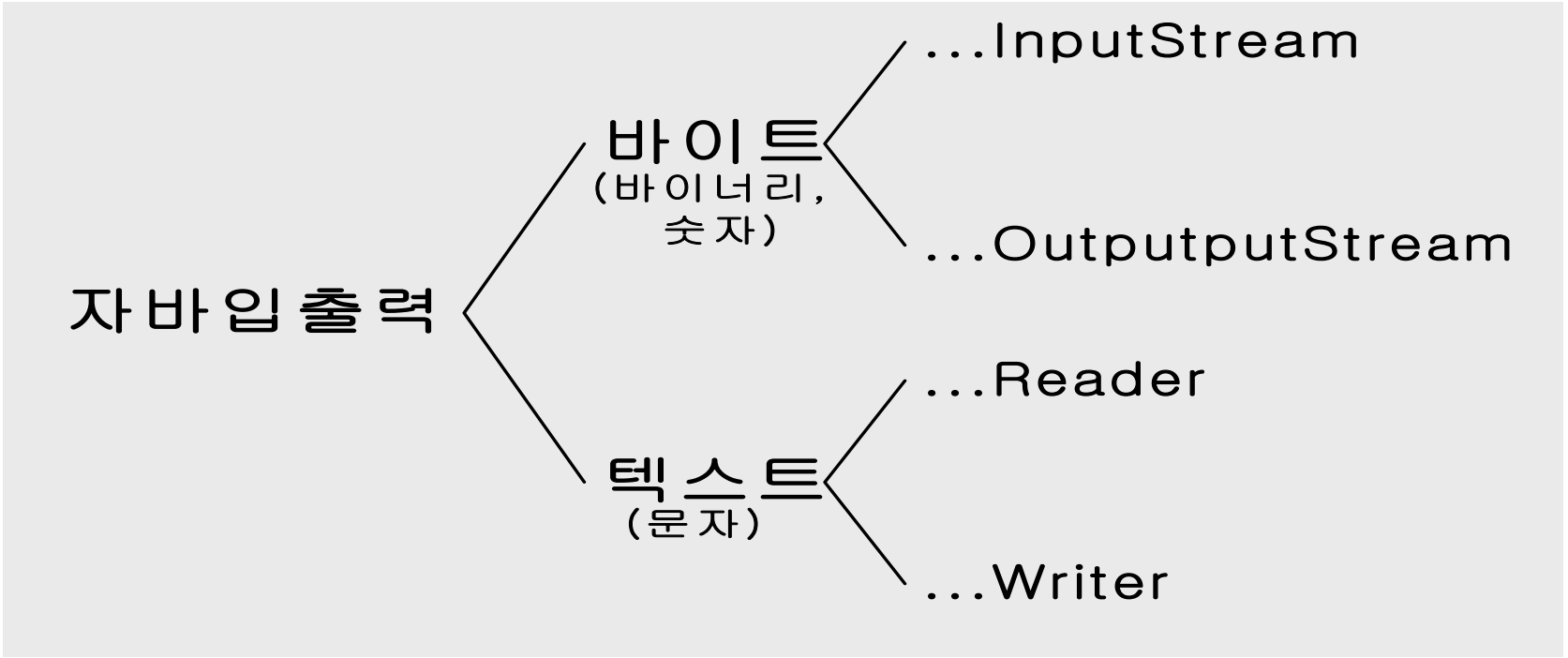
프로그래머를 위한 Java 2, 4판

제11장 자바 입출력

11.1.1 스트림 개념

- 스트림
  - 순서가 있고, 길이가 정해져 있지 않은 데이터 흐름
  - 종류



11.1.1 스트림 개념

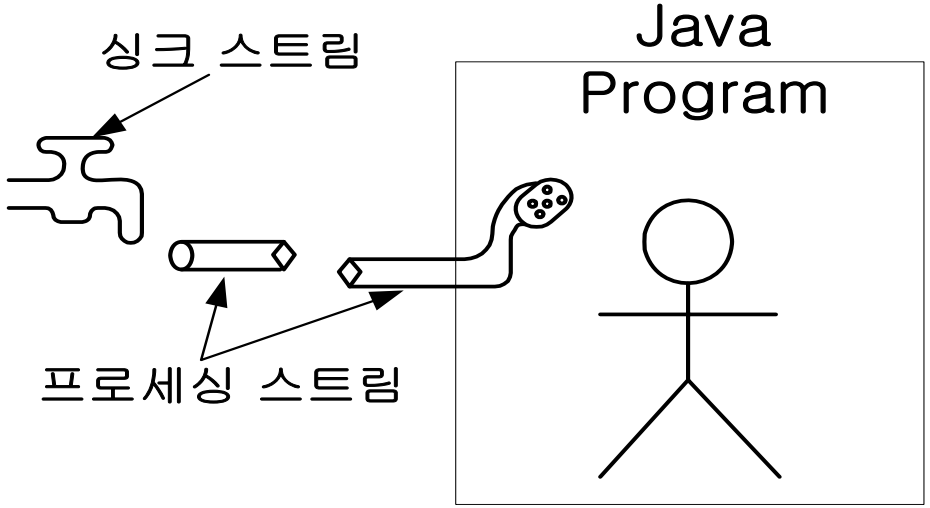
- 바이트 스트림
  - 바이너리(binary) 데이터와 숫자를 읽고, 쓰기 위해서 사용.
    - 예: 그림 파일, 동영상 파일, 응용프로그램 파일 등
  - 바이트 단위로 입출력 작업

