



Java I/O (ver3.5.1)

목차 (Table of Contents)

1. Java I/O 소개
2. Java I/O 프로그래밍
3. Java I/O 이해
4. 자료구조와 알고리즘
5. 자료구조 기초
6. 알고리즘 - 정렬
7. 알고리즘 - 검색
8. 파일 데이터베이스

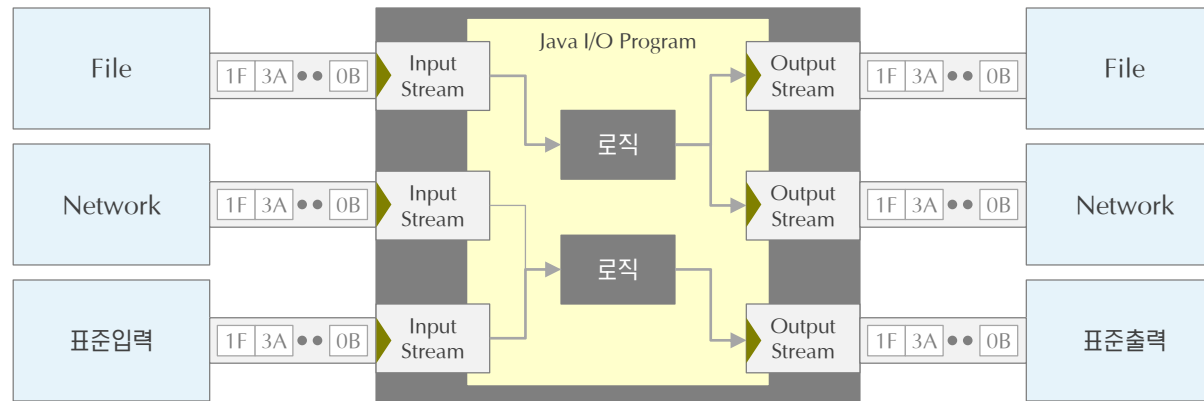


1. Java I/O 소개

- 1.1 Java I/O 소개
- 1.2 Java I/O API
- 1.3 표준 입출력

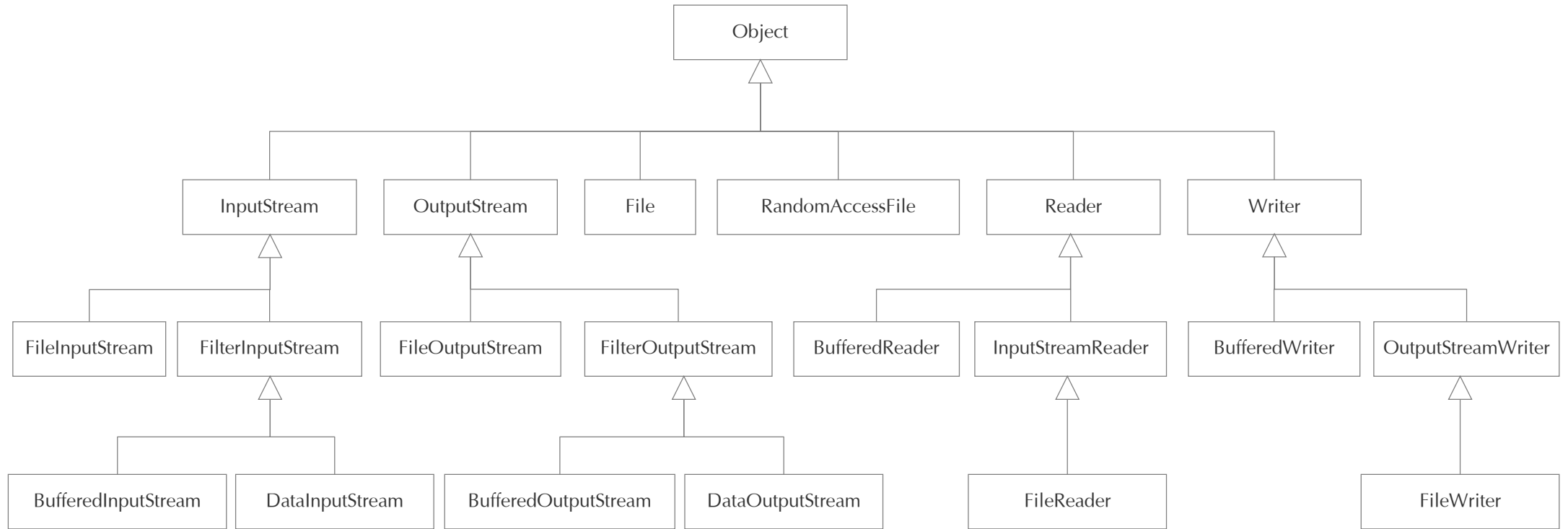
1.1 Java I/O 소개

- ✓ Java I/O 는 Java 프로그램에서 입력과 출력을 처리하는 분야 입니다.
- ✓ Java는 파일, 네트워크, 키보드, 모니터 등으로 입출력할 수 있는 다양한 API를 제공합니다.
- ✓ Java는 입출력을 일련의 바이트로 구성된 스트림(Stream) 기반으로 처리합니다.
- ✓ Java I/O API 와 New IO API 를 적절히 활용하면 입출력 성능을 개선할 수 있습니다.



1.2 Java I/O API

- ✓ java.io 패키지에는 파일과 입출력 스트림 등 Java I/O 관련 클래스가 있습니다.
- ✓ 스트림은 네트워크 또는 파일 등 외부에서 데이터를 입력 받거나 출력할 때 사용하는 일련의 바이트입니다.
- ✓ System 클래스에는 키보드와 모니터와 같은 표준 입출력 장치를 다루는 API가 있습니다.
- ✓ 기본적인 입출력 스트림 클래스와 기능을 확장할 수 있는 필터 스트림 클래스가 있습니다.



1.3 표준 입출력

- ✓ System.in은 InputStream 객체로, 표준 입력 장치(키보드)로부터 데이터를 입력 받습니다.
- ✓ System.out은 PrintStream 객체로, 표준 출력 장치(화면)에 데이터를 출력합니다.
- ✓ System.err은 에러 출력을 위한 특별한 객체로, 에러 내용을 표준 출력 장치에 출력합니다.
- ✓ Java는 하드웨어를 직접 제어할 수 없으므로 네이티브 코드와 연동하여 제어합니다.

```
1 System.out.println("1에서 5사이의 정수를 입력하세요.");
2 Scanner scanner = new Scanner(System.in);
3 int choice = scanner.nextInt();
4
5 if (choice >= 1 && choice <= 5) {
6     System.out.println("선택한 수는 " + choice + "입니다.");
7 } else {
8     System.err.println("1에서 5사이의 정수만 가능합니다.!");
9 }
```

StandardInOut.txt

코드설명

[Line2] 표준입력으로 부터 데이터를 입력받는 Scanner 객체를 생성합니다.
