13장 입출력처리

목차

- 입출력 스트림
- 바이트 스트림
- 문자 스트림
- 파일 관리

입출력 스트림

스트림 개념

- 연속된 데이터의 단방향 흐름을 추상화
- 데이터 소스와 상관없이 적용할 수 있어 매우 효과적
- 스트림의 예
  - 키보드 및 모니터의 입출력
  - 프로그램과 외부 장치, 파일의 입출력에서 데이터 흐름도 스트림
  - 네트워크와 통신하는 데이터의 흐름
  - 데이터 집합체의 각 원소를 순회하면서 람다식으로 반복 처리되는 데이터 흐름

스트림 입출력

- 버퍼를 가지고 순차적으로 이루어지는 입출력

입출력 스트림

자바의 입출력 스트림

- 응용프로그램과 입출력 장치를 연결하는 소프트웨어 모듈
  - 입력 스트림 : 입력 장치로부터 자바 프로그램으로 데이터를 전달
  - 출력 스트림 : 출력 장치로 데이터 출력

입출력 스트림의 특징

- 선입선출 구조라서 순차적으로 흘러가고 순차적으로 접근
- 스트림의 양끝에 입출력장치와 자바 응용프로그램 연결
- 임의 접근 파일 스트림을 제외한 모든 스트림은 단방향
  - 입력과 출력을 동시에 하는 스트림 없음
- 입출력 스트림은 객체
- 출력 스트림과 입력 스트림을 연결해서 파이프라인 구성 가능
- 지연 가능. 프로그램에 연결한 출력 스트림에 데이터가 가득 차면 프로그램을 더 이상 출력할 수 없어 빈 공간이 생길 때까지 지연되며, 데이터 소스에 연결한 입력 스트림도 가득 차면 프로그램이 데이터를 처리해서 빈 공간이 생길 때까지 스트림이 지연

입출력 스트림

바이트 스트림과 문자 스트림

입출력 스트림 기본 단위

- 바이트 스트림의 경우 : 바이트
- 문자 스트림의 경우 : 문자(자바에서는 문자1개 : 2 바이트)

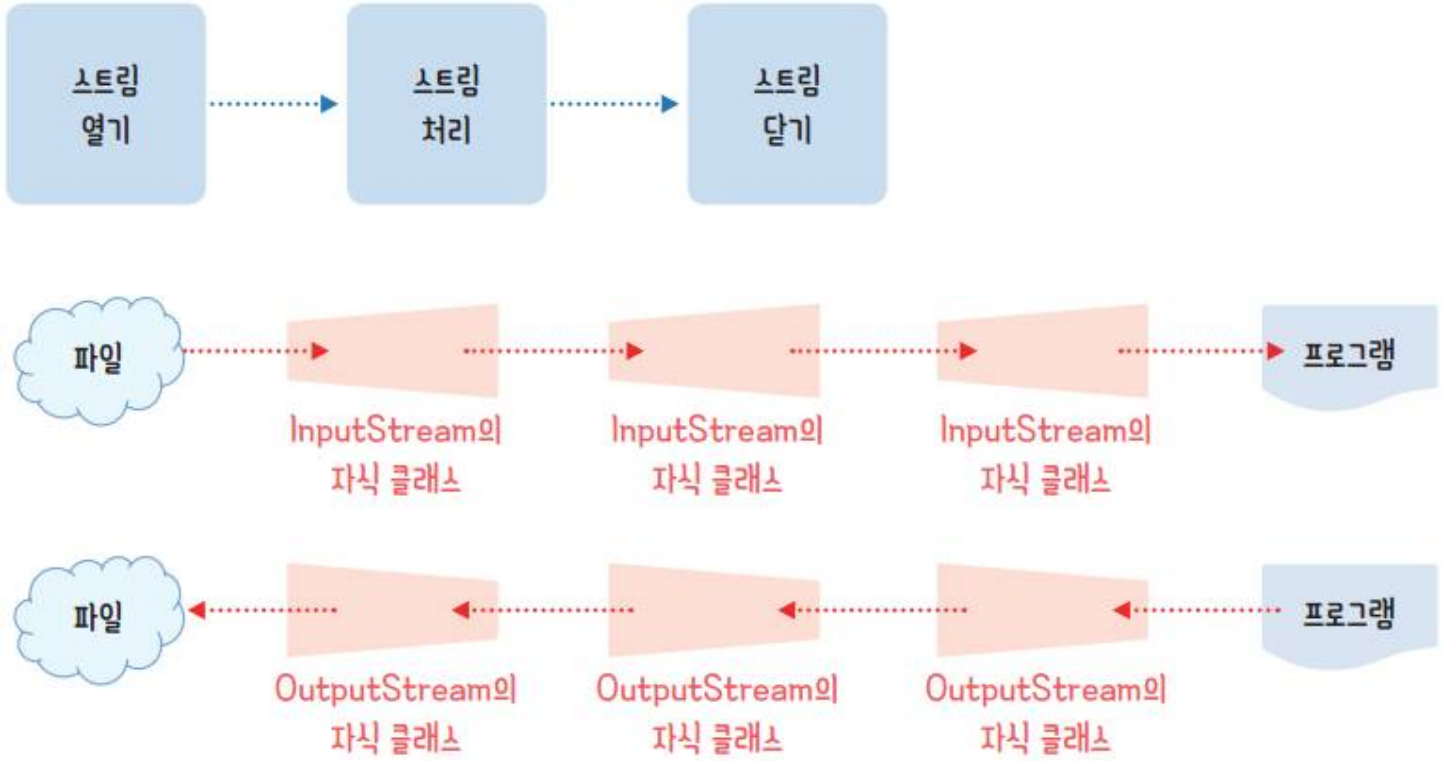
바이트 스트림과 문자 스트림

- 바이트 스트림
  - 입출력되는 데이터를 단순 바이트로 처리
    - 예) 바이너리 파일을 읽는 입력 스트림
- 문자 스트림
  - 문자만 입출력하는 스트림
  - 문자가 아닌 바이너리 데이터는 스트림에서 처리하지 못함
    - 예) 텍스트 파일을 읽는 입력 스트림

