Java Serial Communication

라이브러리 설치(x86) - Java 기본 컴파일러

- 아래 파일들을 해당 위치에 복사한다.
  - <JAVA_HOME> : C:₩Program Files₩Java₩jdkx.x.x_xx

- win32com.dll → <JAVA_HOME>₩bin
- javax.comm.properties → <JAVA_HOME>₩jre₩lib
- comm.jar → <JAVA_HOME>₩jre₩lib₩ext

라이브러리 설치(x86) - eclipse

- 아래 파일들을 해당 위치에 복사한다.
  - <JAVA_HOME> : C:₩Program Files₩Java₩jre6

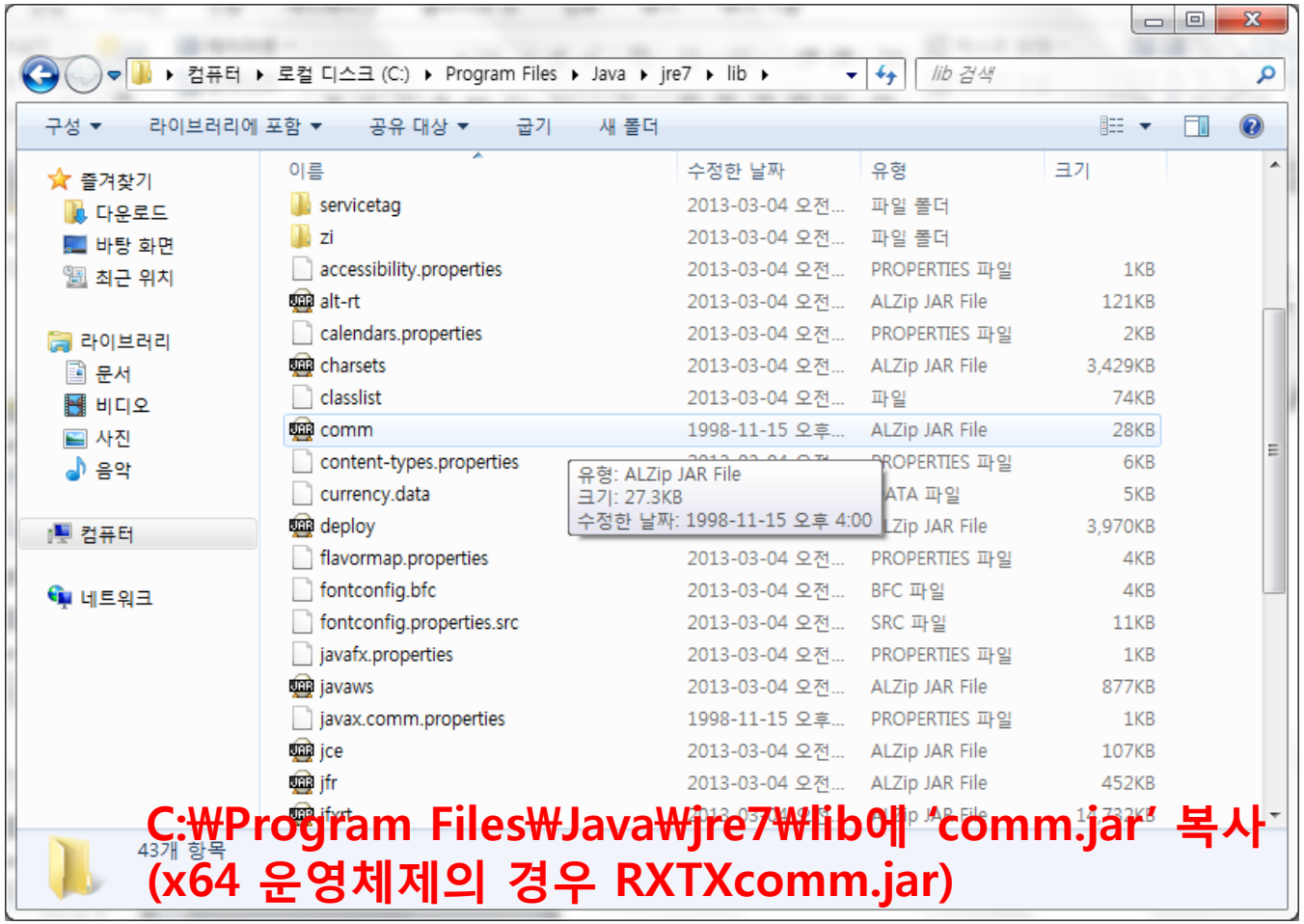
- win32com.dll → <JAVA_HOME>₩bin
- javax.comm.properties → <JAVA_HOME>₩lib
- comm.jar → <JAVA_HOME>₩lib₩ext