[Line3] Scanner 객체를 이용하여 사용자로 부터 정수 값을 입력 받습니다.
[Line6] 표준출력으로 결과를 출력합니다.
[Line8] 표준에러로 결과를 출력합니다.



2. Java I/O 프로그래밍

- 2.1 파일 정보 조회
- 2.2 파일 생성과 삭제
- 2.3 디렉토리 생성
- 2.4 디렉토리 삭제
- 2.5 파일 목록 조회
- 2.6 파일 복사

2.1 파일 정보 조회

- ✓ File 클래스는 파일과 디렉토리를 다루는 클래스입니다.
- ✓ File.getAbsolutePath() 메소드는 파일의 절대 경로를 문자열로 반환합니다.
- ✓ File.getCanonicalPath() 메소드는 파일의 전체 경로(파일을 나타내는 유일한 경로)를 반환합니다.
- ✓ System.getProperty() 메소드는 시스템 프로퍼티 값을 조회할 때 사용합니다.

```
1 public class FileInfoDisplay {
2
3     public static void main(String[] args) throws IOException {
4
5         System.out.println("user.dir : " + System.getProperty("user.dir"));
6
7         File file = new File("sample.txt");
8
9         System.out.println("getName() : " + file.getName());
10        System.out.println("getParent() : " + file.getParent());
11        System.out.println("getPath() : " + file.getPath());
12        System.out.println("getAbsolutePath() : " + file.getAbsolutePath());
13        System.out.println("getCanonicalPath() : " + file.getCanonicalPath());
14        System.out.println("length() : " + file.length() + " bytes");
15        System.out.println("isFile() : " + file.isFile());
16        System.out.println("isDirectory() : " + file.isDirectory());
17    }
18 }
```

FileInfoDisplay.java

코드설명

[Line5] 시스템 프로퍼티 user.dir 값을 조회하여 화면에 출력합니다.
[Line7] sample.txt 파일에 대한 객체를 생성합니다.
[Line 9~16] File 객체의 다양한 메소드를 실행하여 결과를 출력합니다.