JDK는 입출력 스트림을 구현한 다양한 클래스 제공

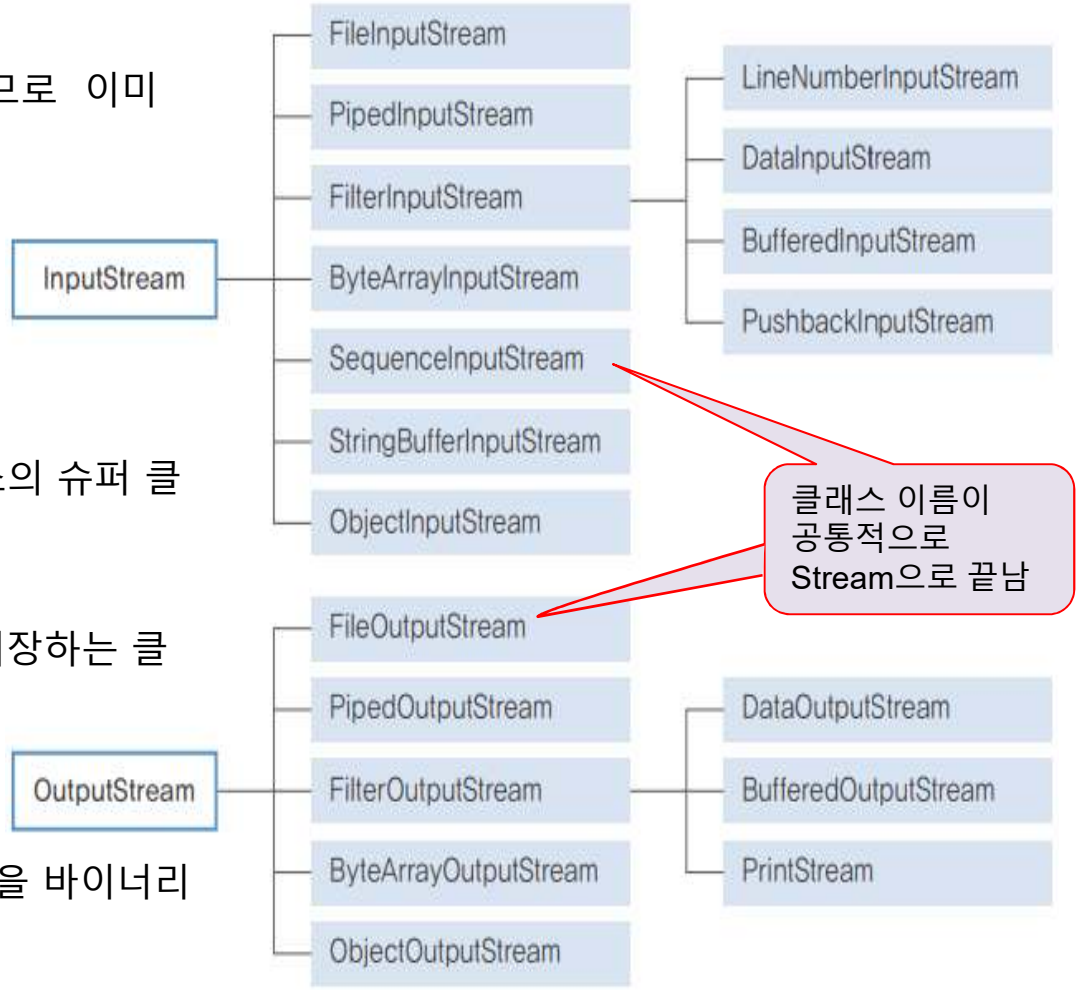
입출력 스트림

- 입출력 스트림의 사용



바이트 스트림

- **기초**
  - 바이트 단위의 이진 데이터를 다루므로 이미지나 동영상 파일을 처리할 때 유용
- **종류**
- **바이트 스트림 클래스**
  - InputStream/OutputStream
    - 추상 클래스
    - 바이트 스트림을 다루는 모든 클래스의 슈퍼 클래스
  - FileInputStream/FileOutputStream
    - 파일로부터 바이트 단위로 읽거나 저장하는 클래스
    - 바이너리 파일의 입출력 용도
  - DataInputStream/DataOutputStream
    - 자바의 기본 데이터 타입의 값(변수)을 바이너리 값 그대로 입출력
    - 문자열도 바이너리 형태로 입출력

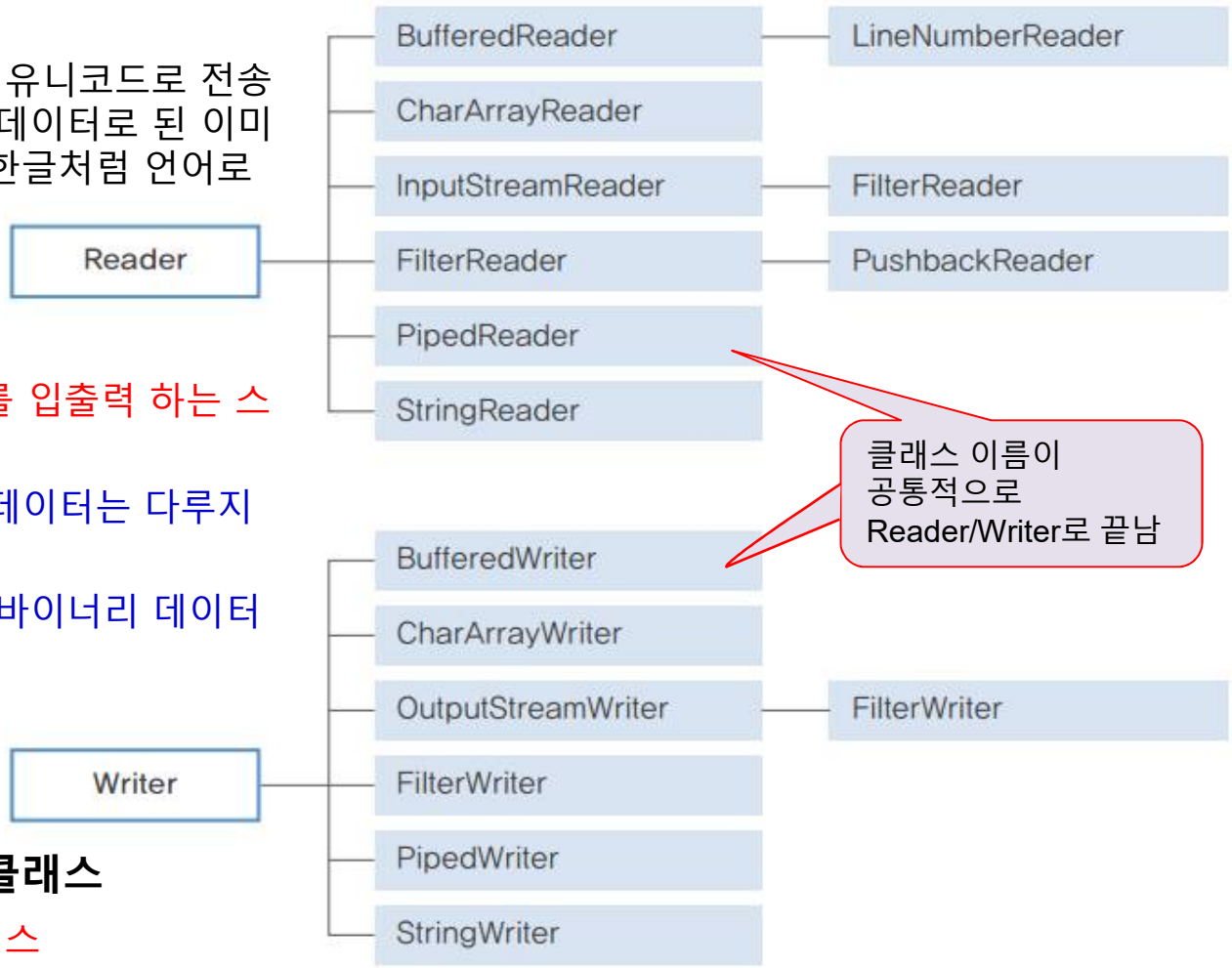


문자 스트림

기초

- 데이터를 2바이트 단위인 유니코드로 전송하거나 수신하는데, 이진 데이터로 된 이미지나 동영상 파일보다는 한글처럼 언어로 된 파일을 처리할 때 유용
- 유니 코드(2바이트) 문자를 입출력 하는 스트림
  - 문자로 표현되지 않는 데이터는 다루지 못함
  - 이미지, 동영상과 같은 바이너리 데이터는 입출력 할 수 없음