라이브러리 설치(x64)

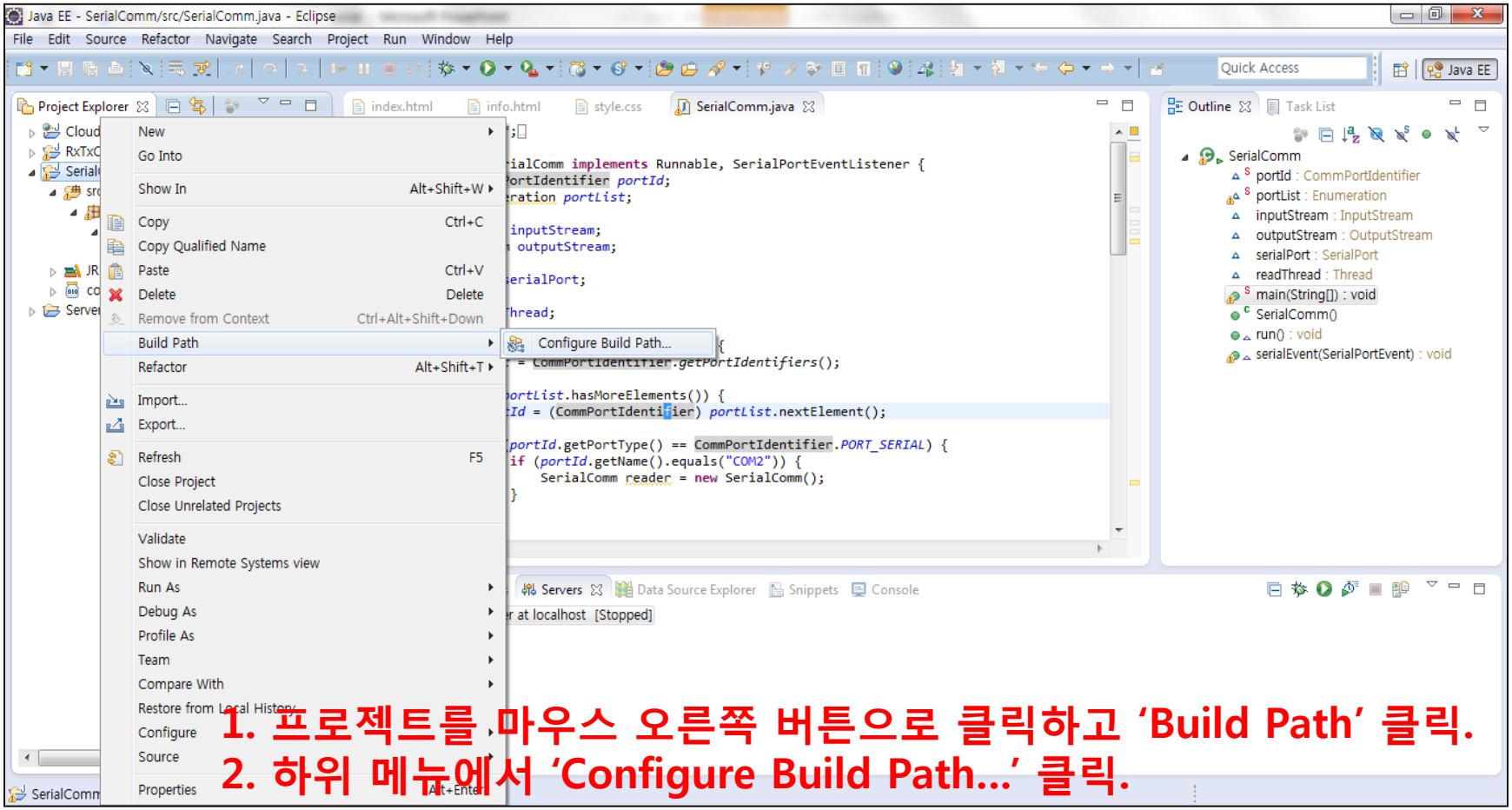
- 아래 파일들을 해당 위치에 복사한다.
  - <JAVA_HOME> : C:₩Program Files₩Java₩jdkx.x.x_xx

- RXTXcomm.jar → <JAVA_HOME>₩jre₩lib₩ext
- rxtxSerial.dll → <JAVA_HOME>₩jre₩bin
- rxtxParallel.dll → <JAVA_HOME>₩jre₩bin

