네트워크프로그래밍 기초 2

한성대학교 컴퓨터공학부

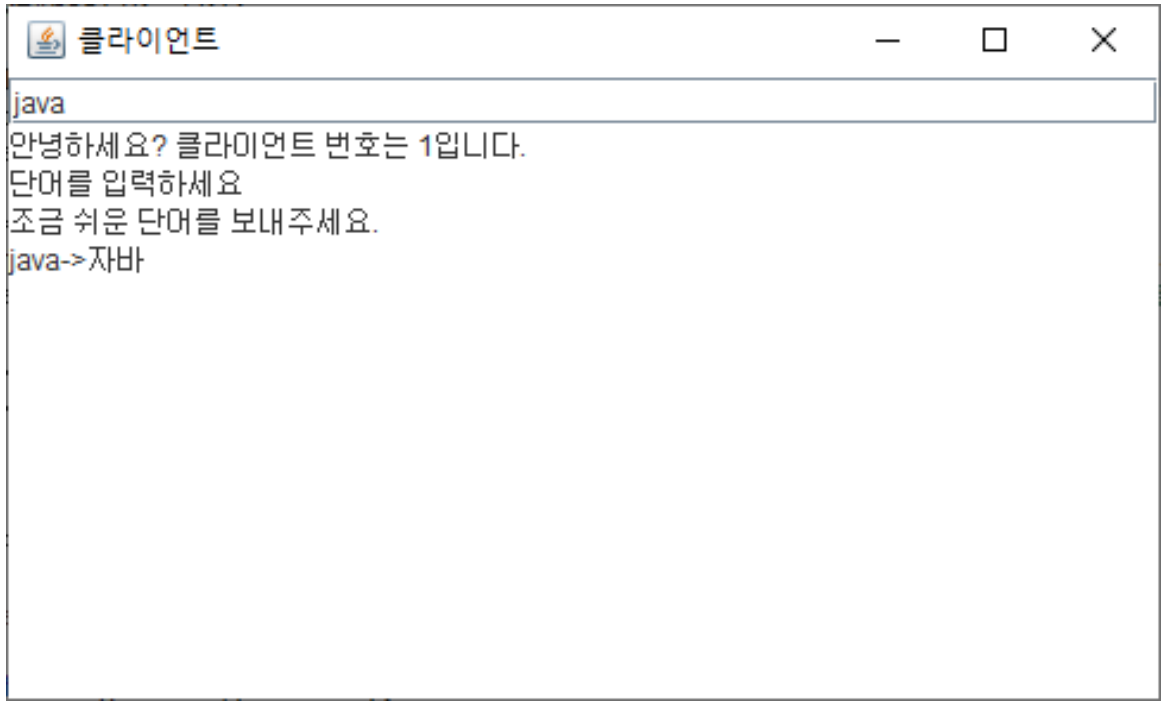
신 성

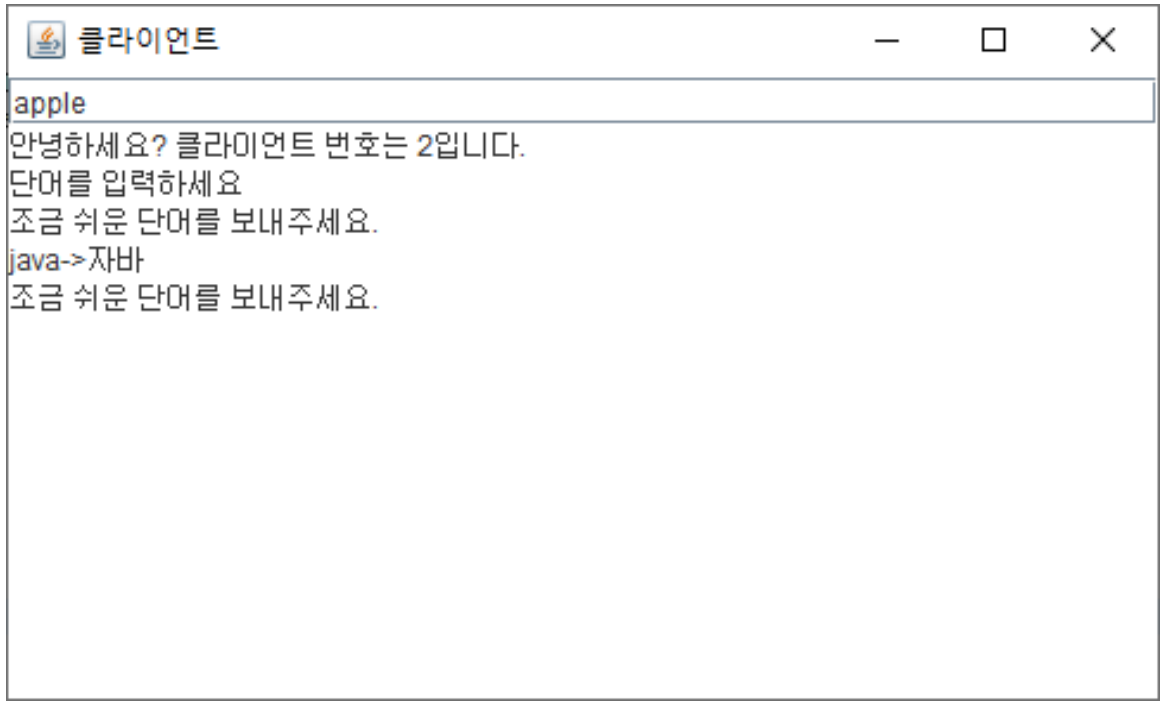
(간단한) 한글 번역 서버(다중 클라이언트)