실행결과

```
user.dir : C:\JavaIOExamples
getName() : sample.txt
getParent() : null
getPath() : sample.txt
getAbsolutePath() : C:\...\sample.txt
getCanonicalPath() : C:\...\sample.txt
length() : 18 bytes
isFile() : true
isDirectory() : false
```


2.2 파일 생성과 삭제

- ✓ `File.createNewFile()` 은 지정한 파일이 없는 경우 파일을 생성합니다.
- ✓ `File.delete()` 메소드는 파일을 삭제하고 삭제여부를 리턴합니다.
- ✓ `File.exists()` 로 파일의 존재여부를 확인할 수 있습니다.

```
1 public class FileCreateAndDelete {
2
3     public static void main(String[] args) throws IOException {
4
5         File file = new File("newFile.txt");
6
7         boolean isCreated = file.createNewFile();
8         System.out.println("new file is created : " + isCreated);
9         System.out.println("file.exists() : " + file.exists());
10
11         boolean isDeleted = file.delete();
12         System.out.println("file is deleted : " + isDeleted);
13         System.out.println("file.exists() : " + file.exists());
14     }
15 }
```

FileCreateAndDelete.java

코드설명

[Line5] newFile.txt 파일을 나타내는 파일 객체를 생성합니다.
[Line8] 새로운 파일을 생성합니다.
[Line13] 파일을 삭제합니다.

실행결과

```
new file is created : true
file.exists() : true
file is deleted : true
file.exists() : false
```

2.3 디렉토리 생성

- ✓ File 클래스의 mkdir() 과 mkdirs() 메소드는 디렉토리를 생성합니다.
- ✓ File.mkdir() 은 주어진 경로이름으로 디렉토리를 생성하는 메소드입니다.
- ✓ File.mkdirs() 는 부모 디렉토리가 없으면 부모 디렉토리까지 생성합니다.
- ✓ 디렉토리 생성 메소드의 반환 값인 boolean 값은 디렉토리의 생성여부를 나타냅니다.

```
1 public class DirectoryCreation {
2
3     public static void main(String[] args) throws IOException {
4
5         String baseDir = "./";
6
7         File dir1 = new File(baseDir + "dir1");
8         boolean isMade = dir1.mkdir();
9         System.out.println("new directory is made : " + isMade);
10
11        File dir2 = new File(baseDir + "parent/dir2");
12        isMade = dir2.mkdirs();
13        System.out.println("new directory is made : " + isMade);
14    }
15 }
```

DirectoryCreation.java

코드설명

[Line7] dir1 이라는 파일 객체를 생성합니다. File 객체는 파일이나 디렉토리 모듈을 나타낼 수 있습니다.

[Line8] dir1 디렉토리를 생성합니다. 정상적으로 생성되었는지 여부를 isMade 변수에 할당합니다.

[Line11] 계층 구조를 가진 디렉토리를 생성합니다. mkdirs() 메소드는 부모 디렉토리가 없는 경우 부모 디렉토리까지 생성합니다.

실행결과

```
new directory is made : true
new directory is made : true
```

2.4 디렉토리 삭제

- ✓ 디렉토리의 삭제는 앞서 살펴본 `File.delete()` 메소드를 사용합니다.
- ✓ `File.delete()`는 디렉토리가 비어있는 경우만 삭제합니다.

```
1 public class DirectoryDelete {
2
3     public static void main(String[] args) throws IOException {
4
5         String baseDir = "./";
6
7         File dir1 = new File(baseDir + "/dir1");
8         System.out.println("file.exists() : " + dir1.exists());
9         boolean isDeleted = dir1.delete();
10        System.out.println("directory is deleted : " + isDeleted);
11
12        File parent = new File(baseDir + "/parent");
13        System.out.println("file.exists() : " + parent.exists());
14        isDeleted = parent.delete();
15        System.out.println("directory is deleted : " + isDeleted);
16
17        File dir2 = new File(baseDir + "/parent/dir2");
18        System.out.println("file.exists() : " + dir2.exists());
19        isDeleted = dir2.delete();
20        System.out.println("directory is deleted : " + isDeleted);
21    }
22 }
```

DirectoryRemove.java

코드설명

[Line7] dir1 이라는 파일 객체를 생성합니다.
File 객체는 파일이나 디렉토리 모두를 나타낼 수 있습니다.

[Line9] dir1 디렉토리를 삭제합니다.

[Line14] parent 디렉토리를 삭제합니다. 하위에 dir2 디렉토리가 존재하므로 삭제되지 않습니다.

[Line20] parent/dir2 디렉토리를 삭제합니다.