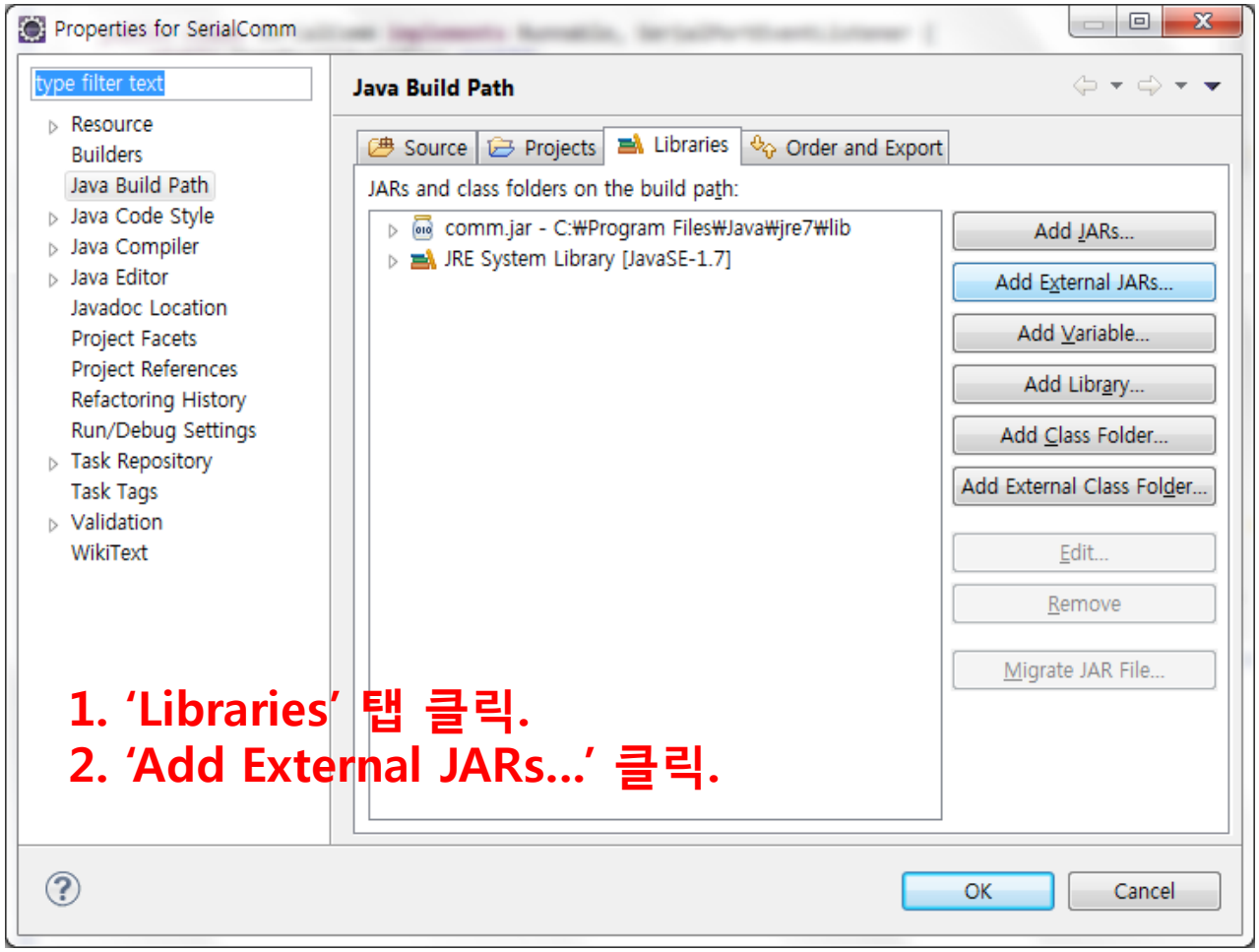
Build Path에 .jar 추가 (1/5)