- 사용자가 네트워크를 통하여 영어 단어를 보내면 한글로 번역하여 보내주는 서버를 구현

  java -> 자바
- 서버가 하나의 클라이언트만 처리하는 것이 아니고 여러 개의 클라이언트를 처리(다중 클라이언트)

  -> 각 클라이언트마다 스레드를 하나씩 생성하여 동시에 여러 클라이언트들에게 서비스를 제공하는 서버





(간단한) 한글 번역 서버(다중 클라이언트)

TranslationServer.java

서버는 1등으로 켜져야함
서버는 GUI 필요없음

클라 번호

accept가 보낸 객체는
1번 스레드에 넘겨지고,
Socket socket 이 받는다,
통신을 쉽게 해주는 함수와
1번 클라의 정보가 들어있다,
(응용프로그램 코드, IP 등...)
1번 클라와 통신할 수
있게 해주는 객체라고 생각하면 됨

accept는 서버를 구동시키고,
클라의 요청이 올 때까지 대기,
요청이 오면, 통신전 여러
테스트를 하고, 문제가 없으면
객체를 만든다

내부클래스, 밖 클래스의 멤버에 모두
접근 가능, setter 같은거 필요없음

```java
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.ServerSocket;
import java.net.Socket;

public class TranslationServer {

    public static void main(String[] args) throws Exception {
        System.out.println("영어 번역 서버가 실행중입니다.");
        int clientId = 0;
        ServerSocket ss = new ServerSocket(9101);
        try {
            while (true) {
                clientId++;
                Translator t = new Translator(ss.accept(), clientId);
                t.start();
            }
        } finally {
            ss.close();
        }
    }

    private static class Translator extends Thread {
        private Socket socket;
        private int myId;

        public Translator(Socket socket, int clientId) {
            this.socket = socket;
            this.myId = clientId;
        }

        public void run() {
            try {
                BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream())); // 소켓 입력 스트림
                BufferedWriter out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream())); // 소켓 출력 스트림

                out.write("안녕하세요? 클라이언트 번호는 " + myId + "입니다." + "\n");
                out.write("단어를 입력하세요" + "\n");
                out.flush();
                :
```

(간단한) 한글 번역 서버(다중 클라이언트)

TranslationClient.java 참고

```java
import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.Socket;

import javax.swing.JFrame;
import javax.swing.JTextArea;
import javax.swing.JTextField;

public class TranslationClient extends JFrame implements ActionListener {

    private BufferedReader in;
    private BufferedWriter out;
    private JTextField field;
    private JTextArea area;

    public TranslationClient() throws Exception, IOException {

        setTitle("클라이언트");
        setSize(500, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);

        field = new JTextField(50);
        field.addActionListener(this);
```

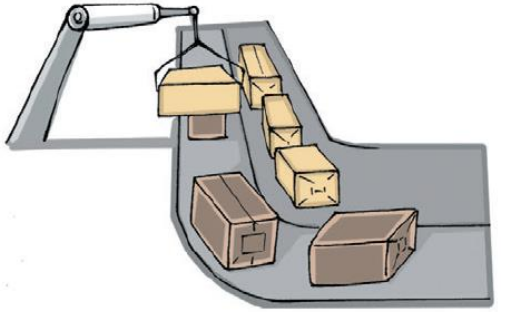
:

멀티태스킹(multi-tasking) 개념

- **멀티태스킹** isp
  - 하나의 응용프로그램이 여러 개의 작업(태스크)을 동시에 처리

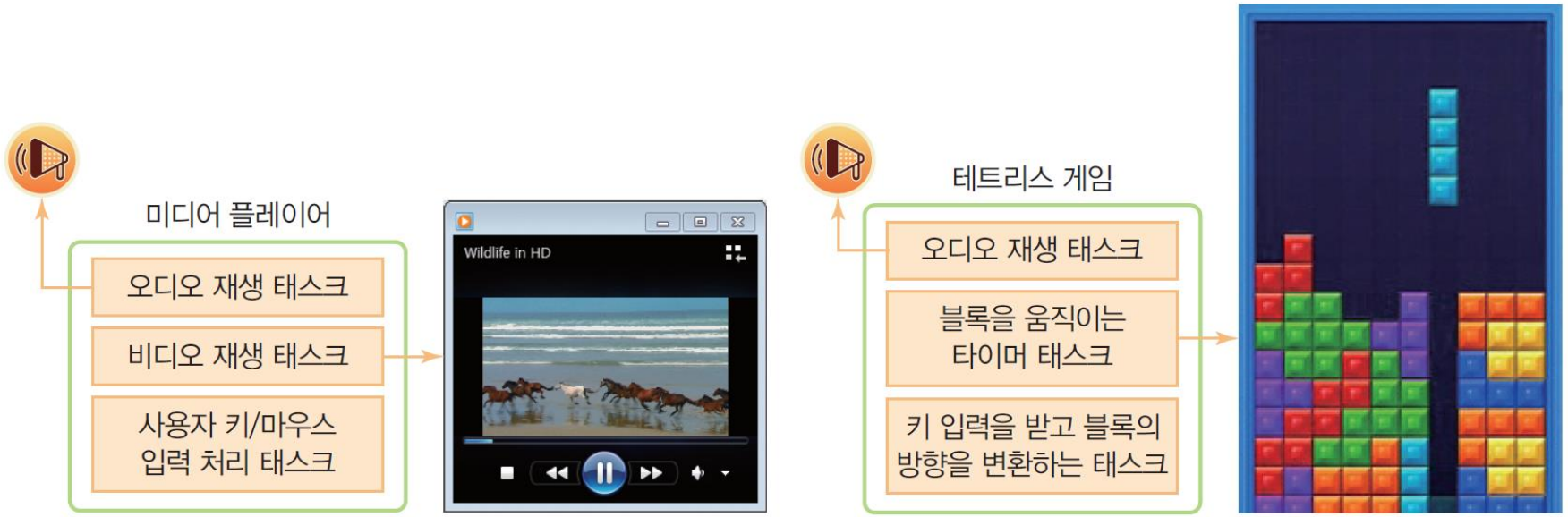
다림질하면서
전화 받는 주부

운전하면서 음악을
듣는 연수 양



판독과 포장을
동시에 하는 기계

멀티태스킹 응용프로그램 사례



