

# Доступ к COM порту из Java приложения

Владислав Каменский

Эта статья посвящена обзору пакета `java.com`, позволяющего получить доступ с COM порту компьютера. Данный пакет не входит в стандартный набор JDK.

Итак, сформулируем задачу. Для примера возьмем обычный модем, который проинсталлирован на стандартный COM порт, нам необходимо, что бы он откликнулся на терминальные команды (AT, ATDP и т.д.), посылаемые ему из java приложения.

Для того, чтобы мы могли написать такое приложение, нам необходимо скачать пакет классов `java.com` (см.[2]). После чего, необходимо совершить ряд манипуляций, чтобы написанные нами приложения могли корректно работать.

## 1. Инсталляция Java Communications API для JDK 1.2 (Windows)

<jdk> корневой каталог JDK. Если вы проинсталлировали JDK в каталог `c:\jdk1.2` тогда замените все ссылки <jdk> на `c:\jdk1.2`.

1. Положите файл `win32com.dll` в директорию `<jdk>\jre\bin`.
2. Положите архив `comm.jar` в директорию `<jdk>\jre\lib\ext`.
3. Положите файл `java.com.properties` в `<jdk>\jre\lib`.
4. Не изменяйте переменную `CLASSPATH`.

### Замечание:

В настоящее время существуют реализации пакета для следующих платформ Windows, Solaris, Linux. (см. [2]).

Только после проделанной работы команды `javac` и `java`, смогут правильно работать.

**Замечание:**

Если вы используете версию JDK ниже, чем 1.2 то вам придется по другому проинсталлировать данный пакет (см. [2])

Рассмотрим следующий фрагмент кода:

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

. . .

public static void main(String[] args) {
    Enumeration portList = CommPortIdentifier.getPortIdentifiers();
    while (portList.hasMoreElements()) {
        CommPortIdentifier portId =
            (CommPortIdentifier) portList.nextElement();
        if (portId.getPortType() ==
            CommPortIdentifier.PORT_SERIAL) {
            if (portId.getName().equals("COM2")) {
                Terminal terminal = new Termianl(portID);
            }
        }
    }
}
```

## 2. Класс CommPortIdentifier

Приведем описание переменных и методов класса `CommPortIdentifier`:

### 2.1. Переменные класса:

- `PORT_SERIAL` `public static final int PORT_SERIAL`  
RS-232 последовательный порт
- `PORT_PARALLEL` `public static final int PORT_PARALLEL`  
IEEE 1284 параллельный порт

### 2.2. Методы Класса:

## Доступ к COM порту из Java приложения

- `public static Enumeration getPortIdentifiers()`  
Метод возвращает объект типа `enumeration`, который содержит объекты типа `CommPortIdentifier` для каждого порта системы.
- `public String getName()`  
Возвращает имя порта. Например, "COM1" и "COM2" на PC;
- `public int getPortType()`  
Метод возвращает тип порта `PORT_SERIAL` или `PORT_PARALLEL`.
- `public synchronized CommPort open(String appname, int timeout) throws PortInUseException`  
Открывает порт. Возвращает объект типа `CommPort`. параметры:
  - `appname` - Имя приложения вызывающего данный метод. (произвольная строка)
  - `timeout` - время в миллисекундах, в течение которого, блокируется доступ к порту для его открытия.Throws: `PortInUseException` если порт используется другим приложением.

### 3. Пример использования: класс Terminal

Таким образом, в методе `main`, мы получили полный перечень идентификаторов портов нашего компьютера, после чего отобрали идентификаторы последовательных портов. Ну, и наконец, выбрали идентификатор COM2 порта (предположим что модем находится именно на нем) и передали его конструктору класса `Terminal`. Следующим шагом будет написание класса `Terminal`, который бы смог отправлять сообщения в COM порт, и прослушивал бы сообщения поступающие из этого порта.

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

public class Terminal implements Runnable,
    SerialPortEventListener {
    InputStream inputStream;
    OutputStream outputStream;
    SerialPort serialPort;
    Thread readThread;
    String[] messageString = {"AT\n", "ATI1\n", "ATI3\n"};
public Terminal() {
    try {
        serialPort = (SerialPort) portId.open("TerminalApp", 2000);
```

```
} catch (PortInUseException e) {}
    try {
outputStream = serialPort.getOutputStream();
inputStream = serialPort.getInputStream();
    } catch (IOException e) {}
    try {
        serialPort.addEventListener(this);
    } catch (TooManyListenersException e) {}
    serialPort.notifyOnDataAvailable(true);
    try {
        // устанавливаем параметры порта
        serialPort.setSerialPortParams(9600,
            SerialPort.DATABITS_8,
            SerialPort.STOPBITS_1,
            SerialPort.PARITY_NONE);
    } catch (UnsupportedCommOperationException e) {}
    readThread = new Thread(this);
    readThread.start();
}
public void run() {
    for(int i=0;i<3;i++){
        try {
            outputStream.write(messageString[i].getBytes());
        } catch (IOException e) {}
        try{
            Thread.sleep(5000);
        } catch (InterruptedException e) {}
    }
    System.exit(1); // ВЫХОД ИЗ ПРОГРАММЫ
}
public void serialEvent(SerialPortEvent event) {
    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:
```