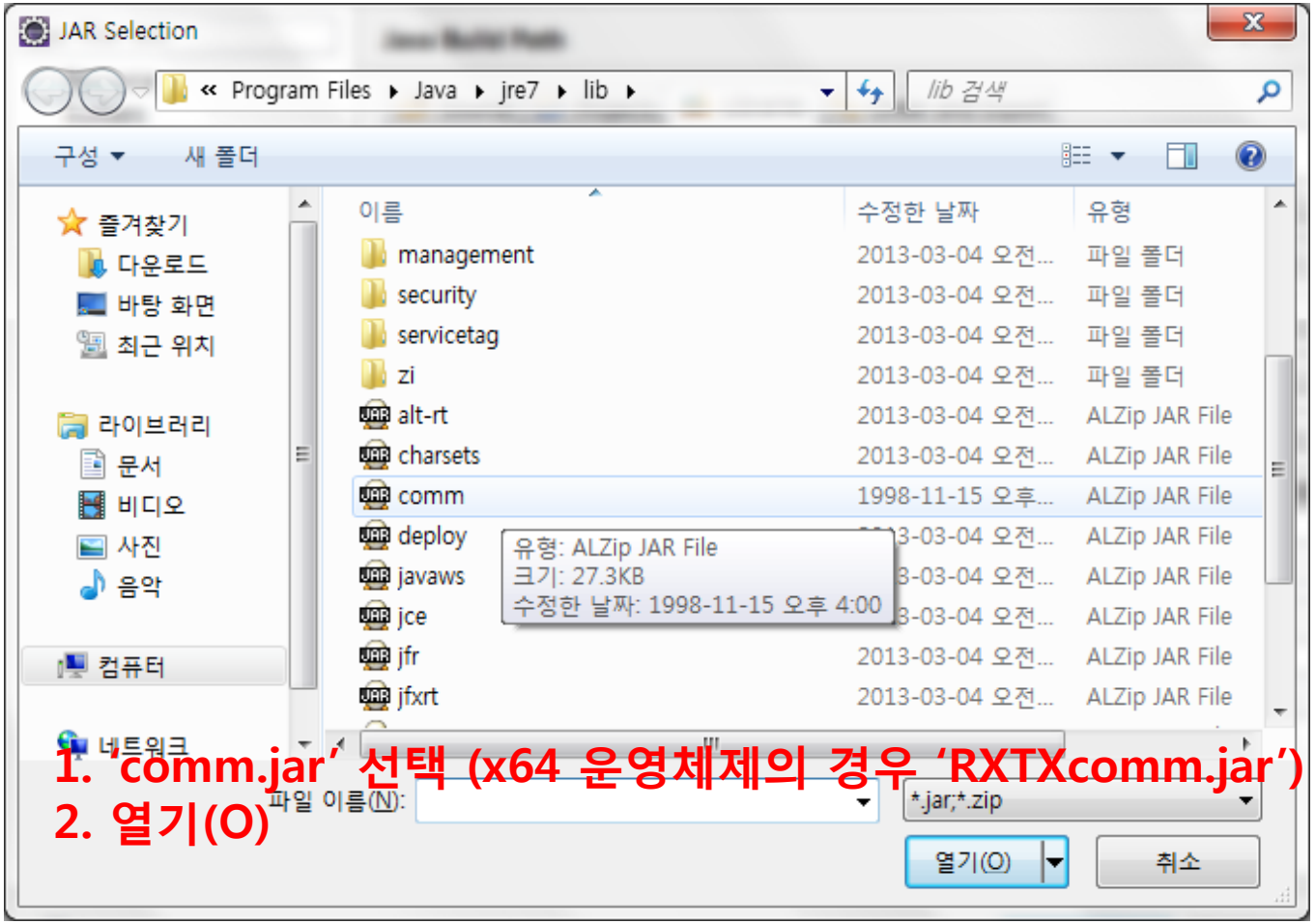
Build Path에 .jar 추가 (2/5)