실행결과

```
file.exists() : true
directory is deleted : true
file.exists() : true
directory is deleted : false
file.exists() : true
directory is deleted : true
```

2.5 파일 목록 조회

- ✓ File 클래스의 list() 와 listFiles() 메소드는 디렉토리에 포함된 파일목록을 제공합니다.
- ✓ File.list() 는 디렉토리에 포함된 파일들의 이름을 문자열 배열로 반환하는 메소드입니다.
- ✓ File.listFiles() 는 디렉토리에 포함된 파일 목록을 File 객체 배열로 반환하는 메소드입니다.
- ✓ File 객체의 isDirectory() 메소드는 디렉토리 여부를 반환합니다.

```
1 public class DirectoryCreation {
2
3     public static void main(String[] args) throws IOException {
4
5         File directory = new File("./");
6
7         if (!directory.isDirectory()) { return; }
8
9         File[] files = directory.listFiles();
10        for (int i=0; i<files.length; i++) {
11            System.out.println((file.isFile() ? "[F] " : "[D] ")
12                               + file.getName());
13        }
14    }
```

FileList.java

코드설명

[Line5] 지정한 경로를 나타내는 파일 객체를 생성합니다. (여기서는 현재 디렉토리)
[Line7] 지정한 경로가 디렉토리인지 여부를 체크합니다.
[Line8] 지정한 경로의 모든 파일을 파일객체 배열로 반환합니다.
[Line9~12] 반환 받은 파일객체 배열을 반복 문을 사용하여 화면에 출력합니다. 이때, 디렉토리는 'D', 파일은 'F' 를 붙여줍니다.

실행결과

[F] .classpath
[F] .project
[D] .settings
[D] bin
[D] src

2.6 파일 복사

- ✓ `FileInputStream.read(byte[])` 는 바이트 배열로 부터 데이터를 입력 받아 읽은 바이트 수를 반환합니다.
- ✓ `FileInputStream.read()` 메소드는 더 이상 읽을 바이트가 없으면 `-1`을 반환합니다.
- ✓ `FileOutputStream.write(byte[], int, int)` 는 주어진 바이트 배열의 내용을 파일에 기록합니다.
- ✓ `close()` 메소드는 입출력 스트림을 닫습니다.

```
1 public class FileCopy {
2     public static void main(String[] args) {
3
4         String baseDir = "./";
5         File sourceFile = new File(baseDir + "Source.txt");
6         File targetFile = new File(baseDir + "Target.txt");
7
8         FileInputStream fis = null;
9         FileOutputStream fos = null;
10
11         try {
12             fis = new FileInputStream(sourceFile);
13             fos = new FileOutputStream(targetFile);
14
15             byte[] buf = new byte[256];
16             int readBytes = 0;
17             while((readBytes = fis.read(buf)) != -1) {
18                 fos.write(buf, 0, readBytes);
19             }
20         } catch (IOException e) {
21             e.printStackTrace();
22         } finally {
23             try {
24                 if (fos != null) fos.close();
25                 if (fis != null) fis.close();
26             } catch (Exception e) {}
27         }
28     }
29 }
```

FileCopy.java

코드설명

[Line5] Source.txt 파일을 나타내는 파일 객체를 생성합니다.
[Line6] Target.txt 파일을 나타내는 파일 객체를 생성합니다.
[Line12] Source.txt 파일에 대한 파일 입력 스트림 객체를 생성합니다. 입력 스트림 객체를 통해 파일의 내용을 읽을 수 있습니다.
[Line13] Target.txt 파일에 대한 파일 출력 스트림 객체를 생성합니다. 출력 스트림 객체를 통해 파일에 내용을 저장할 수 있습니다.
[Line15] 256 바이트 크기를 가진 byte 배열을 준비합니다. 이 배열을 버퍼로 사용합니다.
[Line17] 파일 입력 스트림으로 부터 버퍼만큼 데이터를 읽어 들입니다. 읽어 들인 바이트 수는 `readBytes`에 담고, 그 값이 `-1` (EOF, 파일의 끝)이 아닌 경우, 읽는 작업을 계속 반복합니다.
[Line18] 파일 출력 스트림을 통해 파일에 데이터를 기록합니다. 버퍼(`buf`)의 0번째 바이트 부터 `readBytes` 만큼의 데이터를 파일에 기록합니다.
[Line24~25] 파일 입출력 스트림을 닫습니다.