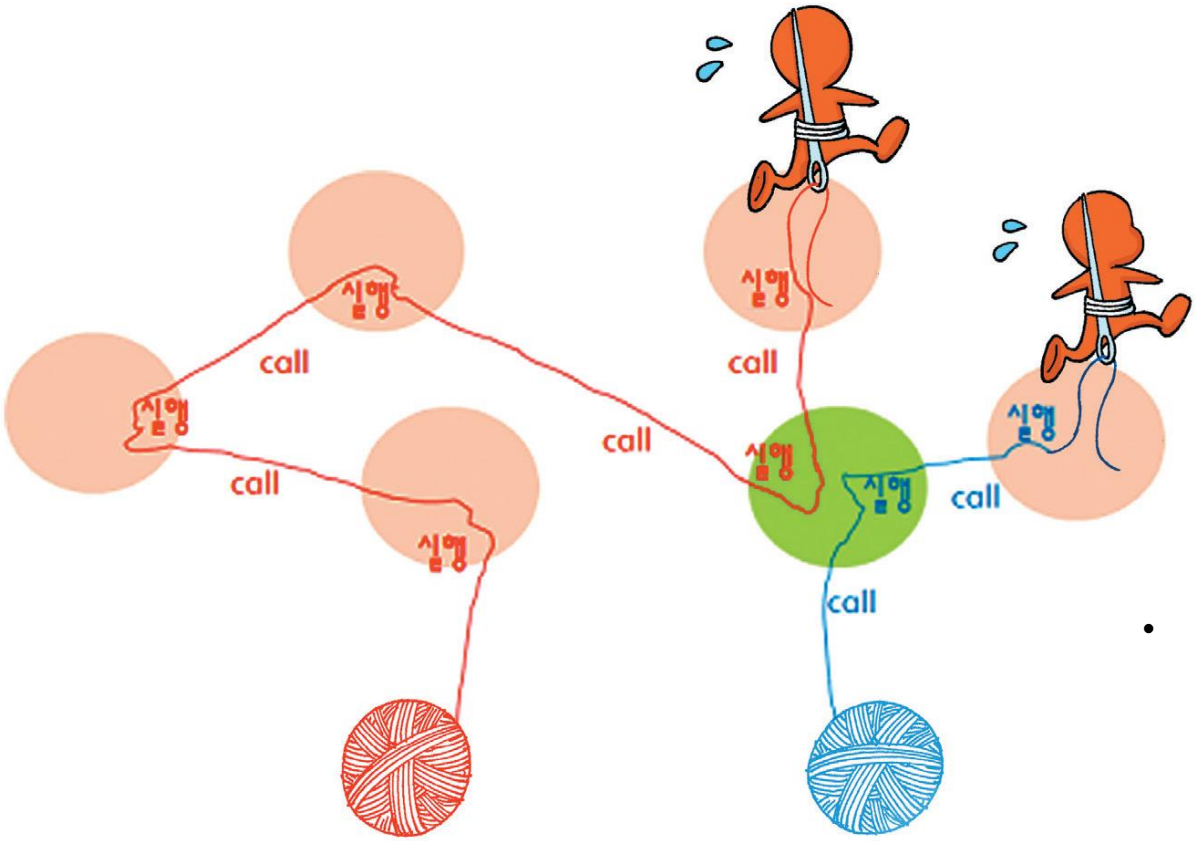
(a) 미디어 플레이어의 멀티태스킹

(b) 테트리스 게임의 멀티태스킹

* 3개의 태스크 동시 실행

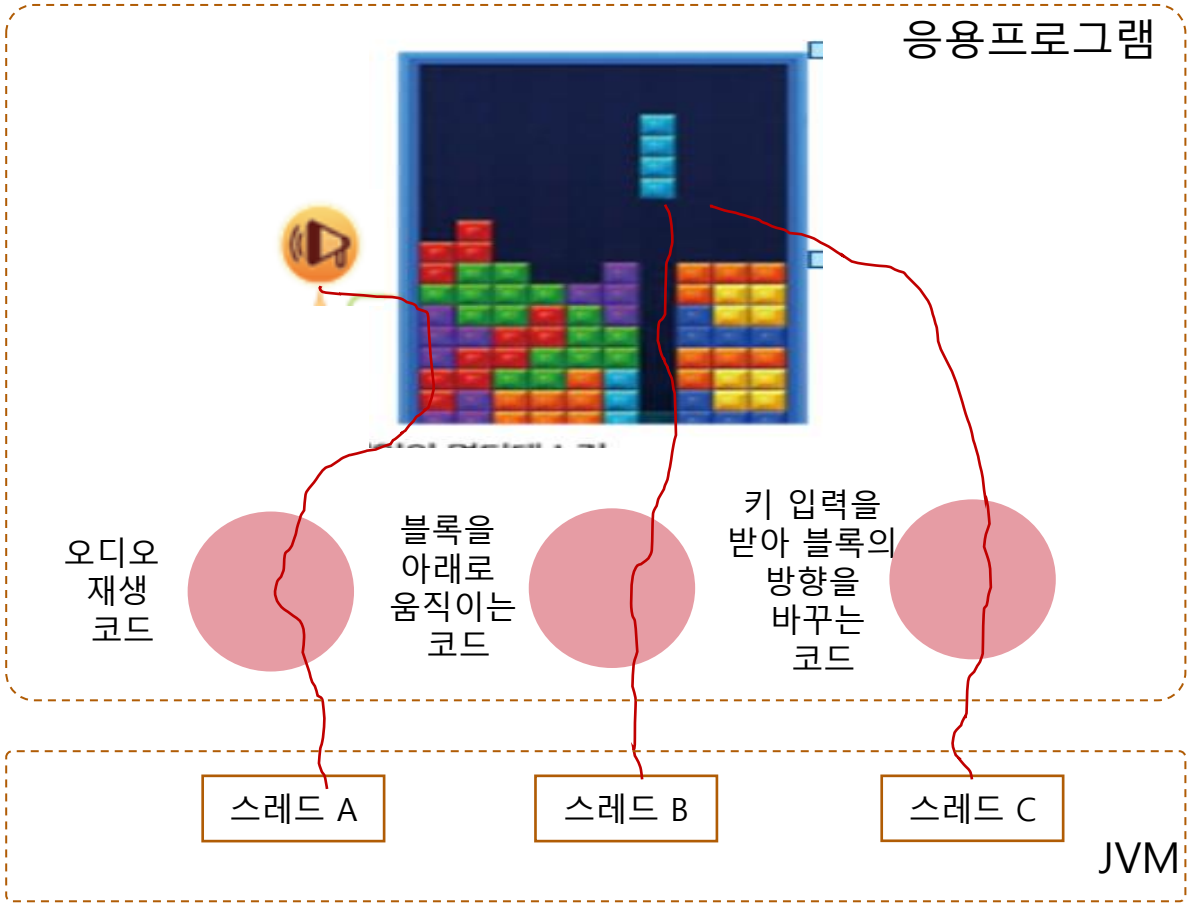
* 3개의 태스크 동시 실행

스레드(thread) 개념과 실(thread)