## Доступ к COM порту из Java приложения

```
        case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
break;
        case SerialPortEvent.DATA_AVAILABLE:
            byte[] readBuffer = new byte[20];
            try {
                while (inputStream.available() > 0) {
                    int numBytes = inputStream.read(readBuffer);
                }
                System.out.print(new String(readBuffer));
            } catch (IOException e) {}
            break;
        }
    }
}
```

Класс `Terminal` реализует интерфейс `SerialPortEventListener`, это подразумевает включение одного единственного метода `public void serialEvent(SerialPortEvent event){}`. Рассмотрим используемые нами методы класса `SerialPort`:

- `public abstract InputStream getInputStream() throws IOException`  
Возвращает поток `InputStream`. Это один единственный способ получить данные из порта. Если порт не поддерживает получение данных то данный метод вернет `null`. Метод наследуется от класса `CommPort`.
- `public abstract OutputStream getOutputStream() throws IOException`  
Возвращает поток `OutputStream`. Это один единственный способ послать данные в порт. Если порт не поддерживает посылку данных то данный метод вернет `null`. Метод наследуется от класса `CommPort`.
- `public void close()`  
Метод закрывает порт. Приложение может вызвать метод `close` когда оно закончило работу с портом. Метод наследуется от класса `CommPort`.
- `public abstract void notifyOnDataAvailable(boolean enable)`  
Когда данные будут доступны в входном буфере, будут послано оповещение `listener` который зарегистрирован методом `addEventListener`. Сообщение будет сгенерировано один раз, когда новые данные поступят в последовательный порт.

Параметры:

enable - true: оповещение включено.

false: оповещение выключено.

- public abstract void  
addEventListener(SerialPortEventListener lsnr) throws  
TooManyListenersException

Регистрирует объект SerialPortEventListener для прослушивания  
SerialEvent-s.

Итак, разберем по шагам, что происходит в конструкторе класса Terminal. Получив объект CommPortIdentifier, мы открываем порт, и преобразуем его к типу SerialPort:

```
serialPort = (SerialPort)  
portId.open("TerminalApp", 2000);
```

после чего открываем входной и выходной потоки:

```
outputStream = serialPort.getOutputStream();  
inputStream = serialPort.getInputStream();
```

регистраем объект SerialPortEventListener и устанавливаем параметры порта:

```
serialPort.addEventListener(this);  
serialPort.setSerialPortParams(9600,  
    SerialPort.DATABITS_8,  
    SerialPort.STOPBITS_1,  
    SerialPort.PARITY_NONE);
```

Ну, и наконец, создаем новый поток, который в цикле, с интервалом в пять секунд, начинает писать в порт сообщения из массива messageString = {"ATI3\n", "ATI1\n", "AT\n"};

```
outputStream.write(messageString[i].getBytes());
```

Сообщения поступающие из COM порта на поможет прослушать метод serialEvent, который при наступлении события SerialPortEvent.DATA\_AVAILABLE читает данные из входного потока.

## Доступ к COM порту из Java приложения

Вот казалось бы и все, только осталась маленькая деталь. Заменяем строку в конструкторе класса Terminal:

```
outputStream = serialPort.getOutputStream();
```

на

```
outputStream = new ConvertedOutputStream  
                (serialPort.getOutputStream());
```

где класс `ConvertedOutputStream` выглядит следующим образом.

```
import java.io.OutputStream;  
import java.io.IOException;  
  
class ConvertedOutputStream extends OutputStream {  
    OutputStream outstream;  
    ConvertedOutputStream(OutputStream outstream) {  
        this.outstream = outstream;  
    }  
    public void flush() throws IOException {  
        outstream.flush();  
    }  
    private int prev = 0;  
    public void write(int b) throws IOException {  
        if (b == '\n') {  
            if (prev != '\r')  
                outstream.write('\r');  
        } else if (b == '\r') {  
            if (prev != '\n')  
                outstream.write('\n');  
        }  
        prev = b;  
        outstream.write(b);  
    }  
}
```

Данный класс расширяет `OutputStream`, перегружая основной метод `write(int b)` и его понимание я оставляю на откуп читателям.

Таким образом, наше приложение вполне сформировано, если его запустить на выполнение, то мы увидим примерно следующее:

```
ATI3
Sportster 33600/Fax V10.0.23
OK
ATI1
7C60
OK
AT
OK
```

Модем корректно откликнулся на команды, следовательно, написанное нами приложение, работает правильно. Так что теперь, вооружившись полученными знаниями вы можете спокойно приступать к написанию аналога HyperTerminal :)). За дополнительными сведениями обращайтесь к литературе (см [1]).

## 4. Ресурсы

- <http://java.sun.com/products/javacomm/javadocs/packages.html> Это полное описание API функций пакета javax.com
- <http://java.sun.com/products/javacomm/index.html> Здесь находится пакет javax.comm и инструкции по его установке.

[ООО 'Кубик' \(www.qbix.ru\), Санкт-Петербург.](http://www.qbix.ru)