종류



문자 스트림을 다루는 클래스

- Reader/Writer : 추상 클래스
- InputStreamReader/OutputStreamWriter
- FileReader/FileWriter
  - 텍스트 파일에서 문자 데이터 입출력

바이트 스트림

InputStream과 OutputStream : 추상 클래스

- 모든 자식 바이트 스트림에서 공통으로 사용하는 메서드를 포함한 바이트 스트림의 최상위 클래스
- 각각 read( )와 write( )라는 추상 메서드를 포함
- 바이트 스트림 클래스가 제공하는 주요 메서드

클래스	메서드	설명
InputStream	int available( )	읽을 수 있는 바이트의 개수를 반환한다.
	void close( )	입력 스트림을 닫는다.
	abstract int read( )	1바이트를 읽는다.
	int read(byte b[ ])	1바이트씩 읽어 b[ ]에 저장한 후 읽은 개수를 반환한다.
	int read(byte b[ ], int off, int len)	len만큼 읽어 b[ ]의 off 위치에 저장한 후 읽은 개수를 반환한다.
	long skip(long n)	입력 스트림을 n바이트만큼 건너뛴다.
OutputStream	void close( )	출력 스트림을 닫는다.
	void flush( )	출력하려고 버퍼의 내용을 비운다.
	abstract void write(int b)	b 값을 바이트로 변환해서 쓴다.
	void write(byte b[ ])	b[ ] 값을 바이트로 변환해서 쓴다.
	void write(byte b[ ], int off, int len)	b[ ] 값을 바이트로 변환해서 off 위치부터 len 만큼 쓴다.

바이트 스트림

InputStream과 OutputStream

- read( ) 메서드의 반환 값은 0~255의 ASCII 값이며, 더 이상 읽을 데이터가 없을 때는 -1을 반환
- read( ) 메서드는 int 타입을 반환
- write( ) 메서드에서 인수가 배열일 때는 byte[ ], 배열이 아닐 때는 int 타입

- 대부분의 운영체제나 JVM은 표준 출력 장치를 효율적으로 관리하려고 버퍼를 사용
- BufferedStream이 아니지만 System.out은 표준 출력이므로 버퍼를 사용
- System.out을 사용해 출력할 때는 버퍼를 비우기 위하여 flush( ) 호출 필요

바이트 스트림

InputStream과 OutputStream

- 예제13-1 : sec02/IOStreamDemo

```
--- 입력 스트림 ---
hello
h(104)e(101)l(108)l(108)o(111)

--- 출력 스트림 ---
hello
```

키보드로 입력한 hello와 Enter이다.

```
public abstract int read() throws IOException
public void write(byte[] b) throws IOException
```

```
package sec02;

import java.io.IOException;

public class IOStreamDemo {
   public static void main(String[] args) throws IOException {
      int b, len= 0;
      int ba[] = new int[100];
      System.out.println("--- 입력 스트림 ---");
      while((b= System.in.read()) != '\n') {
         System.out.printf("%c(%d)", (char) b, b);
         ba[len++] = b;
      }
      System.out.println("\n\n--- 출력 스트림 ---");
      for(int i= 0; i< len; i++)
         System.out.write(ba[i]);
      System.out.flush();  // System.out.close();
//버퍼에 남아있는 데이터를 비워 표준 출력 장치로 전송한다.
// flush()대신에 주석 처리된 close()도 무방하다.
   }
}
```

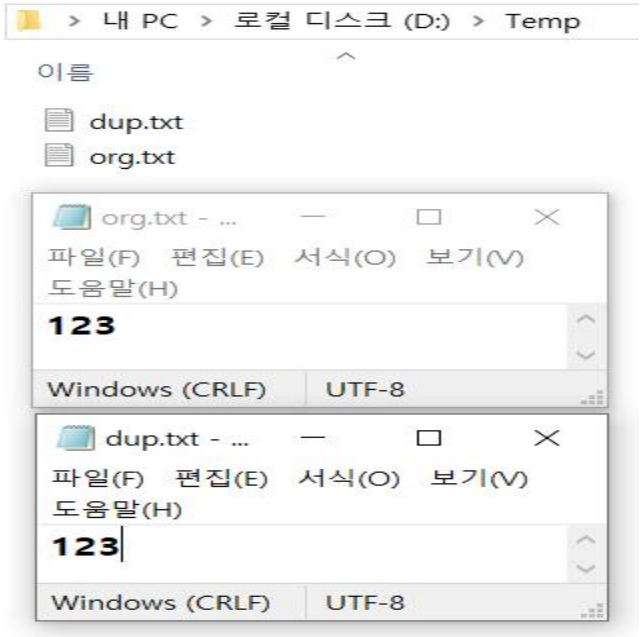
- Read()와 write()가 IOException을 유발할 수 있으므로 main()에 throws IOException 을 추가한다.

바이트 스트림

FileInputStream및 FileOutputStream

- 시스템에 있는 모든 파일을 읽거나 쓸 수 있는 기능을 제공
- 생성자로 스트림 객체를 생성할 때는 FileNotFoundException 예외 가능성이 있기 때문에 반드시 예외 처리 필요
- 예제13-2 : sec02/CopyFileDemo

```
FileInputStream(String name)        파일 시스템의 경로를 나타내는 문자열이다.
FileInputStream(File file)
FileOutputStream(String name)
FileOutputStream(File file)
FileOutputStream(String name, boolean append)    true면 이어쓰고(append), false면 덮어쓴다(overwrite).
FileOutputStream(File file, boolean append)
```




```
package sec02;
import java.io.*;
public class CopyFileDemo {
    public static void main(String[] args) {
        String input = "D:\\Temp\\org.txt";
        String output = "D:\\Temp\\dup.txt";

        try (FileInputStream fis = new FileInputStream(input);
             FileOutputStream fos = new FileOutputStream(output)) {
            int c;
            while ((c = fis.read()) != -1)  //한 바이트씩 읽고, 한 바이트씩 쓴다.
                fos.write(c);
        } catch (IOException e) {
        }
    }
}
```

- 파일 경로 문자열이나 파일 객체를 생성자의 매개변수로 사용해 FileInputStream과 FileOutputStream 객체를 생성할 수 있다.