11.1.1 스트림 개념

- 문자 스트림
  - 문자 스트림은 문자 데이터를 읽고, 쓰기 위해서 사용.
  - 인코딩을 자동으로 변환해줌

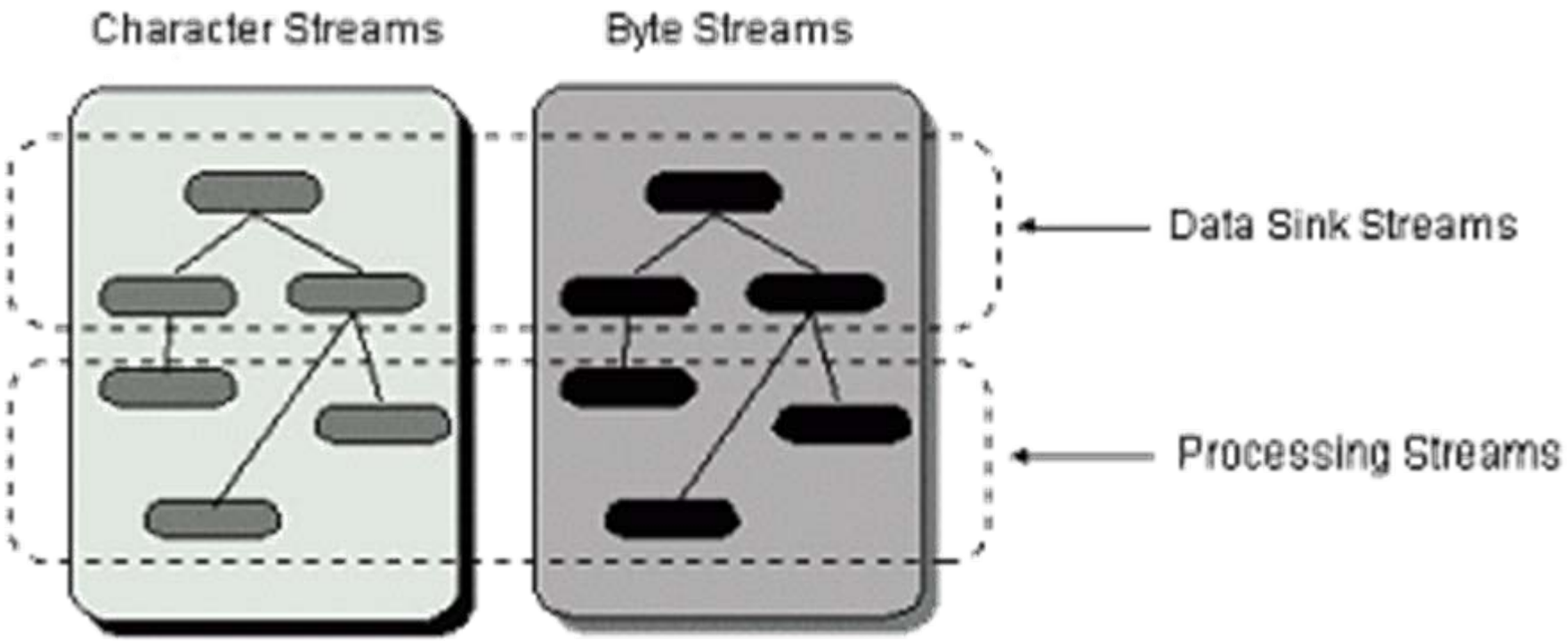
11.1.2 자바 입출력 클래스 분류와 상속 관계

- 용도에 따른 분류
  - 데이터 싱크 스트림
    - 데이터를 근원지에서 직접 읽거나, 목적지에 기록하는 클래스들
  - 데이터 프로세싱 스트림
    - 다른 스트림을 이용해서 중간에서 어떤 작업을 수행하는 클래스들
- 샤워의 비유
  - 수도꼭지 -> 싱크 스트림
  - 샤워기, 수도꼭지와 샤워기를 연결하는 것 -> 프로세싱 스트림



11.1.2 자바 입출력 클래스 분류와 상속 관계

- 클래스들 분류



11.1.2 자바 입출력 클래스 분류와 상속 관계

- 싱크스트림 클래스들

근원지/목적지	바이트 스트림	문자 스트림
메모리	ByteArrayInputStream, ByteArrayOutputStream	CharArrayReader, CharArrayWriter
	StringBufferInputStream	StringReader, StringWriter
파이프	PipedInputStream, PipedOutputStream	PipedReader, PipedWriter
파일	FileInputStream, FileOutputStream	FileReader, FileWriter