- 마치 바늘이 하나의 실(thread)을 가지고 바느질하는 것과 자바의 스레드는 일맥 상통함

테트리스 프로그램을 구성하는 멀티스레드 분석

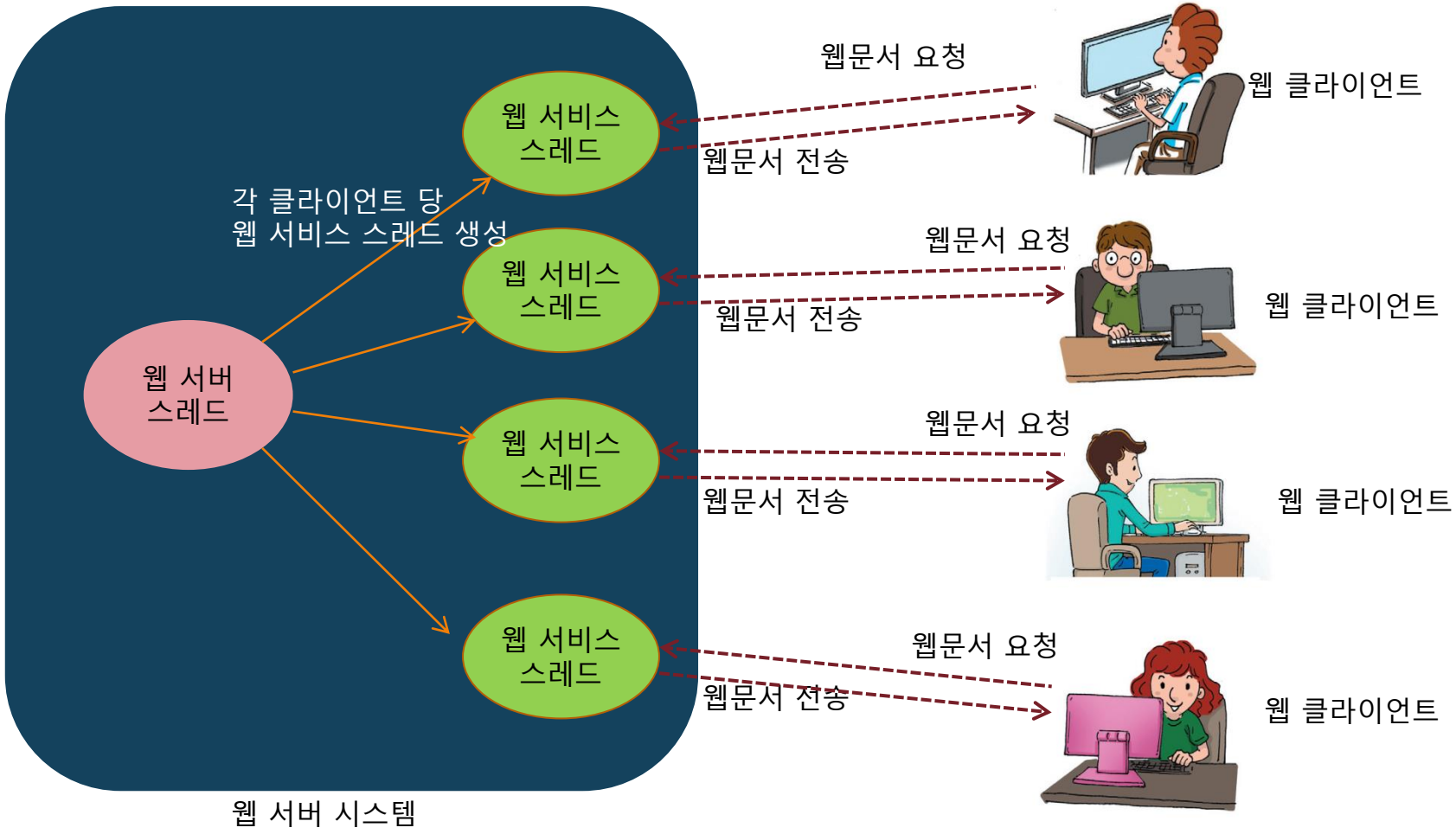


* JVM은 3 개의 스레드 중 하나를 선택하여 실행시킨다. 예를 들어 스레드 A를 선택하면 스레드의 A의 코드(사용자가 작성한 코드)를 호출한다. 스레드 A를 일시 중단하고, JVM이 스레드 B를 실행시켜려면 다시 스레드 B의 코드(사용자가 작성한 코드)를 호출한다.

스레드와 멀티스레딩

- **스레드**
  - 사용자가 작성한 코드로서, JVM에 의해 스케줄링되어 실행되는 단위
- **자바의 멀티태스킹**
  - 멀티스레딩만 가능
    - 스레드는 JVM의 의한 실행 단위, 스케쥴링 단위
  - 하나의 응용프로그램은 여러 개의 스레드로 구성 가능
- **멀티스레딩의 효과**
  - 한 스레드가 대기하는 동안 다른 스레드 실행
  - 프로그램 전체적으로 시간 지연을 줄임