실행결과



복사본이 생성됨

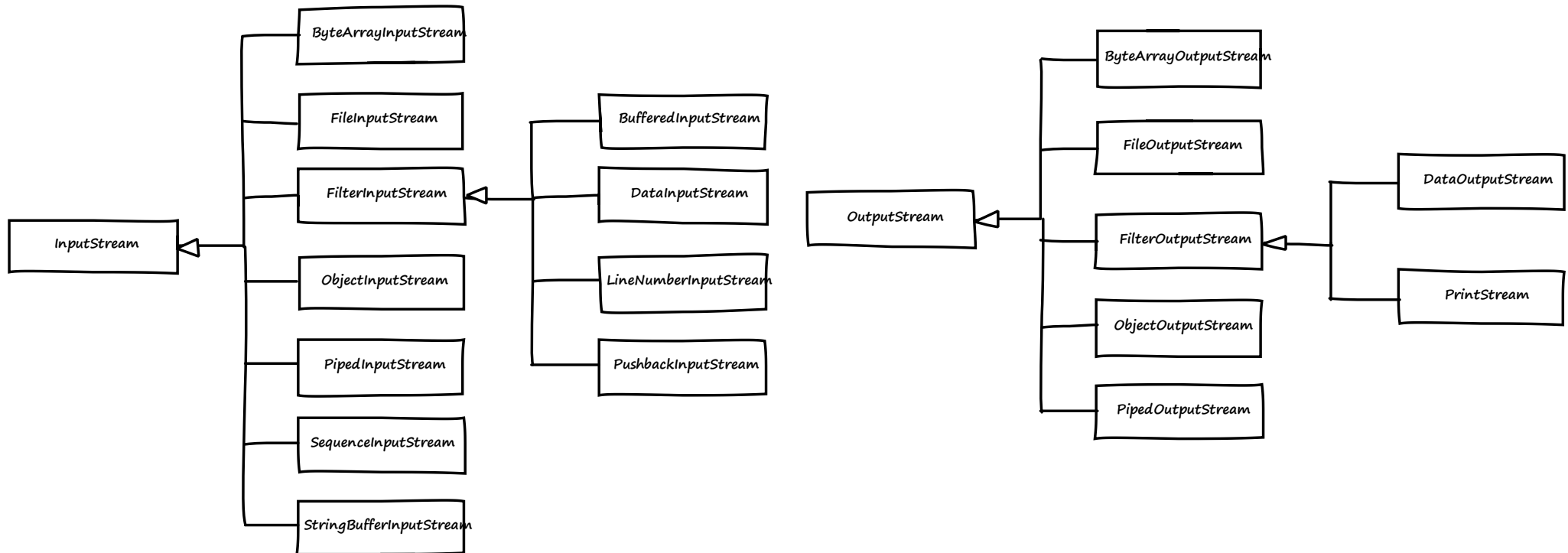


3. Java I/O 이해

- 3.1 바이트 스트림
- 3.2 성적관리 프로그램
- 3.3 문자 스트림
- 3.4 객체 직렬화와 객체 스트림

3.1 바이트 스트림 (1/4) – 개요

- ✓ 바이트 기반 I/O 클래스는 바이트 단위로 입출력을 처리하는 클래스입니다.
- ✓ 파일처리, 네트워크 프로그래밍 또는 데이터베이스와 데이터를 송수신할 때 바이트 스트림을 사용합니다.
- ✓ 바이트 단위 I/O 클래스에는 파일, 바이트 배열, 객체 등을 처리할 수 있는 입출력 스트림 클래스가 있습니다.
- ✓ 그리고, 기존의 입출력 스트림 클래스에 추가적인 기능을 추가할 수 있는 필터 스트림 클래스를 제공합니다.



3.1 바이트 스트림 (2/4) – 추상 클래스

- ✓ InputStream 과 OutputStream 클래스는 바이트 단위로 입출력을 처리하는 최상위 추상 클래스입니다.
- ✓ InputStream 에는 입력을 위한 read() 메소드와, 읽을 수 있는 바이트 수를 리턴하는 available() 메소드가 있습니다.
- ✓ OutputStream 에는 출력을 위한 write() 메소드와, 버퍼에 남은 데이터를 출력하는 flush() 메소드가 있습니다.
- ✓ 스트림을 통한 입출력이 끝난 후에는 close() 메소드를 호출하여 스트림을 종료합니다.

InputStream 주요 메소드	설명
int available()	현재 읽을 수 있는 바이트 수를 반환합니다.
int read()	입력 스트림에서 한 바이트를 읽어 int 값으로 반환합니다. 더 이상 읽을 내용이 없을 경우, -1을 반환합니다.
int read(byte buf[])	입력 스트림에서 buf[] 크기만큼을 읽어 buf에 저장하고 읽은 바이트 수를 반환합니다. 더 이상 읽을 내용이 없을 경우, -1을 반환합니다.
int skip(long numBytes)	numBytes로 지정된 바이트를 무시하고 무시된 바이트 수를 반환합니다.