Console System.in/out

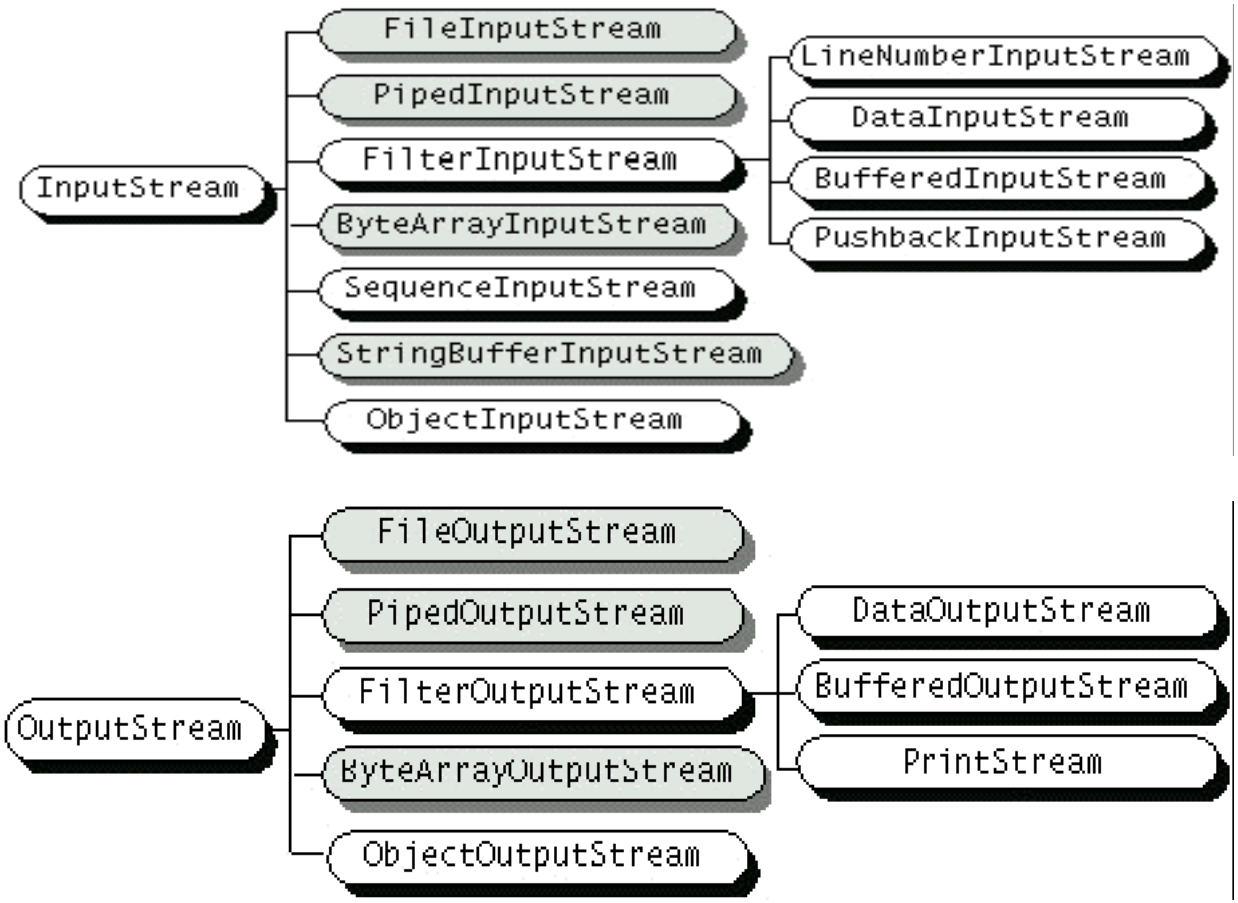
11.1.2 자바 입출력 클래스 분류와 상속 관계

- 데이터 프로세싱 스트림 클래스
  - 데이터 프로세싱 스트림은 항상 다른 스트림을 이용해서만 생성가능
  - 데이터 프로세싱 스트림의 생성자는 다른 스트림을 매개 변수로 받게됨

처 리	바이트 스트림	문자 스트림
버퍼링	BufferedInputStream, BufferedOutputStream	BufferedReader, BufferedWriter
필터링	FilterInputStream, FilterOutputStream	FilterReader, FilterWriter
바이트 스트림 -> 문자 스트림		InputStreamReader, OutputStreamWriter
여러 개의 스트림을 결합	SequenceInputStream	
객체 직렬화	ObjectInputStream, ObjectOutputStream	
자료 변환	DataInputStream, DataOutputStream	
프린트	PrintStream	PrintWriter

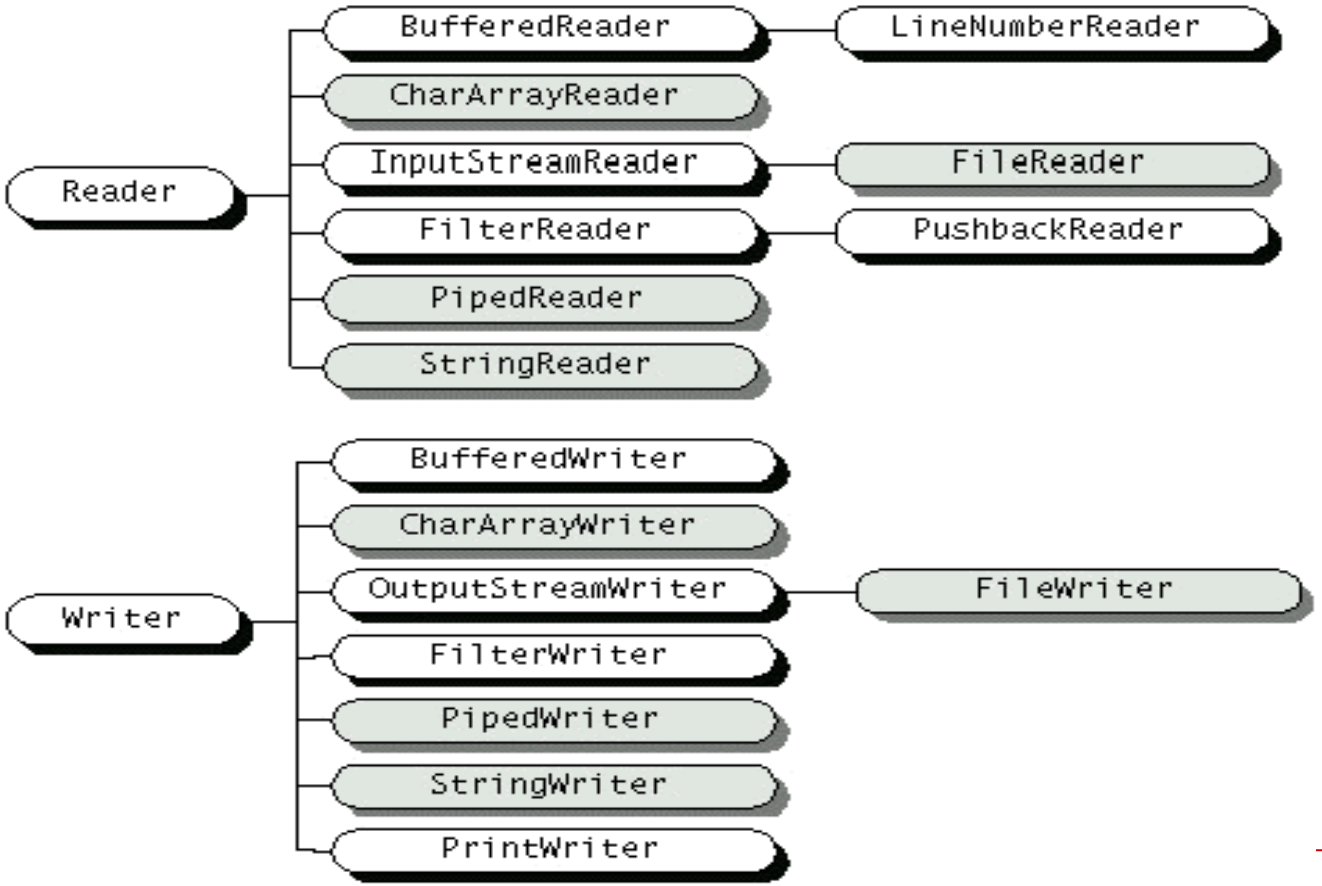
11.1.2 자바 입출력 클래스 분류와 상속 관계

- 바이트 스트림 상속 관계
  - InputStream 과 OutputStream으로 부터 상속받음.
    - ✓ 회색은 싱크 스트림 클래스



11.1.2 자바 입출력 클래스 분류와 상속 관계

- 문자 스트림 상속 관계
  - Reader와 Writer 클래스로부터 상속받는다.
    - ✓ 회색은 싱크 스트림 클래스



11.2.1 InputStream/OutputStream 추상 클래스

- InputStream 클래스
  - 바이트(byte) 입력 스트림을 위한 가장 상위 클래스.
  - System.in
  - 대표적인 메소드들
    - int read() - 입력 스트림에서 한 바이트를 읽어서 리턴.
    - int read(byte[] b, int off, int len) - 입력 스트림에서 len 개의 바이트를 읽어서 바이트 배열 b에 저장.
    - int read(byte[] b) - 배열의 크기만큼 읽는다. 읽은 바이트 수 리턴.