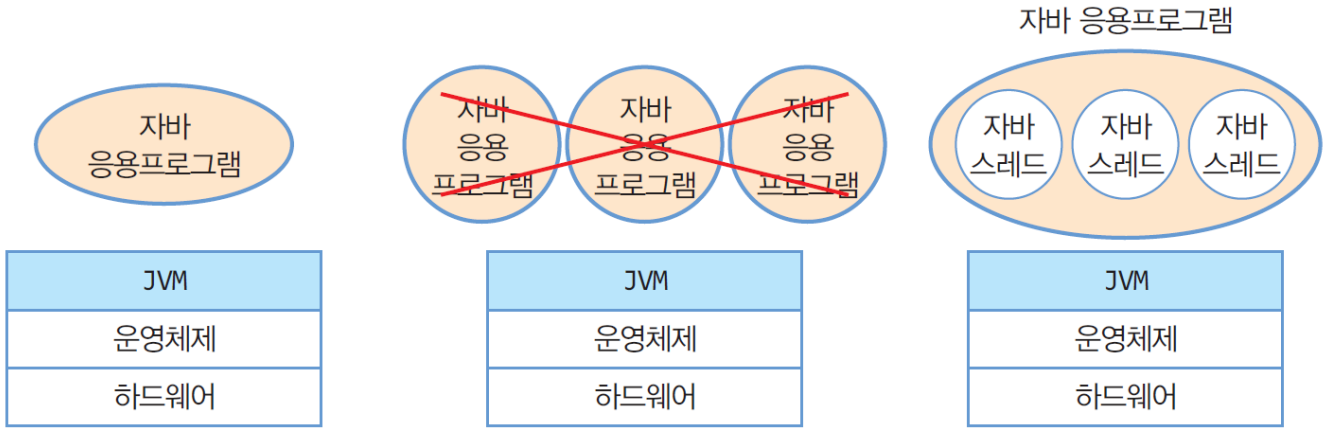
웹 서버의 멀티스레딩 사례

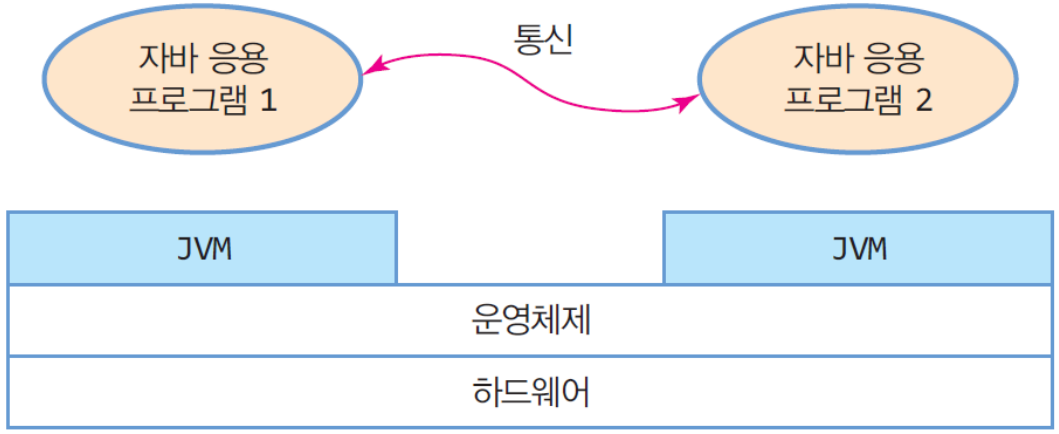


자바 스레드(Thread)란?

- **자바 스레드**
  - 자바 가상 기계(JVM)에 의해 스케쥴되는 실행 단위의 코드 블럭
  - 스레드의 생명 주기는 JVM에 의해 관리됨
    - JVM은 스레드 단위로 스케쥴링
- **JVM과 멀티스레드의 관계**
  - 하나의 JVM은 하나의 자바 응용프로그램만 실행
    - 자바 응용프로그램이 시작될 때 JVM이 함께 실행됨
    - 자바 응용프로그램이 종료하면 JVM도 함께 종료함
  - 하나의 응용프로그램은 하나 이상의 스레드로 구성 가능