바이트 스트림(예제13-2: quiz_1_식별자)

```
package sec02;  //예외처리 방법1 => 자원 반환 필요
import java.io.*;
public class quiz_1_1_111 {
   public static void main(String[] args) {
      String input = "D:\\Temp\\org.txt";
      String output = "D:\\Temp\\dup.txt";
      try {FileInputStream fis = new FileInputStream(input);
           FileOutputStream fos = new FileOutputStream(output);
          int c;
          while ((c = fis.read()) != -1)  //한 바이트씩 읽고, 한 바이트씩 쓴다.
             {System.out.print((char) c); fos.write(c); }
       fis.close();  fos.close();
       } catch (IOException e) {       }
   }  }
```

```
package sec02; //예외처리 방법2 => 자동 자원 반납
import java.io.*;
public class quiz_1_2_111 {
   public static void main(String[] args) {
      String input = "D:\\Temp\\org.txt";
      String output = "D:\\Temp\\dup.txt";
      try (FileInputStream fis = new FileInputStream(input);
           FileOutputStream fos = new FileOutputStream(output)) {
          int c;
          while ((c = fis.read()) != -1)  //한 바이트씩 읽고, 한 바이트씩 쓴다.
              {System.out.print((char) c); fos.write(c); }
      } catch (IOException e) {       }
   }  // 자동으로 close()메서드를 호출한다.
}
```

- 파일 경로 문자열이나 파일 객체를 생성자의 매개변수로 사용해 FileInputStream과 FileOutputStream 객체를 생성할 수 있다.
- 생성자로 스트림 객체를 생성할 때는 FileNotFoundException 예외 가능성이 있기 때문에 반드시 예외 처리를 해야 한다.
  - 예외 처리 방법

방법1)
```
try {
} catch (IOException e) {
}
```

방법2)
```
try (리소스) {
} catch (IOException e ) {
}
```

방법1) try 블록에서 파일 등과 같은 리소스를 사용한다면 try 블록을 실행한 후 자원 반환 필요

리소스를 관리하는 코드를 추가하면 가독성도 떨어지고, 개발자도 번거롭다.

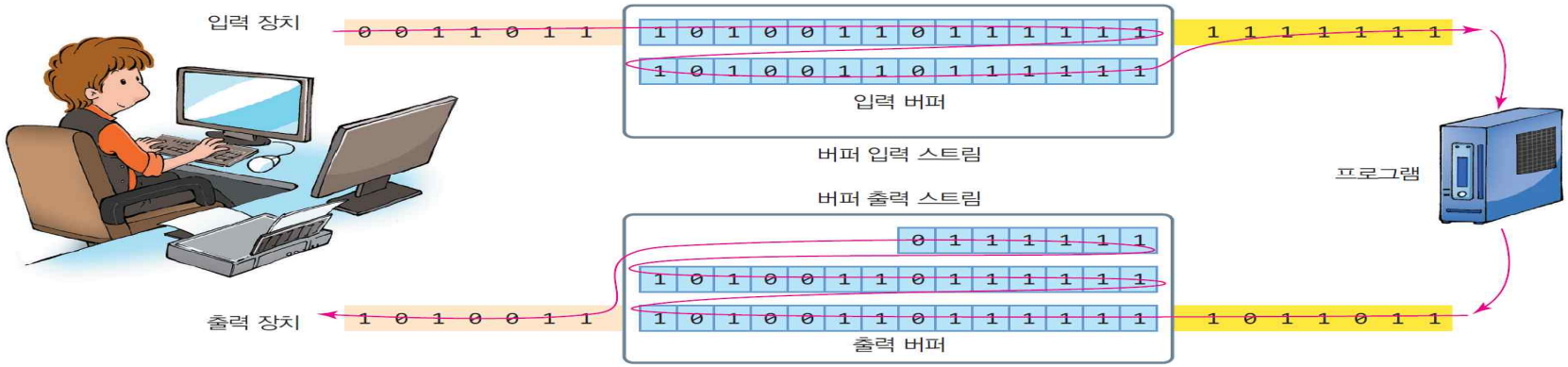
방법2) try~with~resource 문

=> 만약 try-with-resources 구문을 사용한다면 조금 더 깔끔하게 표현이 가능하다. try() 괄호 구문안에 해제되어야 할 생성 객체를 포함하는 것이 특징입니다. 예외 발생 여부와 상관없이 사용한 리소스를 자동 반납하는 수단 제공한다. 단, AutoCloseable 인터페이스를 구현하는 예외에서만 동작합니다.

바이트 스트림

BufferedInputStream및 BufferedOutputStream

- 버퍼는 스트림과 프로그램 간에 데이터를 효율적으로 전송하려고 사용하는 메모리
- 입출력 장치와 프로그램 간 동작 속도가 크게 차이가 날 때 버퍼를 사용하면 매우 효율적



바이트 스트림

BufferedInputStream및 BufferedOutputStream

- 생성자

```
BufferedInputStream(InputStream in)
BufferedInputStream(InputStream in, int size)
BufferedOutputStream(OutputStream out)
BufferedOutputStream(OutputStream out, int size)
```
버퍼의 크기를 나타낸다.

- 예제 13-3:

sec02/BufferedStreamDemo

```
버퍼를 사용한 경우 : 1369231621
버퍼를 사용하지 않은 경우 : 3691319106
```

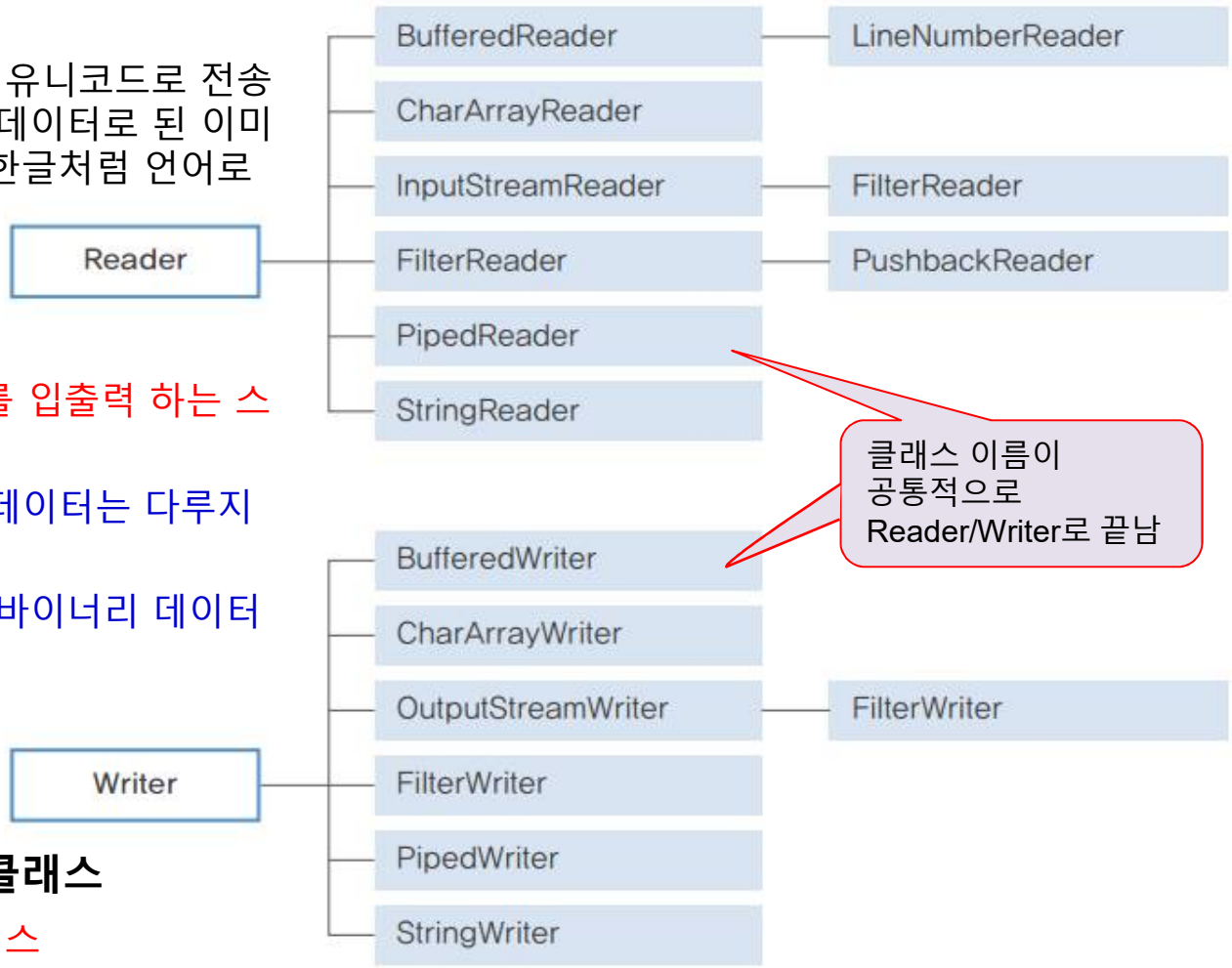
```
package sec02;
import java.io.*;
public class BufferedStreamDemo {
   public static void main(String[] args) {
      long start, end, duration;
      String org = "C:\\Program Files (x86)\\Internet Explorer\\iexplore.exe";
      String dst = "D:\\Temp\\iexplore1.exe";
      start = System.nanoTime();
      try (BufferedInputStream bis = new BufferedInputStream(new
FileInputStream(org));
          BufferedOutputStream bos = new BufferedOutputStream(new
FileOutputStream(dst));) {
         while (bis.available() > 0) {
            int b = bis.read();
            bos.write(b);
         }
         bos.flush();
      } catch (IOException e) {
      }
      end = System.nanoTime();
      duration = end - start;
      System.out.println("버퍼를 사용한 경우 : " + duration);
      start = System.nanoTime();
      try (FileInputStream fis = new FileInputStream(org); FileOutputStream fos = new
FileOutputStream(dst);) {
         while (fis.available() > 0) {
            int b = fis.read();
            fos.write(b);
         }
         fos.flush();
      } catch (IOException e) {
      }
      end = System.nanoTime();
      duration = end - start;
      System.out.println("버퍼를 사용하지 않은 경우  : " + duration);
   }
}
```