11.2.1 InputStream/OutputStream 추상 클래스

- 예제: SimpleRead.java

```
 8          try {
 9              b = System.in.read();
10              while(b != -1) {
11                  System.out.print((char)b);
12                  count++;
13                  b = System.in.read();
14              }
15          } catch(IOException e) {
16              System.out.println(e);
17          }
18          System.out.println();
19          System.out.println("total byte:" + count);
```

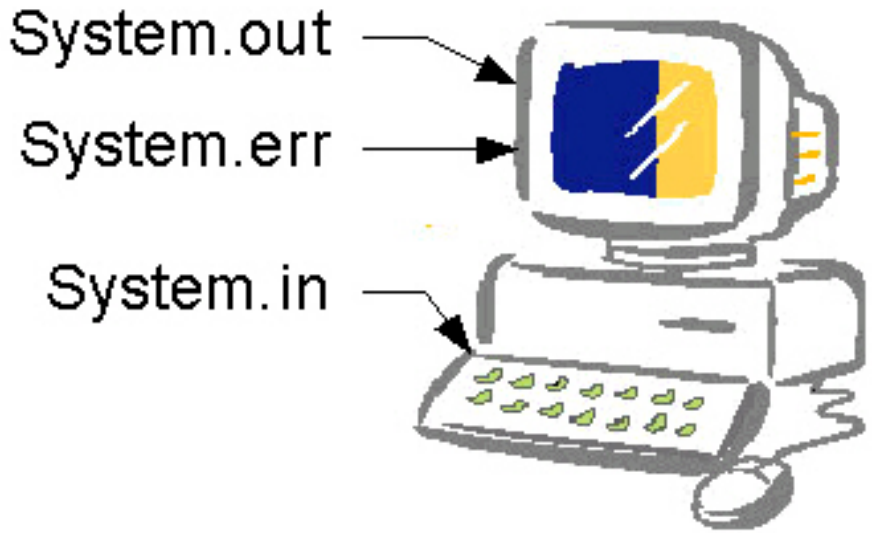
- 결과

```
C:\> java SimpleRead
Hello
Hello
한글
??±?
^Z

total bytes:13
```

11.2.1 InputStream/OutputStream 추상 클래스

- 콘솔



- 리다이렉션

```
C:₩> java SimpleRead  < SimpleRead.java
import java.io.IOException;
...
```

```
C:₩> java SimpleRead < SimpleRead.java > output.txt
C:₩> more output.txt
import java.io.IOException;
```

11.2.1 InputStream/OutputStream 추상 클래스

- Scanner 클래스
  - C 언어의 scanf() 함수와 비슷한 형태의 입력을 지원함
- 예제: ScannerTest.java

```
 7        Scanner sc = Scanner.create(System.in);
 8        while(sc.hasNextInt()) {
 9            int i = sc.nextInt();
10            System.out.println(i);
11        }
12
13        System.out.println("----------
14        while(sc.hasNext()) {
15            String s = sc.next();
```

●결과

```
C:\> java ScannerTest
->1 2 a 4 b
1
2
----------
a
4
b
^Z
```

11.2.1 InputStream/OutputStream 추상 클래스

- ConsoleReader 클래스
  - 콘솔에서 데이터를 쉽게 읽기 위한 사용자 정의 클래스
- 예제: ConsoleReader.java

```
3  public class ConsoleReader extends BufferedReader {
4      public ConsoleReader(Reader in) throws IOException {
5          super(in);
6      }
7
8      public ConsoleReader(InputStream in) throws IOException {
9          super(new InputStreamReader(in));
10     }
11
12     public int readInt() throws IOException {
13         return Integer.parseInt(readLine().trim());
```

11.2.1 InputStream/OutputStream 추상 클래스

- 예제: ConsoleReader.java(계속)

```
16    public String readString() throws IOException {
17        return readLine().trim();
18    }
19
20    public char readChar() throws IOException {
21        String line = readLine().trim();
22        return line.charAt(0);
23    }
24
25    public double readDouble() throws IOException {
26        Double d = new Double(readLine().trim());
27        return d.doubleValue();
```

11.2.1 InputStream/OutputStream 추상 클래스

- 예제: ConsoleReaderTest.java

```
11          ConsoleReader in = new ConsoleReader(System.in);
12          System.out.print("정수를 입력해주세요:");
13          ivalue = in.readInt();
14          System.out.println("value = " + ivalue);
15          System.out.print("₩n실수를 입력해주세요:");
16          dvalue = in.readDouble();
17          System.out.println("value = " + dvalue);
18          System.out.print("₩n문자를 입력해주세요:");
19          cvalue = in.readChar();
20          System.out.println("value = " + cvalue);
21          System.out.print("₩n문자열을 입력해주세요:");
22          svalue = in.readString();
23          System.out.println("value = " + svalue);
```

11.2.1 InputStream/OutputStream 추상 클래스

- Copier 클래스
  - 입력 스트림에서 읽은 데이터를 출력 스트림으로 출력하는 사용자 정의 클래스
- 예제: Copier.java

```
3  public class Copier {
4      protected InputStream in;
5      protected OutputStream out;
6      protected byte data[];
7
8      public Copier(InputStream in, OutputStream out, int sz) {
9          this.in = in;
10         this.out = out;
11         data = new byte[sz];
```

11.2.1 InputStream/OutputStream 추상 클래스

- 예제: Copier.java(계속)

```
14      public Copier(InputStream in, OutputStream out) {
15          this(in, out, 1024);
16      }
17
18      public void copy() throws IOException {
19          int n = 0;
20          while((n = in.read(data)) != -1) {
21              out.write(data, 0, n);
22          }
23          in.close();
24          out.close();
```

