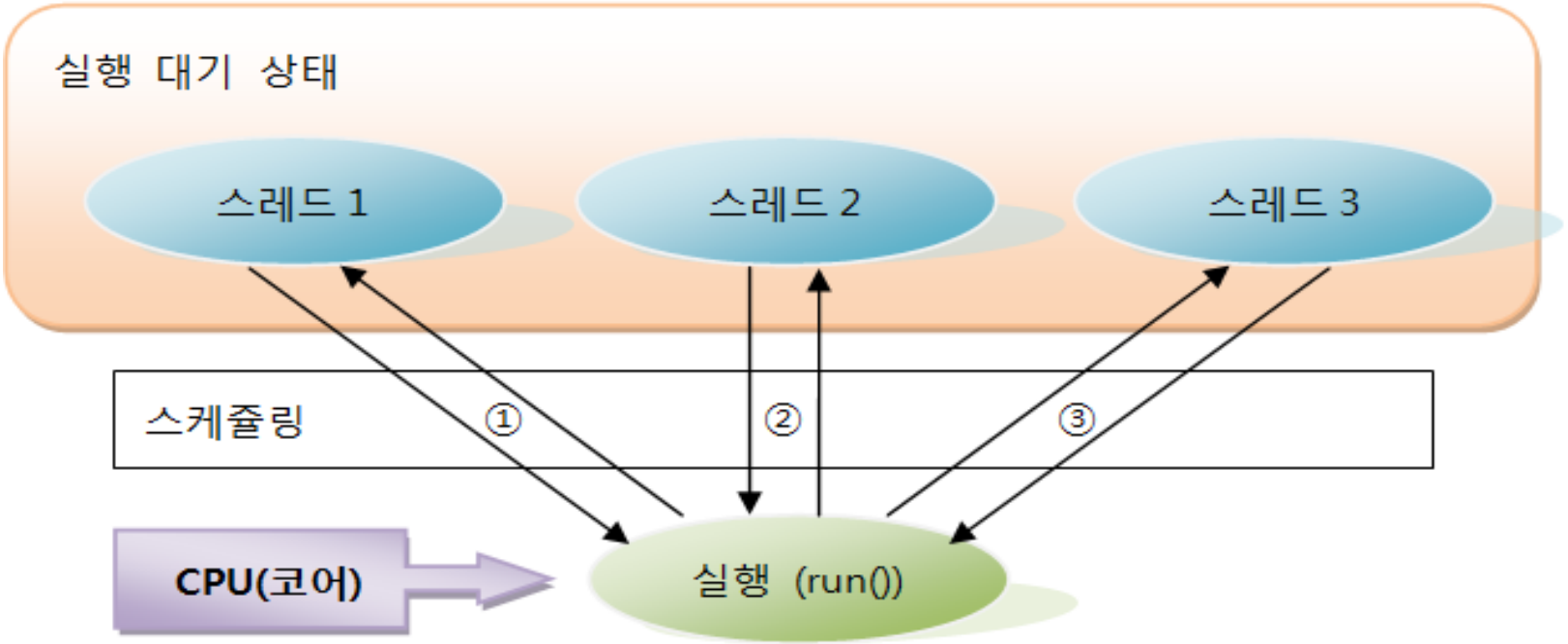
- ㅇ 동시성
  - ▪ 멀티 작업 위해 하나의 코어에서 멀티 스레드가 번갈아 가며 실행하는 성질
- ㅇ 병렬성
  - ▪ 멀티 작업을 위해 멀티 코어에서 개별 스레드를 동시에 실행하는 성질





❖ 스레드 스케줄링

- 스레드의 개수가 코어의 수보다 많을 경우
- 스레드를 어떤 순서로 동시성으로 실행할 것인가 결정 → 스레드 스케줄링
- 스케줄링 의해 스레드들은 번갈아 가며 run() 메소드를 조금씩 실행



스레드 제어

❖ 자바의 스레드 스케줄링

- 우선 순위(Priority) 방식과 순환 할당(Round-Robin) 방식 사용
- 우선 순위 방식 (코드로 제어 가능)
  - 우선 순위가 높은 스레드가 실행 상태를 더 많이 가지도록 스케줄링
  - 1~10까지 값을 가질 수 있으며 기본은 5
- 순환 할당 방식 (코드로 제어할 수 없음)
  - 시간 할당량(Time Slice) 정해서 하나의 스레드를 정해진 시간만큼 실행

```
thread.setPriority(우선순위);

thread.setPriority(Thread.MAX_PRIORITY);
thread.setPriority(Thread.NORM_PRIORITY);
thread.setPriority(Thread.MIN_PRIORITY);
```

❖ CalcThread.java

```
public class CalcThread extends Thread {
    public CalcThread(String name) {
        setName(name);
    }

    public void run() {
        for(int i=0; i<2000000000; i++) {
        }
        System.out.println(getName());
    }
}
```

❖ PriorityExample.java

```
public class PriorityExample {
    public static void main(String[] args) {
        for(int i=1; i<=10; i++) {
            Thread thread = new CalcThread("thread" + i);
            if(i != 10) {
                thread.setPriority(Thread.MIN_PRIORITY);
            } else {
                thread.setPriority(Thread.MAX_PRIORITY);
            }
            thread.start();
        }
    }
}
```