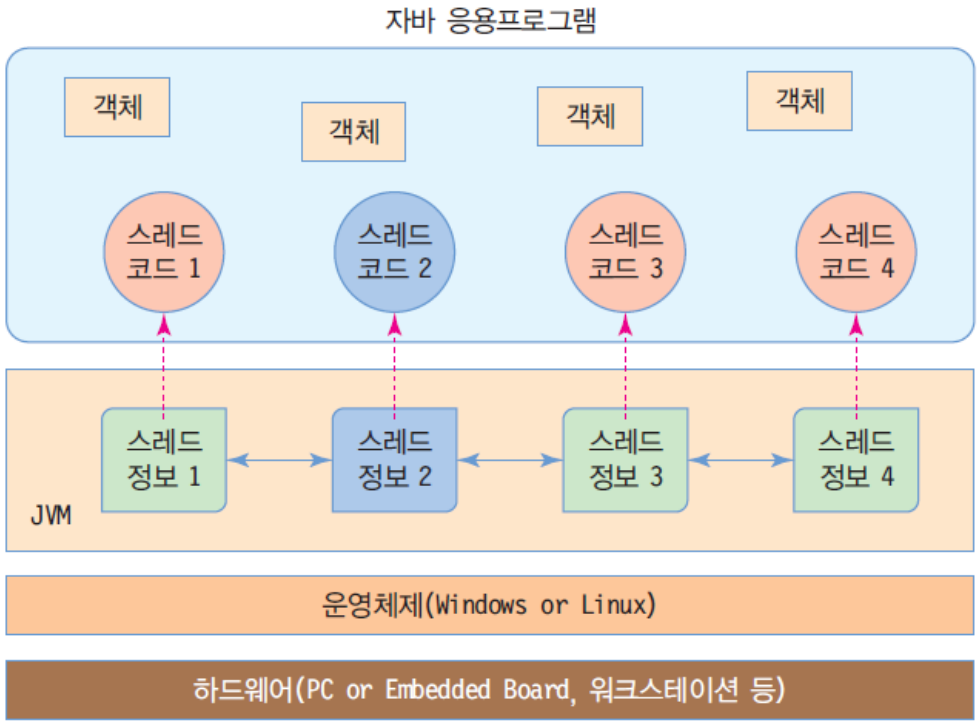
JVM과 자바 응용프로그램, 스레드의 관계





두 개의 자바 응용프로그램을
동시에 실행시키고자 하면
두 개의 JVM을 이용하고
응용프로그램은
서로 소켓 등을 이용하여 통신

자바 스레드와 JVM



현재 하나의 JVM에 의해 4 개의 스레드가 실행 중이며
그 중 스레드 2가 JVM에 의해 스케쥴링되어 실행되고 있음

- 각 스레드의 스레드 코드는 응용프로그램 내에 존재함

JVM이 스레드를 관리함
- 스레드가 몇 개인지?
- 스레드 코드의 위치가 어디인지?
- 스레드의 우선순위는 얼마인지?
- 등

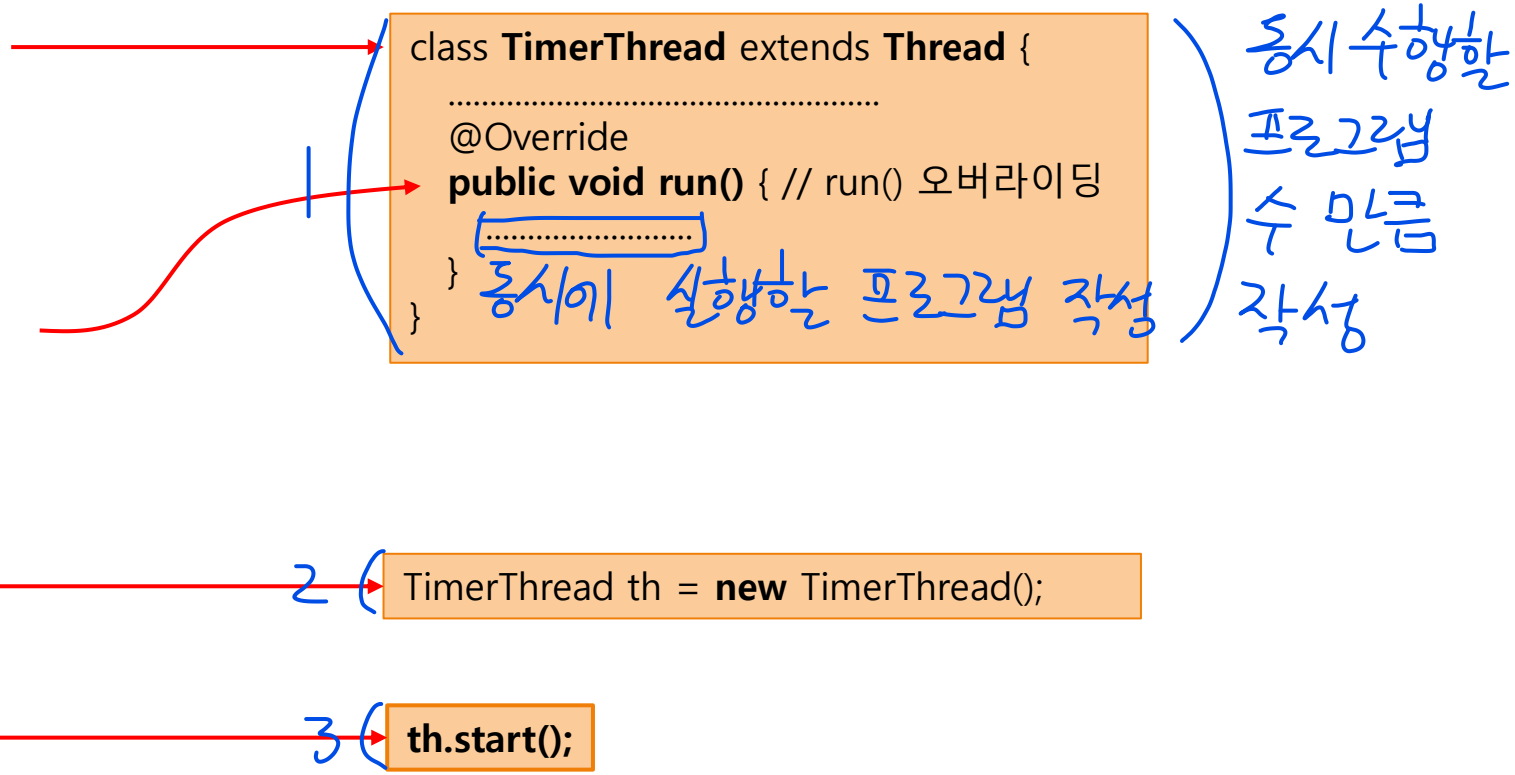
자바에서 스레드 만들기

- **스레드 실행을 위해 개발자가 하는 작업**
  - 스레드 코드 작성
  - 스레드를 생성하고 스레드 코드를 실행하도록 JVM에게 요청

- **스레드 만드는 2 가지 방법**
  - java.lang.Thread 클래스를 이용하는 경우
  - java.lang.Runnable 인터페이스를 이용하는 경우

Thread 클래스를 이용한 스레드 생성

- **스레드 클래스 작성**
  - Thread 클래스 상속. 새 클래스 작성
- **스레드 코드 작성**
  - run() 메소드 오버라이딩
    - run() 메소드를 스레드 코드라고 부름
    - run() 메소드에서 스레드 실행 시작
- **스레드 객체 생성**
- **스레드 시작**
  - start() 메소드 호출
    - 스레드로 작동 시작, JVM에 의해 스케쥴되기 시작함



* Thread를 상속받아 1초 단위로 초 시간을 출력하는 TimerThread 스레드 작성

스레드 클래스 정의

스레드 코드 작성

1초에 한 번씩 n을 증가시켜 콘솔에 출력한다.

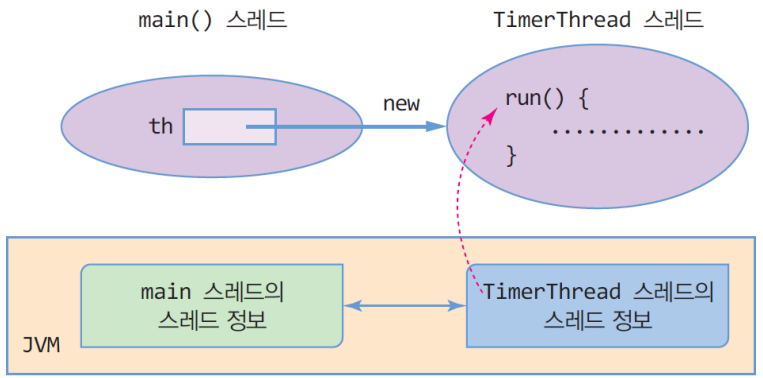
```
class TimerThread extends Thread {
    int n = 0;

    @Override
    public void run() {
        while(true) { // 무한루프를 실행한다.
            System.out.println(n);
            n++;
            try {
                sleep(1000); // 1초 동안 잠을 잔 후 깨어난다.
            }
            catch(InterruptedException e){return;}
        }
    }
}
```

```
0
1
2
3
4
.
.
.
.
```

스레드 객체 생성

스레드 시작

```
public class TestThread {
    public static void main(String [] args) {
        TimerThread th = new TimerThread();
        th.start();