문자 스트림

기초

- 데이터를 2바이트 단위인 유니코드로 전송하거나 수신하는데, 이진 데이터로 된 이미지나 동영상 파일보다는 한글처럼 언어로 된 파일을 처리할 때 유용
- 유니 코드(2바이트) 문자를 입출력 하는 스트림
  - 문자로 표현되지 않는 데이터는 다루지 못함
  - 이미지, 동영상과 같은 바이너리 데이터는 입출력 할 수 없음

종류



문자 스트림을 다루는 클래스

- Reader/Writer : 추상 클래스
- InputStreamReader/OutputStreamWriter
- FileReader/FileWriter
  - 텍스트 파일에서 문자 데이터 입출력

문자 스트림

기초

- Reader와 Writer는 객체를 생성할 수 없는 추상 클래스이기 때문에 Reader와 Writer의 자식인 구현 클래스를 사용
- FileReader와 FileWriter는 파일 입출력 클래스로, 파일에서 문자 데이터를 읽거나 파일에 문자 데이터를 저장할 때 사용
- InputStreamReader및 OutputStreamWriter는 바이트 스트림과 문자 스트림을 연결하는 브리지 스트림으로 사용
- BufferedReader와 BufferedWriter는 데이터를 효율적으로 전송하려고 버퍼로 처리할 때 사용

문자 스트림

Reader와 Writer

- 추상 메서드인 read( ), close( )와 write( ), flush( ), close( )를 각각 포함하는 추상 클래스
- 문자 스트림 클래스가 제공하는 주요 메서드

클래스	메서드	설명
Reader	abstract void close( )	입력 스트림을 닫는다.
	int read( )	1개의 문자를 읽는다.
	int read(char[ ] cbuf)	문자 단위로 읽어 cbuf[ ]에 저장한 후 읽은 개수를 반환한다.
	abstract int read(char cbuf[ ], int off, int len)	len만큼 읽어 cbuf[ ]의 off 위치에 저장한 후 읽은 개수를 반환한다.
	long skip(long n)	입력 스트림을 n 문자만큼 건너뛴다.
Writer	abstract void close( )	스트림을 닫고 관련된 모든 자원을 반납한다.
	abstract void flush( )	버퍼의 내용을 비운다.
	void write(int c)	c 값을 char로 변환해 출력 스트림에 쓴다.
	void write(char cbuf[ ])	cbuf[ ] 값을 char로 변환해 출력 스트림에 쓴다.
	abstract void write(char cbuf[ ], int off, int len)	cbuf[ ] 값을 char로 변환해 off 위치부터 len만큼 출력 스트림에 쓴다.
	void write(String str)	문자열 str을 출력 스트림에 쓴다.

문자 스트림

FileReader와 FileWriter

- 시스템에 있는 모든 문자 파일을 읽거나 파일에 쓸 수 있는 기능을 제공
- 텍스트 파일에서 문자 데이터 입출력
- 생성자로 스트림 객체를 생성할 때는 FileNotFoundException 예외 처리 필요
- 생성자

```
FileReader(String name)        파일 시스템의 경로를 나타내는 문자열이다.
FileReader(File file)
FileWriter(String name)
FileWriter(File file)
FileWriter(String name, boolean append)    true면 이어쓰고(append), false면 덮어쓴다(overwrite).
FileWriter(File file, boolean append)
```

- 예제 : sec03/CopyFileDemo

```
//D:₩Temp₩org.txt
abcd
efg
```

```
//D:₩Temp₩dup.txt
abcd
efg
```

```
package sec03;

import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class CopyFileDemo {
   public static void main(String[] args) {
      String input = "D:\\Temp\\org.txt";
      String output = "D:\\Temp\\dup.txt";

      try (FileReader fr = new FileReader(input);
           FileWriter fw = new FileWriter(output)) {
         int c;

         while ((c = fr.read()) != -1)
            fw.write(c);
      } catch (IOException e) {
      }
   }
}
```

D:₩Temp₩org.txt 파일을 열고 파일과 입력 바이트 스트림 객체 fr 연결

파일 끝까지 바이트씩 c에 읽어 들임. 파일의 끝을 만나면 read()는 -1 리턴

문자 스트림

BufferedReader및 BufferedWriter

- 스트림의 효율을 높이려고 버퍼를 사용
- 생성자

```
BufferedInputStream(InputStream in)
BufferedInputStream(InputStream in, int size)
BufferedOutputStream(OutputStream out)
BufferedOutputStream(OutputStream out, int size)
```

버퍼의 크기를 나타낸다.

- BufferedReader 클래스에 추가된 주요 메서드

메서드	설명
Stream<String> lines( )	읽은 행을 스트림으로 반환한다.
String readLine( )	한 행을 읽어 문자열로 반환한다.