OutputStream 주요 메소드	설명
void flush()	버퍼에 남은 출력 스트림을 출력합니다.
void write(int i)	정수 i의 하위 8비트를 출력합니다.
void write(byte buf[])	buf 배열의 내용을 출력합니다.
void write(byte buf[], int index, int size)	buf 배열의 index 위치부터 size만큼의 바이트를 출력합니다.

3.1 바이트 스트림 (3/4) – 파일 입출력

- ✓ FileInputStream과 FileOutputStream 클래스는 InputStream 또는 OutputStream을 상속합니다.
- ✓ FileInputStream은 파일을 바이트 단위로 읽고, FileOutputStream은 파일에 바이트 단위로 기록합니다.
- ✓ FileInputStream 또는 FileOutputStream은 File 객체 또는 파일의 이름으로 입출력 스트림 객체를 생성합니다.
- ✓ FileOutputStream을 생성할 때, append 매개변수의 값을 true로 설정하면 기존 파일에 이어서 기록합니다.



FileInputStream 생성자	설명
FileInputStream(String name)	name에 해당하는 파일로부터 바이트 단위로 읽어 들이는 스트림 객체를 생성합니다.
FileInputStream(File file)	file 객체로 지정한 파일로부터 바이트 단위로 읽어 들이는 스트림 객체를 생성합니다.

FileOutputStream 생성자	설명
FileOutputStream(String name)	name에 해당하는 파일에 대한 출력 스트림을 생성합니다.
FileOutputStream(String name, boolean append)	지정한 파일로 출력 스트림을 생성합니다. Append 변수 값이 true로 설정되면 기존 파일에 이어서 쓰게 됩니다.
FileOutputStream(File file)	File 객체로 지정된 파일에 대한 출력 스트림을 생성합니다.

3.1 바이트 스트림 (4/4) – 기본 데이터형 입출력

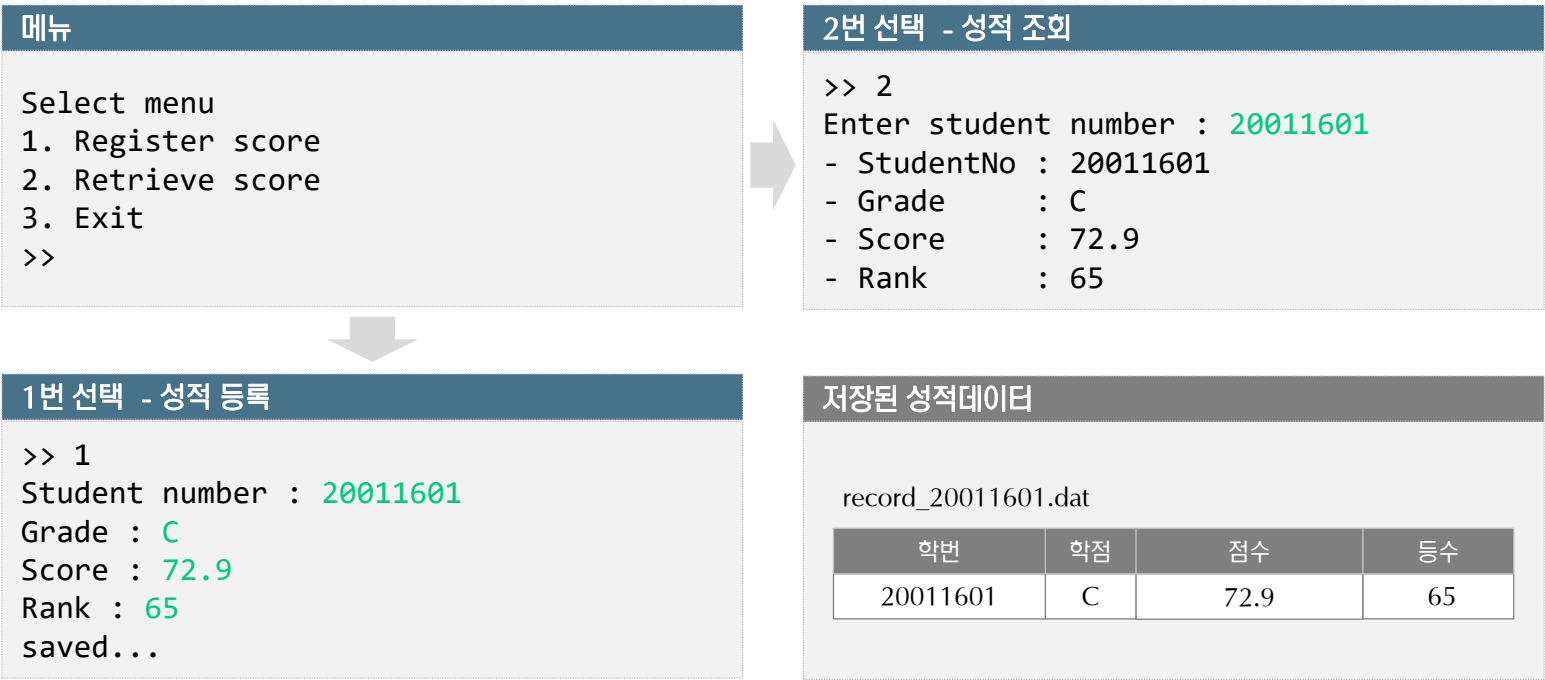
- ✓ DataInputStream과 DataOutputStream 클래스는 기본 데이터 형에 대한 입력과 출력을 지원합니다.
- ✓ Data*Stream 은 필터 스트림 클래스로 입출력 스트림에 기본 데이터형 처리 기능을 추가한 클래스 입니다.
- ✓ DataInputStream 은 입력 스트림으로 부터 기본 데이터 형 값을 읽을 수 있는 클래스 입니다.
- ✓ DataOutputStream 은 기본 데이터형 값을 출력 스트림에 기록하는 클래스 입니다.