11.2.2 FileInputStream/FileOutputStream

- FileInputStream
  - 파일에서 바이트 단위로 내용을 읽기 위해 사용
  - 생성자
    - FileInputStream(File file)
    - FileInputStream(String name)

- FileOutputStream
  - 파일로 바이트 단위로 내용을 출력하기 위해 사용
  - 생성자
    - FileOutputStream(File file)
    - FileOutputStream(String name, boolean append)
    - FileOutputStream(String name)

11.2.2 FileInputStream/FileOutputStream

- 예제: ConsoleCopy.java

```
4     public static void main(String args[
5         if(args.length != 1) {
6             System.out.println(
7                 "usage: java ConsoleCopy
8             System.exit(1);
9         }
10        try {
11            Copier cp = new Copier(System.in,
12                new FileOutputStream(args[0]));
13            cp.copy();
```

●결과

```
C:\> java ConsoleCopy abc.txt
한글입니다.
^Z

C:\> more abc.txt
한글입니다.
```

11.2.2 FileInputStream/FileOutputStream

- 예제: More.java

```
4    public static void main(String args[]) {
5        if(args.length != 1) {
6            System.out.println(
7                "usage: java More <filename>");
8            System.exit(1);
9        }
10       try {
11           Copier cp = new Copier(new FileInputStream(args[0]),
12                   System.out);
13           cp.copy();
```

●결과

```
C:\> java More More.java
import java.io.*;

public class More {
.....
```

11.2.2 FileInputStream/FileOutputStream

- 예제: FileCopy.java

```
5        if(args.length != 2) {
6            System.out.print("usage: jav
7            System.out.println("<filenam
8            System.exit(1);
9        }
10       try {
11           Copier cp = new Copier(new FileInputStream(args[0]),
12                                  new FileOutputStream(args[1]));
13           cp.copy();
14           System.out.println("파일 복사가 완료되었습니다.");
```

●결과

```
C:\> java FileCopy abc.txt  bc.txt
파일 복사가 완료되었습니다.

C:\> more bc.txt
한글입니다.
```

11.2.3 FileReader/FileWriter

- FileReader
  - 파일에서 문자 단위로 내용을 읽을 때 사용
  - 생성자
    - FileReader(File file)
    - FileReader(String fileName)
- FileWriter
  - 파일에 문자 데이터를 출력할 때 사용
  - 생성자
    - FileWriter(File file)
    - FileWriter(String fileName, boolean append)
    - FileWriter(String fileName)

11.2.3 FileReader/FileWriter

- 예제: ReadFileData.java

```
10          try {
11              ConsoleReader in = new ConsoleReader(
12                  new FileReader("data.txt"));
13              ivalue = in.readInt();
14              System.out.println("value = " + ivalue);
15              dvalue = in.readDouble();
16              System.out.println("value = " + dvalue);
17              cvalue = in.readChar();
18              System.out.println("value = " + cvalue);
19              svalue = in.readString();
20              System.out.println("value = " + svalue);
```

11.2.4 InputStreamReader/OutputStreamWriter

- InputStreamReader
  - 입력 스트림을 Reader 타입으로 변환
  - 생성자
    - InputStreamReader(InputStream in, String enc)
    - InputStreamReader(InputStream in)

- OutputStreamWriter
  - 출력 스트림을 Writer 클래스로 변환
  - 생성자
    - OutputStreamWriter(OutputStream out, String enc)
    - OutputStreamWriter(OutputStream out)

11.2.5 PrintStream/PrintWriter

- PrintStream
  - 데이터 출력(예: System.out, System.err)
  - 생성자
    - PrintStream(OutputStream out, boolean autoFlush)
    - PrintStream(OutputStream out)

- PrintWriter
  - 문자 데이터 출력
  - 생성자
    - PrintWriter(OutputStream out, boolean autoFlush)
    - PrintWriter(OutputStream out)
    - PrintWriter(Writer out, boolean autoFlush)
    - PrintWriter(Writer out)

11.2.5 PrintStream/PrintWriter

- ConsoleWriter
  - 출력을 보다 편리하게 할 수 있는 사용자 정의 클래스
- 예제: ConsoleWriter.java

```
 4  public class ConsoleWriter extends PrintWriter {
 5      public ConsoleWriter(OutputStream out) {
 6          super(out, true);
...
 9      public ConsoleWriter(Writer out) {
10          super(out, true);
...
13      public void printWon(long value) {
14          String format = "W#,###";
15          DecimalFormat df = new DecimalFormat(format);
16          print(df.format(value));
```

11.2.5 PrintStream/PrintWriter

- 예제: ConsoleWriter.java(계속)

```
24      public void printDollar(float value) {
25          String format = "$#,###.00";
26          DecimalFormat df = new DecimalFormat(format);
27          print(df.format(value));
...
35      public void print(double value, int digit) {
36          String format = "#.";
37          for(int i=0; i < digit; i++) {
38              format += "#";
39          }
40          DecimalFormat df = new DecimalFormat(format);
41          print(df.format(value));
42          flush();
```

11.2.5 PrintStream/PrintWriter

- 예제: ConsoleWriterTest.java

```
5   ConsoleWriter out = new ConsoleWriter(System.out);
6   double d = 123.456789;
7   int money = 1234567;
8
9   out.println(d);
10  out.println(d, 2);
11  out.printlnWon(money);
12  out.printlnDollar(money);
```

●결과

```
C:\> java ConsoleWriterTest
123.456789
123.46
W1,234,567
$1,234,567.00
```

11.2.6 BufferedInputStream/BufferedOutputStream

- BufferedInputStream
  - 바이트 입력에서 성능을 높이기 위해서 버퍼를 사용하는 클래스
  - 생성자
    - BufferedInputStream(InputStream in, int size)
    - BufferedInputStream(InputStream in)

- BufferedOutputStream
  - 바이트 출력에서 성능을 높이기 위해서 버퍼를 사용하는 클래스
  - 생성자
    - BufferedOutputStream(OutputStream out, int size)
    - BufferedOutputStream(OutputStream out)

11.2.7 BufferedReader/BufferedWriter

- BufferedReader
  - 버퍼를 사용하는 Reader
  - 생성자 및 중요 메소드
    - BufferedReader(Reader in, int sz)
    - BufferedReader(Reader in)
    - String readLine()