- 예제 : sec03/BufferedReaderDemo

```
package sec03;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class BufferedReaderDemo {
    public static void main(String[] args) {
        try (BufferedReader br = new BufferedReader(new
FileReader("D:\\Temp\\org.txt"));) {

            br.lines().forEach(s -> System.out.println(s));
        } catch (IOException e) {
        }
    }
}
```

```
//D:₩Temp₩org.txt
abcd
efg
```

```
abcd
efg
```

파일 관리

기초

- 입출력 스트림은 파일이나 장치를 읽거나 쓰기 위해 사용
- 입출력 스트림에 파일 생성 혹은 삭제, 파일 속성 변경 등의 관리 기능은 없음
- 자바는 파일을 관리하기 위해 File 클래스를 제공
- 여전히 File 클래스를 많이 사용하지만, 자바 4부터 도입되고 자바 7에서 기능을 보완한 NIO(java.nio) 및 NIO2(java.nio2) 기반의 Path 인터페이스, Files 클래스 및 FileChannel도 유용

파일 관리

File 클래스

- 파일이나 폴더의 경로를 추상화한 클래스로 java.io 패키지에 포함
- 파일 유무, 삭제, 접근 권한 조사 등을 수행
- File 클래스 생성자

생성자	설명
File(File parent, String child)	parent 객체 폴더의 child라는 File 객체를 생성한다.
File(String pathname)	pathname에 해당하는 File 객체를 생성한다.
File(String parent, String child)	parent 폴더에 child라는 File 객체를 생성한다.
File(URI uri)	uri 경로에서 File 객체를 생성한다.

파일 관리

- **File 클래스**
  - File 클래스의 주요 메서드

메서드	설명
boolean canExecute( )	실행 가능한 파일인지 여부를 반환한다.
boolean canRead( )	읽을 수 있는 파일인지 여부를 반환한다.
boolean canWrite( )	쓸 수 있는 파일인지 여부를 반환한다.
boolean createNewFile( )	파일을 새로 생성하면 true, 아니면 false를 반환한다.
boolean delete( )	파일을 삭제하면 true, 아니면 false를 반환한다.
boolean exists( )	파일의 존재 유무를 반환한다.
String getAbsolutePath( )	파일의 절대 경로를 반환한다.
String getName( )	파일의 이름을 반환한다.
String getPath( )	파일의 경로를 반환한다.
boolean isDirectory( )	폴더 존재 유무를 반환한다.
boolean isFile( )	파일 존재 유무를 반환한다.
long lastModified( )	파일의 마지막 수정 시간을 반환한다.
long length( )	파일의 크기를 반환한다.
String[ ] list( )	모든 자식 파일과 폴더를 문자열 배열로 반환한다.
File[ ] listFiles( )	모든 자식 파일과 폴더를 File 배열로 반환한다.
boolean mkdir( )	폴더를 생성하면 true, 아니면 false를 반환한다.
Path toPath( )	파일 경로에서 구성한 Path 객체를 반환한다.

파일 관리

File 클래스

- 예제 : sec04/FileDemo

```
dir : C:\Windows\addins
dir : C:\Windows\appcompat
dir : C:\Windows\AppPatch
dir : C:\Windows\AppReadiness
dir : C:\Windows\assembly
dir : C:\Windows\bcastdvr
file: C:\Windows\bfsvc.exe(61440 bytes)
...
```

```
package sec04;

import java.io.File;
import java.io.IOException;

public class FileDemo {
   public static void main(String[] args) throws IOException {
      File file = new File("C:\\Windows");
      File[] fs = file.listFiles();

      for (File f : fs)
         if (f.isDirectory())
            System.out.printf("dir : %s\n", f);
         else
            System.out.printf("file: %s(%d bytes)\n", f, f.length());
   }
}
```

파일 관리

Path 인터페이스

- 운영체제에 따라 일관성 없이 동작하는 File 클래스를 대체하는 것
- 기존 File 객체도 File 클래스의 toPath( ) 메서드를 이용해 Path 타입으로 변환 가능
- Path 인터페이스의 구현 객체는 파일 시스템에서 경로를 나타낸다.
- java.nio.file 패키지에 포함
- java.io.File 클래스와 마찬가지로 파일의 유무, 삭제, 접근 권한 조사 등이 주요 기능

메서드	설명
Path getFileName( )	객체가 가리키는 파일(폴더) 이름을 반환한다.
FileSystem getFileSystem( )	객체를 생성한 파일 시스템을 반환한다.
int getNameCount( )	객체가 가리키는 경로의 구성 요소 개수를 반환한다.
Path getParent( )	부모 경로를 반환하며, 없으면 null을 반환한다.
Path getRoot( )	루트를 반환하며, 없으면 null을 반환한다.
boolean isAbsolute( )	절대 경로 여부를 반환한다.
Path toAbsolutePath( )	절대 경로를 나타내는 객체를 반환한다.
URI toUri( )	객체가 가리키는 경로에서 URI를 반환한다.

파일 관리

Files 클래스

- 파일 연산을 수행하는 정적 메서드로 구성된 클래스
- java.nio.file 패키지에 포함

Paths 클래스

- 정적 메서드
  Path get(String first, String... more)

파일 관리

Files 클래스

- 주요 메서드

메서드	설명
long copy( )	파일을 복사한 후 복사된 바이트 개수를 반환한다.
Path copy( )	파일을 복사한 후 복사된 경로를 반환한다.
Path createDirectory( )	폴더를 생성한다.
Path createFile( )	파일을 생성한다.
void delete( )	파일을 삭제한다.
boolean deleteIfExists( )	파일이 있으면 삭제한다.
boolean exists( )	파일의 존재 유무를 조사한다.
boolean isDirectory( )	폴더인지 조사한다.
boolean isExecutable( )	실행 가능한 파일인지 조사한다.
boolean isHidden( )	숨김 파일인지 조사한다.
boolean isReadable( )	읽기 가능한 파일인지 조사한다.
boolean isWritable( )	쓰기 가능한 파일인지 조사한다.
Path move( )	파일을 이동한다.
boolean notExists( )	파일(폴더)의 부재를 조사한다.
byte[ ] readAllBytes( )	파일의 모든 바이트를 읽어 배열로 반환한다.
List<String> readAllLines( )	파일의 모든 행을 읽어 리스트로 반환한다.
long size( )	파일의 크기를 반환한다.
Path write( )	파일에 데이터를 쓴다.

quiz_2_식별자: 문자 스트림의 FileWriter로 텍스트 파일 저장

Scanner로 입력 받은 이름과 전화번호를 한 중에 한 사람씩 d:₩Temp₩phone.txt파일에 5개 저장하는 프로그램을 작성하시오. 그만을 입력하면 프로그램을 종료한다.
=> 문자 스트림의 FileWriter로 텍스트 파일 저장