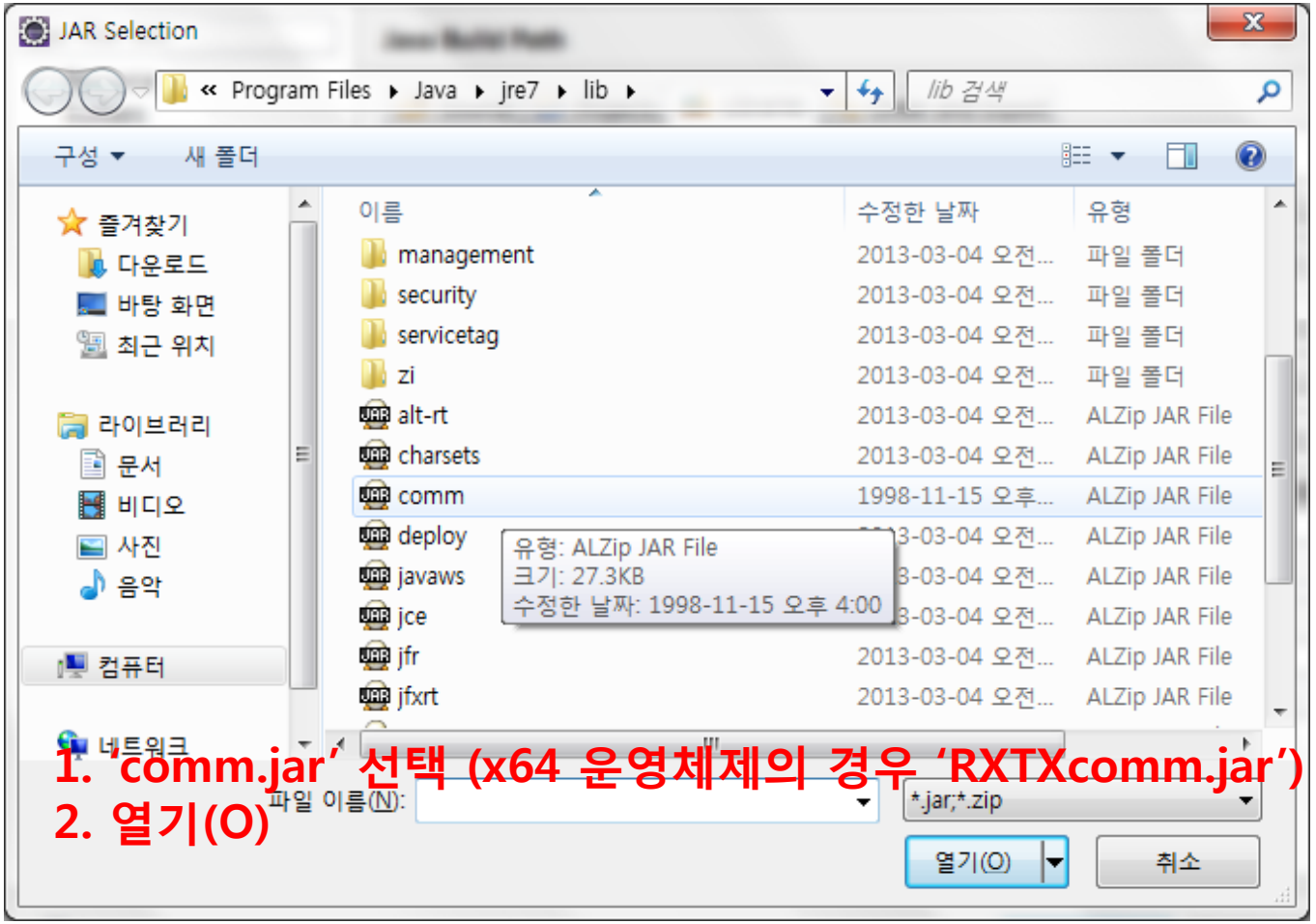
Build Path에 .jar 추가 (3/5)



Build Path에 .jar 추가 (4/5)



Build Path에 .jar 추가 (5/5)



SimpleRead (1/9)

```
import java.io.*;
import java.util.*;
import javax.comm.*;        // x64 운영체제의 경우 import gnu.io.*;

public class SimpleRead implements Runnable, SerialPortEventListener {
    static CommPortIdentifier portId;
    static Enumeration portList;

    InputStream inputStream;
    SerialPort serialPort;
    Thread readThread;
```

SimpleRead (2/9)

```
public static void main(String[] args) {
    // 시스템에 있는 가능한 드라이버의 목록을 받아온다.
    portList = CommPortIdentifier.getPortIdentifiers();

    // enumeration type 인 portList 의 모든 객체에 대하여
    while (portList.hasMoreElements()) {
        // enumeration 에서 객체를 하나 가져온다.
        portId = (CommPortIdentifier) portList.nextElement();
        // 가져온 객체의 port type 이 serial port 이면
        if (portId.getPortType() ==
                CommPortIdentifier.PORT_SERIAL) {
```

SimpleRead (3/9)

```
            if (portId.getName().equals("COM2")) {
                                                  ↑
            // Linux 의 경우     자신의 컴퓨터의 시리얼 포트 번호로 변경
               if (portId.getName().equals("/dev/term/a"))

                // 객체 생성
                SimpleRead reader = new SimpleRead();
            }
        }
    }
}
```