- BufferedWriter
  - 버퍼를 사용하는 Writer
  - 생성자
    - BufferedWriter(Writer out, int sz)
    - BufferedWriter(Writer out)

11.2.7 BufferedReader/BufferedWriter

- 예제: ConfigurationReader.java

```
28          BufferedReader br =
29              new BufferedReader(new FileReader(file) );
30
31          while((rl=br.readLine()) != null) {
32              rl = rl.trim();
33              if(rl.charAt(0) == comment )    {
34                  continue;
35              } else {
36                  StringTokenizer st =
37                      new StringTokenizer(rl,delm,false);
38                  String key = st.nextToken();
39                  String value = st.nextToken();
40                  ht.put(key, value);
```

11.2.7 BufferedReader/BufferedWriter

- 예제: ConfigurationReader.java(계속)

```
46      public String getValue(String name) {
47          return (String)ht.get(name);
48      }
49
50      public static void main(String args[])
51          ConfigurationReader cr =
52          new ConfigurationReader("server.conf");
53          try {
54              cr.parse();
55              System.out.println(cr.getValue("port"));
56              System.out.println(cr.getValue("server"));
57              System.out.println(cr.getValue("admin"));
```

- 결과

```
C:\> java ConfigurationReader
7690
it.mokpo.ac.kr
admin@it.mokpo.ac.kr
```

11.2.8 DataInputStream/DataOutputStream

- DataInputStream
  - 데이터 타입별로 읽을 때 사용
  - 생성자 및 중요 메소드
    - DataInputStream(InputStream in)
    - char readChar()
    - double readDouble()
    - int readInt()
- DataOutputStream
  - 데이터 타입별로 출력
  - 생성자 및 중요 메소드
    - DataOutputStream(OutputStream out)
    - void writeChar(int v)
    - void writeInt(int v)

11.2.8 DataInputStream/DataOutputStream

- 예제: DataIOTest.java

```
12      public void write() throws IOException {
13          DataOutputStream out = new
14              DataOutputStream(new FileOutputStream(file));
15          int value[] = Fibonachi.calculate(n);
16          for(int i=0; i <= n; i++) {
17              out.writeInt(value[i]);
18              //System.out.println(value[i]);
19          }
20          out.flush();
21          out.close();
```

11.2.8 DataInputStream/DataOutputStream

- 예제: DataIOTest.java(계속)

```
24      public void read() throws IOException {
25          DataInputStream in =
26              new DataInputStream(new FileInputStream(file));
27          int n = in.available()/4;
28          //System.out.println("# of data = " + n);
29          for(int i=0; i < n; i++) {
30              int v = in.readInt();
31              System.out.println("f["+i+"] = " + v);
32          }
33          in.close();
34      }
```

11.2.9 ObjectInputStream/ObjectOutputStream

- ObjectInputStream
  - 저장된 객체를 읽는다.
  - 생성자 및 중요 메소드
    - ObjectInputStream(InputStream in)
    - Object readObject()

- ObjectOutputStream
  - 객체를 저장한다.
  - 생성자 및 중요 메소드
    - ObjectOutputStream(OutputStream out)
    - void writeObject(Object obj)

11.2.9 ObjectInputStream/ObjectOutputStream

- 예제: SerializedDate.java

```
15    public void  write(String file) {
16        try {
17            FileOutputStream  fout = new FileOutputStream(file);
18            os = new ObjectOutputStream(fout);
19            System.out.println("We save this date.. :"+ date);
20            os.writeObject(date);
21            os.close();
22        } catch(IOException e) {
23            e.printStackTrace();
24        }
25    }
```

11.2.9 ObjectInputStream/ObjectOutputStream

- 예제: SerializedDate.java(계속)

```
27      public void read(String file) {
28          try {
29              FileInputStream  fin = new FileInputStream(file);
30              is = new ObjectInputStream(fin);
31              date = (Date)is.readObject();
32              System.out.println("We gets saved old date.. :"+ date);
33              System.out.println("Current date is .. :"+ new Date());
34          } catch(IOException e) {
35              e.printStackTrace();
36          } catch(ClassNotFoundException nfe) {
37              nfe.printStackTrace();
38          }
```

11.2.10 StringReader/StringWriter

- StringReader
  - 문자열에서 다른 Reader와 동일한 방법으로 데이터를 읽은 수 있는 방법을 제공
  - 생성자
    - StringReader(String s)

- StringWriter
  - 문자열에 다른 Writer와 동일한 방법으로 데이터를 출력하는 방법 제공
  - 생성자
    - StringWriter()
    - StringWriter(int initialSize)

11.2.11 RandomAccessFile

- RandomAccessFile
  - 입출력 포인터의 위치를 자유롭게 이동
  - 생성자 및 중요 메소드
    - RandomAccessFile(String filename, String openmode)
    - void seek(long pos)
    - long length()
    - boolean readBoolean()
    - char readChar()
    - double readDouble()
    - float readFloat()
    - int readInt()
    - long readLong()

11.2.11 RandomAccessFile

- Record
  - 직원 정보를 표현하기 위한 사용자 정의 클래스
- 예제: Record.java

```
1  public class Record {
2      private  String  name;
3      private  int     num;
4      private  String  part;
5      private  String  tel;
6
7      public Record(String na, int n, String p, String t) {
8          name  = na;
9          num   = n;
10         part  = p;
11         tel   = t;
```

11.2.11 RandomAccessFile

- DataTable
  - Record 클래스를 RandomAccessFile을 이용해서 파일에 저장 및 읽기 위한 사용자 정의 클래스
  - Record 객체의 크기 : 42 바이트
  - RandomAccessFile의 첫 4 바이트는 Record 객체의 개수를 의미
- 예제: DataTable.java

```
 9      public DataTable(String file) {
10          try {
11              dbfile = new RandomAccessFile(file, "rw");
12              if(dbfile.length() ==0) {
13                  rc = 0;
14              } else {
15                  rc = dbfile.readInt();
```

11.2.11 RandomAccessFile

- 예제: DataTable.java(계속)

```
   39     public Record readRec(int index) {
...
   44           dbfile.seek(index * recordLength + recordOffset);
   45           for(int i = 0; i < 5; i++)  {
   46               name += dbfile.readChar();
...
   49           num = dbfile.readInt();
   50
   51           for(int i = 0; i < 6;i++) {
   52               part += dbfile.readChar();
...
   61        return (new Record(name, num, part, tel));
```