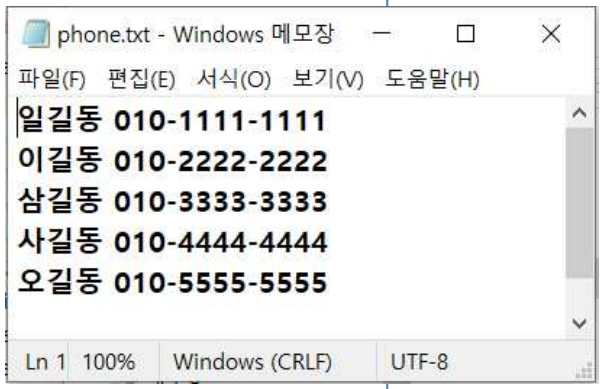
```
package w14_111;
import java.io.*;
import java.util.*;

public class quiz2_111 {
    public static void main(String[] args) {
        FileWriter fw = null;
        File f = new File("d:₩₩temp₩₩phone.txt");
        try {
            fw = new FileWriter(f);
            Scanner scanner = new Scanner(System.in);

            System.out.println("전화번호 입력 프로그램입니다.");
            while (true) {
                System.out.print("이름 전화번호 >> ");
                String line = scanner.nextLine(); // 한줄을 읽는다.
                if (line.equals("그만"))
                    break; // 입력 종료
                fw.write(line + "₩r₩n"); // 한 줄 띄어 저장하기 위해 "₩r₩n"을 붙인다.
            }
            System.out.println(f.getPath() + "에 저장하였습니다.");

            scanner.close();
            fw.close();
        } catch (IOException e) { // 파일을 저장할 수 없는 경우 예외
            e.printStackTrace();
        }
    }
}
```

```
전화번호 입력 프로그램입니다.
이름 전화번호 >> 일길동 010-1111-1111
이름 전화번호 >> 이길동 010-2222-2222
이름 전화번호 >> 삼길동 010-3333-3333
이름 전화번호 >> 사길동 010-4444-4444
이름 전화번호 >> 오길동 010-5555-5555
이름 전화번호 >> 그만
d:\temp\phone.txt에 저장하였습니다.
```

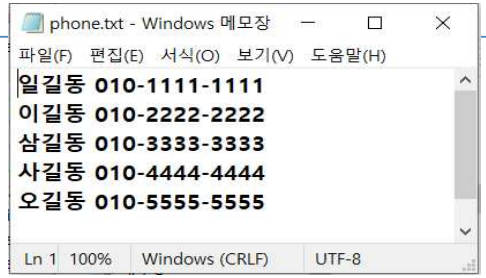


quiz_3_식별자: 문자 스트림의 FileReader로 텍스트 파일 읽기

quiz_2_식별자 문제에서 d:₩Temp₩phone.txt파일에 저장한 내용을 읽어 화면에 출력하는 프로그램을 작성하시오.
=> 문자 스트림의 FileReader로 텍스트 파일 읽기와 Scanner를 이용한 파일 읽기

```
//문자 스트림의 FileReader로 텍스트 파일 읽기
package w14_111;
import java.io.*;

public class quiz3_1_111 {
    public static void main(String[] args) {
        FileReader fr = null;
        File f = new File("d:₩₩temp₩₩phone.txt");
        try {
            fr = new FileReader(f);
            System.out.println(f.getPath() + "를 출력합니다.");
            while(true) {
                int c = fr.read();
                if(c == -1)
                    break;
                System.out.print((char)c);
            }
            fr.close();
        }
        catch (IOException e) { // 파일을 저장할 수 없는 경우 예외
            e.printStackTrace();
        } }
}
```




```
//Scanner를 이용한 파일 읽기
package w14_111;
import java.io.*;
import java.util.*;
public class quiz3_2_111 {
    public static void main(String[] args) {
    FileReader fr = null;
        File f = new File("c:₩₩temp₩₩phone.txt");
        try {
            fr = new FileReader(f);
            Scanner scanner = new Scanner(fr);
            System.out.println(f.getPath() + "를 출력합니다.");
            while(scanner.hasNext()) {
                String line = scanner.nextLine(); // 한줄을 읽는다.
                System.out.println(line);
            }
            fr.close();
            scanner.close();
        }       catch (IOException e) {
            e.printStackTrace();
        } }
}
```

```
d:\temp\phone.txt를 출력합니다.
일길동 010-1111-1111
이길동 010-2222-2222
삼길동 010-3333-3333
사길동 010-4444-4444
오길동 010-5555-5555
```

quiz_4_식별자: HashMap에 객체 저장, 학생 정보 관리

id와 전화번호로 구성되는 Student 클래스를 만들고, 이름을 '키'로 하고 Student 객체를 '값'으로 하는 해시맵을 작성하라.

```
package w14_111;
import java.util.*;
class Student { // 학생을 표현하는 클래스
    int id;
    String tel;
    public Student(int id, String tel) {    this.id = id; this.tel = tel;    }
   public int getId() {     return id;    }
    public String getTel() {      return tel;   }
}
public class quiz_4_111 {
    public static void main(String[] args) {
        // 학생 이름과 Student 객체를 쌍으로 저장하는 HashMap 컬렉션 생성
        HashMap<String, Student> map = new HashMap<String, Student>();
        // 3 명의 학생 저장
        map.put("일길동", new Student(1, "010-111-1111"));
        map.put("이길동", new Student(2, "010-222-2222"));
        map.put("삼길동", new Student(3, "010-333-3333"));
        Scanner scanner = new Scanner(System.in);
        while(true) {
            System.out.print("검색할 이름?");
            String name = scanner.nextLine(); // 사용자로부터 이름 입력
            if(name.equals("exit"))
                break; // while 문을 벗어나 프로그램 종료
            Student student = map.get(name); // 이름에 해당하는 Student 객체 검색
            if(student == null)
                System.out.println(name + "은 없는 사람입니다.");
            else
                System.out.println("id:" + student.getId() + ", 전화:" + student.getTel());
        }
        scanner.close();
    }
}
```

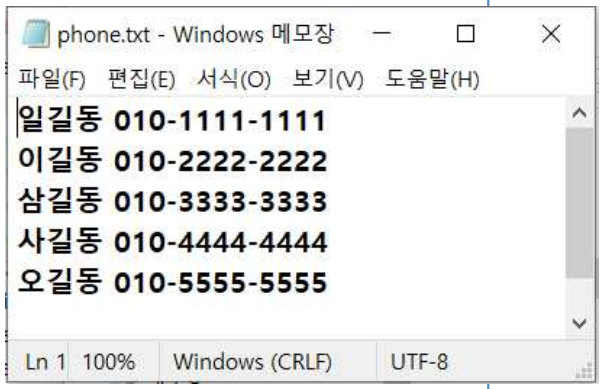
```
검색할 이름?이길동
id:2, 전화:010-222-2222
검색할 이름?삼길동
id:3, 전화:010-333-3333
검색할 이름?exit
```

quiz_5_식별자: HashMap 컬렉션과 파일 입출력 활용

전화번호를 미리 d:₩Temp₩phone.txt파일에 5개 저장해 둔다. 이 파일을 읽어 다음 실행 예시와 같은 작동하는 검색 프로그램을 작성하시오.
=> 이름을 '키'로 하고 전화 번호를 '값'으로 하는 HashMap<string, String>을 이용하고, 하라인 씩 읽어 이름과 전화번호를 해시맵에 저장하고, 사용자의 이름을 입력 받아 검색한다.