SimpleRead (4/9)

```
// SimpleRead 생성자
public SimpleRead() {
    try {
        /* 사용 메소드 :
           public CommPort open(java.lang.String appname, int timeout)
           기능 :
           어플리케이션 이름과 타임아웃 시간 명시 */
        serialPort = (SerialPort) portId.open("SimpleReadApp", 2000);
    } catch (PortInUseException e) { }
    try {
        // 시리얼 포트에서 입력 스트림을 획득한다.
        inputStream = serialPort.getInputStream();
    } catch (IOException e) { }
```

SimpleRead (5/9)

```
// 시리얼 포트의 이벤트 리스너로 자신을 등록한다.
try {
    serialPort.addEventListener(this);
} catch (TooManyListenersException e) { }

/* 시리얼 포트에 데이터가 도착하면 이벤트가 한 번 발생되는데
   이 때, 자신이 리스너로 등록된 객체에게 이벤트를 전달하도록 허용. */
serialPort.notifyOnDataAvailable(true);

// 시리얼 통신 설정. Data Bit는 8, Stop Bit는 1, Parity Bit는 없음.
try {
    serialPort.setSerialPortParams(9600,
            SerialPort.DATABITS_8, SerialPort.STOPBITS_1,
            SerialPort.PARITY_NONE);
} catch (UnsupportedCommOperationException e) { }
```

SimpleRead (6/9)

```
        // 쓰레드 객체 생성
        readThread = new Thread(this);

        // 쓰레드 동작
        readThread.start();
}
```

SimpleRead (7/9)

```
    public void run() {
    try {
        Thread.sleep(20000);
    } catch (InterruptedException e) { }
}
```

SimpleRead (8/9)

```
// 시리얼 포트 이벤트가 발생하면 호출. 시리얼 포트 이벤트를 전달한다.
public void serialEvent(SerialPortEvent event) {
    // 이벤트의 타입에 따라 switch 문으로 제어.
    switch (event.getEventType()) {
    case SerialPortEvent.BI:
    case SerialPortEvent.OE:
    case SerialPortEvent.FE:
    case SerialPortEvent.PE:
    case SerialPortEvent.CD:
    case SerialPortEvent.CTS:
    case SerialPortEvent.DSR:
    case SerialPortEvent.RI:
    case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
        break;
```

SimpleRead (9/9)

```
        // 데이터가 도착하면
        case SerialPortEvent.DATA_AVAILABLE:
            byte[] readBuffer = new byte[1];    // byte 배열 객체 생성

            // 입력 스트림이 사용가능하면, 버퍼로 읽어 들인 후
            // String 객체로 변환하여 출력
            try {
                while (inputStream.available() > 0) {
                    int numBytes =
                            inputStream.read(readBuffer);
                }
                System.out.print(new String(readBuffer));
            } catch (IOException e) { }
            break;
        }
    }
}
```

SimpleWrite (1/4)

```
import java.io.*;
import java.util.*;
import javax.comm.*;

public class SimpleWrite {
    static Enumeration portList;
    static CommPortIdentifier portId;
    static String messageString = "Hello, world!\n";
    static SerialPort serialPort;
    static OutputStream outputStream;
```

SimpleWrite (2/4)

```
public static void main(String[] args) {
    portList = CommPortIdentifier.getPortIdentifiers();

    while (portList.hasMoreElements()) {
        portId = (CommPortIdentifier) portList.nextElement();
        if (portId.getPortType() ==
                        CommPortIdentifier.PORT_SERIAL) {
           if (portId.getName().equals("COM1")) {
          //if (portId.getName().equals("/dev/term/a")) {
                try {
                    serialPort = (SerialPort)
                        portId.open("SimpleWriteApp", 2000);
                } catch (PortInUseException e) {}
```

SimpleWrite (3/4)

```
try {
    outputStream = serialPort.getOutputStream();
} catch (IOException e) { }
try {
    serialPort.setSerialPortParams(9600,
                SerialPort.DATABITS_8, SerialPort.STOPBITS_1,
              SerialPort.PARITY_NONE);
} catch (UnsupportedCommOperationException e) { }
```

SimpleWrite (4/4)

```
                    try {
                        outputStream.
                                write(messageString.getBytes());
                    } catch (IOException e) {}
                }
            }
        }
    }
}
```

과제

- SimpleWrite와 SimpleRead를 참고하여 양방향 채팅이 가능한 프로그램을 만들어 제출하세요.
- 2인 1조도 좋고, 1인이 해도 좋습니다.
- 단, 2인 1조로 할 경우 제출을 각자 해 주시길 바랍니다.
- 제출 일시 : 2014.04.20 23:59까지