    }
}
```



예제 13-1 : Thread를 상속받아 1초 단위의 타이머 만들기

```
import java.awt.*;
import javax.swing.*;

class TimerThread extends Thread {
    private JLabel timerLabel;

    public TimerThread(JLabel timerLabel) {
        this.timerLabel = timerLabel;
    }

    @Override
    public void run() {
        int n=0;
        while(true) {
            timerLabel.setText(Integer.toString(n));
            n++;
            try {
                Thread.sleep(1000);
            }
            catch(InterruptedException e) {
                return;
            }
        }
    }
}
```

```
public class ThreadTimerEx extends JFrame {
    public ThreadTimerEx() {
        setTitle("Thread를 상속받은 타이머 스레드 예제");
        setDefaultCloseOperation(Jframe.EXIT_ON_CLOSE);
        Container c = getContentPane();
        c.setLayout(new FlowLayout());

        JLabel timerLabel = new JLabel();
        timerLabel.setFont(new Font("Gothic", Font.ITALIC, 80));
        c.add(timerLabel);

        TimerThread th = new TimerThread(timerLabel);

        setSize(300,170);
        setVisible(true);

        th.start();
    }
    public static void main(String[] args) {
        new ThreadTimerEx();
    }
}
```

스레드 만들 때 주의 사항

- **run() 메소드가 종료하면 스레드는 종료한다.**
  - 스레드가 계속 살아있게 하려면 run() 메소드 내 무한루프 작성
- **한번 종료한 스레드는 다시 시작시킬 수 없다.**
  - 다시 스레드 객체를 생성하고 start()를 호출해야 함
- **한 스레드에서 다른 스레드를 강제 종료할 수 있다.**
  - 뒤에서 다룸

Q&A

담당교수 : 신성

E-mail : sihns@hansung.ac.kr

연구실 : 우촌관 702호

휴대폰 번호 : 010-8873-8353