11.2.11 RandomAccessFile

- 예제: DataTable.java(계속)

```
   88     public void writeRec(String name, int num, String part,
   89             String tel, int index) {
...
   93             dbfile.seek(index * recordLength + recordOffset);
   94             len = name.length();
   95             for(int i =0; i < 5; i++) {
   96                 dbfile.writeChar((i < len ? name.charAt(i):' '));
...
   99             dbfile.writeInt(num);
...
  111              dbfile.seek(0);
  112              dbfile.writeInt(rc);
```

11.2.11 RandomAccessFile

- CMD
  - 작업 명령어를 기술하기 위한 사용자 정의 열거형
- 예제: CMD.java

```
1  public enum CMD {
2      NONE, ADD, TOTAL, SEARCH, REMOVE, ESC;
3  }
```

11.2.11 RandomAccessFile

- 예제: EmployeeDB.java

```
 18          action = CMD.NONE;
...
 27          table = new DataTable(dbFile);
...
121      private void add() {
...
135          nameString = name.getText().trim();
136          num  = number.getText().trim();
137          part = dept.getText().trim();
138          tel = phone.getText().trim();
139
140          display.setText(nameString + num + part + tel);
141          table.add(nameString, Integer.parseInt(num), part, tel);
142          display.append("₩n 등록을 성공적으로 마쳤습니다.");
```

11.2.11 RandomAccessFile

- 예제: EmployeeDB.java(계속)

```
154     private void total() {
155         action = CMD.TOTAL;
156         int len = table.length();
157         display.setText("₩n 검색 건수 : 총 " + len + " 건₩n");
158         display.append("₩n================================");
159         display.append("₩n 순번  ₩t성명   ₩t사번   ₩t부서   ₩t전화");
160         display.append("₩n================================₩n");
161         if(len == 0) {
162             display.append(" 등록된 자료가 없습니다. ₩n");
163             return;
164         }
165         for(int i = 0; i < len; i++) {
166             Record rec = table.readRec(i);
167             print(i, rec.getName(), rec.getNum(),
168                     rec.getPart(), rec.getTel());
```

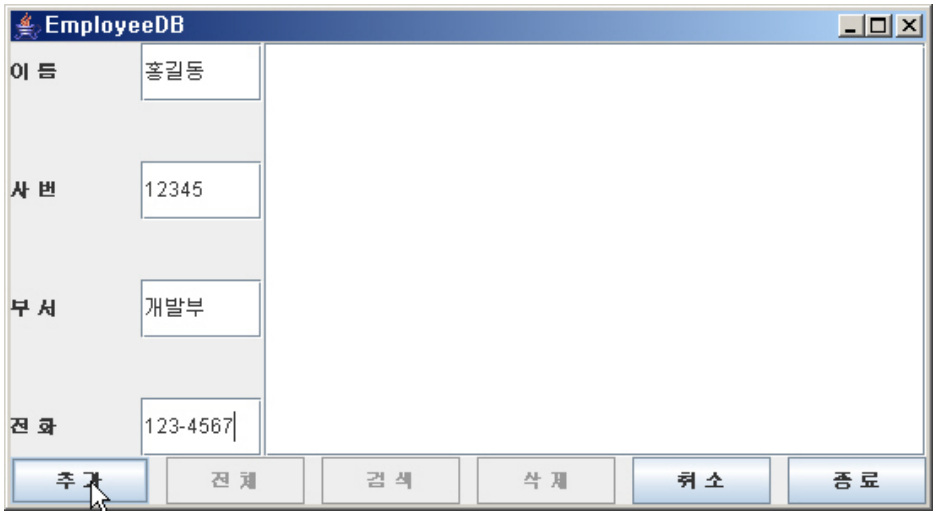
11.2.11 RandomAccessFile

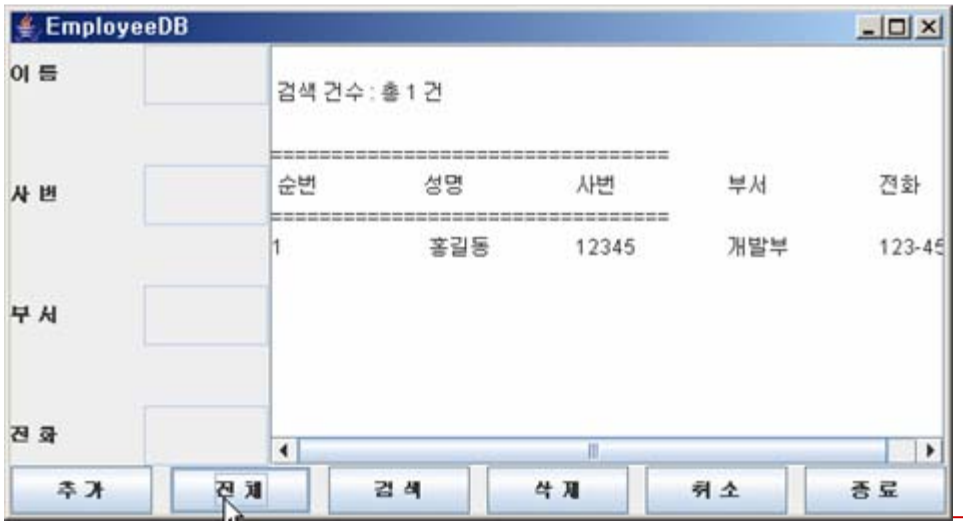
- 예제: EmployeeDB.java(계속)

```
  181     private void search() {
...
  193         int len = table.length();
...
  202         for(int i = 0; i < len ; i++) {
  203             rec = table.readRec(i);
  204             if(nameString.equals(table.readName(i,
  205                     nameString.length()))) {
  206                 found = true;
  207                 print(i, rec.getName(), rec.getNum(),
  208                         rec.getPart(), rec.getTel());
  209             }
  210         }
  211         if (!found) {
  212             display.setText(" 조회하신 직원이 없습니다.");
```