DataInputStream 메소드	설명	DataOutputStream 메소드	설명
boolean readBoolean()	스트림에서 boolean 값을 읽습니다.	void writeBoolean()	boolean 값을 스트림에 기록합니다.
byte readByte()	스트림에서 byte 값을 읽습니다.	void writeByte()	byte 값을 스트림에 기록합니다.
char readChar()	스트림에서 char 값을 읽습니다.	void writeChar()	char 값을 스트림에 기록합니다.
double readDouble()	스트림에서 double 값을 읽습니다.	void writeDouble()	double 값을 스트림에 기록합니다.
float readFloat()	스트림에서 float 값을 읽습니다.	void writeFloat()	float 값을 스트림에 기록합니다.
long readLong()	스트림에서 long 값을 읽습니다.	void writeLong()	long 값을 스트림에 기록합니다.
short readShort()	스트림에서 short 값을 읽습니다.	void writeShort()	short 값을 스트림에 기록합니다.
int readInt()	스트림에서 int 값을 읽습니다.	void writeInt()	int 값을 스트림에 기록합니다.
int readUTF()	스트림에서 UTF-8 문자열을 읽습니다.	void writeUTF()	UTF-8 문자열을 스트림에 기록합니다.

3.2 성적관리 프로그램 (1/4) – 실습개요

- ✓ 학생들의 성적을 관리하는 프로그램을 바이트 단위 I/O 클래스를 사용하여 구현합니다.
- ✓ 성적관리 프로그램은 학생들의 성적을 등록하는 기능과, 조회하는 기능을 제공합니다.
- ✓ 학생들의 성적 데이터는 record_{학번}.dat 라는 파일로 저장되며, 학번, 학점, 점수, 등수가 포함됩니다.
- ✓ 학번은 8자리로 int형, 학점은 하나의 문자로 char형, 점수는 실수형 float, 등수는 short 형을 사용합니다.



3.2 성적관리 프로그램 (2/4) – 사용자 메뉴

- ✓ 프로그램이 실행되면 사용자 메뉴가 출력됩니다. 사용자는 표준 입력장치를 통해 사용할 기능을 선택합니다.
- ✓ 1을 선택하면, 성적을 입력 받습니다. 입력 항목은 학번, 학점(등급), 점수 그리고 등수입니다.
- ✓ 2를 선택하면, 학번을 입력 받아 저장된 성적 데이터를 출력합니다. 단, 데이터가 없으면 메시지를 보여줍니다.
- ✓ 3을 선택하면, 프로그램을 종료합니다. 입력한 번호가 없는 메뉴거나, 다른 작업이 완료되면 다시 메뉴를 보여줍니다.

```
1 public class ScoreManager {
2     private Scanner scanner;
3     private void launch() throws IOException {
4         this.scanner = new Scanner(System.in);
5         while (true) {
6             System.out.println("Select menu");
7             System.out.println("1. Register score");
8             System.out.println("2. Retrieve score");
9             System.out.println("3. Exit");
10            System.out.print(">> ");
11            String selectNum = scanner.nextLine();
12            if ("3".equals(selectNum)) {
13                break;
14            } else {
15                if ("1".equals(selectNum)) {
16                    doRegister();
17                } else if ("2".equals(selectNum)) {
18                    doRetrieve();
19                } else {
20                    System.out.println("Unknown menu. Please retry.");
21                }
22            }
23        }
24    }
}
```

ScoreManager.java

코드설명

[Line4] 표준 입력장치로 부터 데이터를 입력 받는 Scanner 객체를 생성합니다.