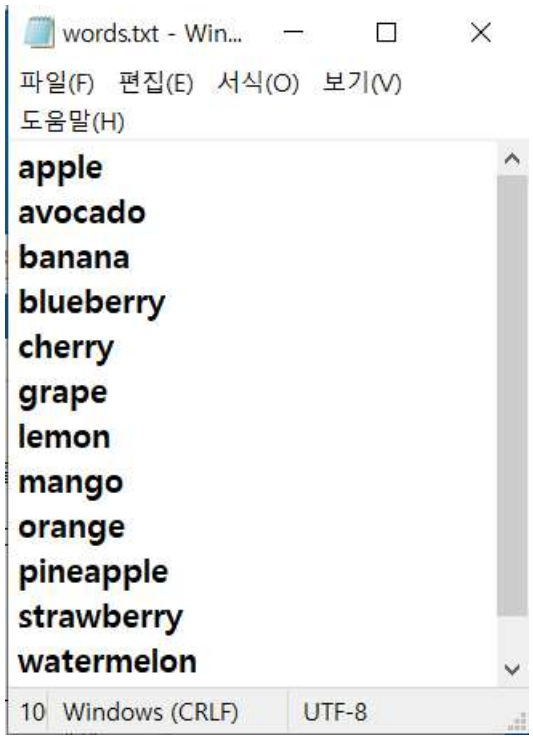
```
package w14_111;
import java.io.*;
import java.util.*;
public class quiz5_111 {
    private String fileName = "D:\\temp\\phone.txt";
    private HashMap<String, String> phoneMap = new HashMap<String, String>();
    public quiz5_111() { }
    private void readPhoneFile() {
        try {    Scanner fScanner = new Scanner(new FileReader(new File(fileName)));
            while (fScanner.hasNext()) {
                String name = fScanner.next(); // 이름 읽기
                String tel = fScanner.next(); // 전화번호 읽기
                phoneMap.put(name, tel); // 해시맵에 저장
            }
            fScanner.close();
        } catch (IOException e) { // 파일을 저장할 수 없는 경우 예외
            e.printStackTrace();
        }
        System.out.println("총 " + phoneMap.size() + "개의 전화번호를 읽었습니다."); }
    private void processQuery() {
        Scanner scanner = new Scanner(System.in);
        while (true) {
            System.out.print("이름>> ");
            String name = scanner.next(); // 이름 입력
            if (name.equals("그만"))       break;
            String tel = phoneMap.get(name);
            if (tel == null) {
                System.out.println("찾는 이름이 없습니다.");
            } else {
                System.out.println(tel);            }        }
        scanner.close();  }
    public void run() {
        readPhoneFile();
        processQuery();  }
    public static void main(String[] args) {
        quiz5_111 phoneExplorer = new quiz5_111();
        phoneExplorer.run(); }    }
```

```
총 5개의 전화번호를 읽었습니다.
이름>> 삼길동
010-3333-3333
이름>> 오길동
010-5555-5555
이름>> 이길동
010-2222-2222
이름>> 그만
```



quiz_6_식별자: 행맨게임 만들기(BufferedReader 활용)

주어진 단어를 문자 하나씩 추측해서 맞추는 행맨(hangman)프로그램을 작성하시오.
- 처음에는 단어에 포함된 문자의 개수만큼 빈칸이 나타나며, 사용자는 빈칸에 들어갈 문자를 하나씩 추측한다.
- 추측한 문자가 맞으면 빈칸 대신에 맞춘 문자를 출력한다. 프로그램에서 사용할 문자열은 12개의 단어로 구성된 D:₩Temp₩words.txt 파일에 있는 문자열 중 무작위로 선택한다.
- 여섯 번을 초과해서 잘못된 추측을 하면 게임이 종료된다.
- 아래 결과창은 최초 화면과 하나의 문자를 맞춘 실행결과이다.




```
추측할 단어입니다 : ------
지금까지 추측한 내용입니다 :
추측한 문자를 입력하세요 : a
추측을 잘못했습니다 - 5번 더 추측할 수 있습니다.
추측할 단어입니다 : ------
지금까지 추측한 내용입니다 : a
추측한 문자를 입력하세요 : b
추측을 잘못했습니다 - 4번 더 추측할 수 있습니다.
추측할 단어입니다 : ------
지금까지 추측한 내용입니다 : ab
추측한 문자를 입력하세요 : o
추측을 잘못했습니다 - 3번 더 추측할 수 있습니다.
추측할 단어입니다 : ------
지금까지 추측한 내용입니다 : abo
추측한 문자를 입력하세요 : c
정확한 추측입니다 - 3번 더 추측할 수 있습니다.
추측할 단어입니다 : c-----
지금까지 추측한 내용입니다 : aboc
추측한 문자를 입력하세요 : cherry
승리!
한 번 더 게임할래요 (y/n)?:
```

quiz_6_식별자: 행맨게임 만들기(BufferedReader 활용)

```
package w14_111;  //1
import java.io.*;
import java.util.*;
public class quiz6_111 {
public static void main(String[] args) throws IOException {
Scanner in = new Scanner(System.in);
char again = 'n';
String secret;
StringBuffer dashes;
int leftCount;
boolean done;
String guess;
String guesses;
char letter;
Words words = new Words("D:\\temp\\words.txt");
do {
        secret = words.getRandomWord();
        guesses = "";
        done = false;
        leftCount = 6;
        dashes = makeDashes(secret);
```

```
class Words {  //3
        private String fileName;
        private Random r = new Random();
        public Words(String fileName) {
        this.fileName = fileName;
        }
        public String getRandomWord() {
        String line = null;
        int n = r.nextInt(10);
        try (BufferedReader in = new BufferedReader(new
        FileReader(fileName));) {
        while (n-- >= 0)
        line = in.readLine();
        } catch (Exception e) {
        }
        return line;
        }
}
```

```
while (!done) {   //2
        System.out.println("추측할 단어입니다 : " + dashes);
        System.out.println("지금까지 추측한 내용입니다 : " + guesses);
        System.out.print("추측한 문자를 입력하세요 : ");
        guess = in.next();
        if (guess.length() > 1) {
        if (guess.equals(secret))
        System.out.println("승리!");
        else
        System.out.println("실패.");
        done = true;
        } else {
        letter = guess.charAt(0);
        guesses += letter;
        if (secret.indexOf(letter) < 0) {
        --leftCount;
        System.out.print("추측을 잘못했습니다 - ");
        } else
        matchLetter(secret, dashes, letter);
        System.out.println(leftCount + "번 더 추측할 수 있습니다.");
        if (leftCount == 0) {
        System.out.println("실패.");
        done = true;
        }
        if (secret.equals(dashes.toString())) {
        System.out.println("승리!");
        done = true;}
        }
        }
        System.out.print("한 번 더 게임할래요 (y/n)?: ");
        again = in.next().charAt(0);
        } while (again == 'Y' || again == 'y');
}
public static void matchLetter(String secret, StringBuffer dashes, char
letter) {
        for (int index = 0; index < secret.length(); index++)
        if (secret.charAt(index) == letter)
        dashes.setCharAt(index, letter);
        System.out.print("정확한 추측입니다 - ");  }
public static StringBuffer makeDashes(String s) {
        StringBuffer dashes = new StringBuffer(s.length());
        for (int count = 0; count < s.length(); count++)
        dashes.append('-');
        return dashes;   }
}
```