11.2.11 RandomAccessFile

- EmployeeDB 결과





11.2.12 StreamTokenizer

- StreamTokenizer
  - 입력 스트림의 내용을 토큰(token)으로 잘라서 리턴하는 클래스
  - 입력 스트림의 내용을 공백 문자, 알파벳, 숫자, 문자열, 주석이라는 5개 종류의 토큰으로 파싱할 수 있다.
  - 생성자, 멤버 필드, 메소드
    - StreamTokenizer(Reader r)
    - double nval - 현재 토큰이 숫자인 경우에 현재 토큰의 숫자 값
    - String sval - 현재 토큰이 단어인 경우에 현재 토큰의 문자열
    - static int TT_EOF - 파일의 끝
    - static int TT_EOL - 현재 라인의 끝
    - static int TT_NUMBER - 읽은 토큰이 숫자임
    - static int TT_WORD - 읽은 토큰이 단어임
    - int ttype - nextToken()

11.2.12 StreamTokenizer

- StreamTokenizer 클래스의 상태 변화

상태	입력	액션	새로운 상태
Idle	단어 문자	입력 스트림에 돌려준다.	Accumulate
	ordinary 문자	문자를 리턴한다	Idle
	공백 문자	문자를 소비한다	Idle
accumulate	단어 문자	현재 단어에 추가한다.	Accumulate
	ordinary 문자	현재 단어를 리턴하고, 현재 문자는 입력 스트림에 돌려준다.	Idle
	공백 문자	현재 단어를 리턴하고, 현재 문자는 입력 스트림에 돌려준다.	Idle

11.2.13 파일(File)

- File
  - 파일과 디렉토리를 표현하기 위해서 사용
  - File 클래스가 할 수 있는 일
    - 디렉토리 내용을 알아본다.
    - 파일 속성을 알아보거나 설정한다.
    - 파일의 이름을 변경하거나 삭제한다.
  - 생성자 및 메소드
    - File(File parent, String child)
    - File(String pathname)
    - String getName()
    - long length()
    - File[] listFiles()

11.2.13 파일(File)

- FilenameFilter
  - 파일 이름을 필터링해서 원하는 형태의 파일 이름을 갖는 파일들만 선택하도록 지원한다.
  - 메소드
    - boolean accept(File dir, String name)

11.2.13 파일(File)

- 예제: ExtensionFilter.java

```
3  public class ExtensionFilter implements FilenameFilter {
4      protected String ext;
5
6      public ExtensionFilter(String ext) {
7          this.ext = ext;
8      }
9
10     public boolean accept(File dir, String name) {
11         if(name.endsWith(ext)) {
12             return true;
13         } else {
14             return false;
```

11.2.13 파일(File)

- 예제: Dir.java

```
13          File files[] = null;
14          if(filter != null) {
15              files = d.listFiles(filter);
16          } else {
17              files = d.listFiles();
18          }
19          if(files == null) {
20              System.err.println(d.getName() + " 디렉터리가 없습니다.");
21              return;
22          }
23          for(int i=0; i < files.length; i++) {
24              System.out.print(files[i].getName());
25              System.out.print(" ₩t₩t");
26              System.out.println(files[i].length());
```

11.3 정규식(regular expression)

- 정규식 표기법

문자	의미
.	임의의 한 문자와 매치된다. (라인 끝과는 플래그 값에 따라 매치될 수도 있고, 매치되지 않을 수도 있다.)
[abc]	[] 안에 포함된 한 문자와 매치된다. 즉, a 혹은 b 혹은 c
-	[]안에서 사용되며, 범위를 나타낸다. [a-c]은 [abc]와 동일하다.
[^abc]	[]안에서 사용되는 ^은 []안에 포함된 내용은 제외한다는 의미이다.
&&	AND 혹은 교집합을 의미한다.
^	[] 밖에서 사용되는 ^은 줄의 처음을 의미한다.
$	$는 줄의 끝을 의미한다.
X?	?는 앞의 내용(예: X)이 한번 혹은 0번 나타난다는 의미이다.
X*	*는 앞의 내용(예: X)이 0번 이상 나타난다는 의미이다.
X+	+ 는 앞의 내용(예: X)이 1번 이상 나타난다는 의미이다.
X{n}	{n}은 앞의 내용이 n번 나타난다는 의미이다.
X{n,+}	X가 n번 이상 나타난다는 의미이다.
X{n,m}	X가 n번 이상, m번 이하로 나타난다는 것을 의미한다.
X | Y	X 혹은 Y와 매치된다는 의미이다.

11.3 정규식(regular expression)

- Pattern
  - 정규식을 표현하기 위해서 사용되는 클래스
  - 메소드들
    - static Pattern compile(String regex)
    - static Pattern compile(String regex, int flags)
    - Matcher matcher(CharSequence input)
  - 플래그들
    - static int CASE_INSENSITIVE
    - static int COMMENTS
    - static int DOTALL
    - static int MULTILINE
    - static int UNICODE_CASE
    - static int UNIX_LINES

11.3 정규식(regular expression)

- 예제: DataSpliter.java

```
 6      public DataSpliter(String reg) {
 7          pat = Pattern.compile(reg);
 8      }
 9
10      public String[] getValue(String data) {
11          return pat.split(data);
12      }
13
14      public static void main(String args[]) {
15          String data = "홍길동, 123-4567, hong@some.com";
16          DataSpliter ds = new DataSpliter("[,₩₩s]+");
17          String value[] = ds.getValue(data);
18          for(int i=0; i < value.length; i++) {
19              System.out.println(value[i]);
```

11.3 정규식(regular expression)

- Matcher 클래스
  - 문자들을 패턴에 매치시키는 역할을 함.
  - 메소드들
    - boolean matches()
    - boolean find()
    - Matcher appendReplacement(StringBuffer sb, String replacement)
    - StringBuffer appendTail(StringBuffer sb)

11.3 정규식(regular expression)

- 예제: ReplaceTest.java

```
5        String data = "홍길동, 123-4567, hong@some.com";
6        Pattern pat = Pattern.compile("[,₩₩s]+");
7        Matcher m = pat.matcher(data);
8        StringBuffer sb = new StringBuffer();
9        boolean result = m.find();
10       while(result == true) {
11           m.appendReplacement(sb, ":");
12           result = m.find();
13       }
14       m.appendTail(sb);
15       System.out.println(data + "  =>");
16       System.out.println(sb.toString());
```

11.4 새로운 I/O (NIO)

- NIO
  - 버퍼 관리와 네트워크 등에서 성능을 향상시키기 위한 기능들을 제공
- 클래스 상속 관계