[Line11] 사용자로 부터 메뉴를 입력 받습니다.

[Line12] 3을 선택하면 프로그램을 종료합니다.

[Line15] 1을 선택하면 성적을 입력하는 메소드를 호출합니다.

[Line17] 2를 선택하면 성적을 조회하는 메소드를 호출합니다.

[Line19] 알 수 없는 메뉴번호를 입력한 경우 다시 메뉴를 보여줍니다.

실행결과

```
Select menu
1. Register score
2. Retrieve score
3. Exit
>>
```


3.2 성적관리 프로그램 (3/4) – 성적 등록

- ✓ doRegister() 메소드는 학생의 성적 데이터를 입력 받아 파일로 저장하는 메소드입니다.
- ✓ 사용자는 프로그램의 안내에 따라 성적 데이터(학번, 학점(등급), 점수, 등수)를 입력합니다.
- ✓ 성적 데이터는 FileOutputStream을 사용하여 record_{학번}.dat 파일로 저장합니다.
- ✓ 데이터를 기본 데이터형으로 처리하기 위하여 필터 스트림 클래스인 DataOutputStream 객체를 사용합니다.

```
1 private void doRegister() throws IOException {
2     System.out.print("Student number : ");
3     String studentNo = scanner.nextLine();
4
5     System.out.print("Grade : ");
6     String grade = scanner.nextLine();
7
8     System.out.print("Score : ");
9     String score = scanner.nextLine();
10
11    System.out.print("Rank : ");
12    String rank = scanner.nextLine();
13
14    File dataFile = new File("record_" + studentNo + ".dat");
15    DataOutputStream dos = new DataOutputStream(new FileOutputStream(dataFile));
16
17    dos.writeInt(Integer.parseInt(studentNo));
18    dos.writeChar(grade.charAt(0));
19    dos.writeFloat(Float.parseFloat(score));
20    dos.writeShort(Short.parseShort(rank));
21
22    System.out.println("saved...");
23    dos.close();
24 }
```

ScoreManager.java

코드설명

[Line3] 사용자로부터 학번을 입력 받습니다.
[Line6] 사용자로부터 학점을 입력 받습니다.
[Line9] 사용자로부터 점수를 입력 받습니다.
[Line12] 사용자로부터 등수를 입력 받습니다.
[Line14] 성적데이터를 저장하기 위한 파일 객체를 생성합니다. (파일명 : record_{학번}.dat)
[Line15] 성적데이터 파일에 저장하는 출력 스트림 객체를 생성합니다. 기본 데이터형으로 스트림에 출력하기 위하여 DataOutputStream 객체를 생성합니다.
[Line17~20] 데이터 출력 스트림 객체를 사용하여 출력 스트림에 기본 데이터형 값을 기록합니다.
[Line23] 출력 스트림을 종료합니다.

실행결과

```
>> 1
Student number : 20011601
Grade : C
Score : 72.9
Rank : 65
saved...
```

3.2 성적관리 프로그램 (4/4) – 성적 조회

- ✓ doRetrieve() 메소드는 학번을 입력 받아, 학생의 성적 데이터를 화면에 출력하는 메소드입니다.
- ✓ 사용자는 프로그램의 안내에 따라 표준 입력장치로 학번을 입력하여 성적 데이터를 조회합니다.
- ✓ File 객체의 exists() 메소드를 사용하여 학번에 해당하는 성적 데이터가 존재여부를 체크합니다.
- ✓ FileInputStream으로 파일 스트림을 읽고, DataInputStream 객체를 통해 기본 데이터형으로 조회합니다.

```
1 private void doRetrieve() throws IOException {
2     System.out.print("Enter student number : ");
3     String number = scanner.nextLine();
4
5     File dataFile = new File("record_" + number + ".dat");
6     if (!dataFile.exists()) {
7         System.out.println("Data file not found.");
8         return;
9     }
10
11     DataInputStream dis = new DataInputStream(new FileInputStream(dataFile));
12
13     int studentNo = dis.readInt();
14     char grade = dis.readChar();
15     float score = dis.readFloat();
16     short rank = dis.readShort();
17
18     System.out.println("- StudentNo : " + studentNo);
19     System.out.println("- Grade : " + grade);
20     System.out.println("- Score : " + score);
21     System.out.println("- Rank : " + rank);
22
23     dis.close();
24 }
```

ScoreManager.java

코드설명

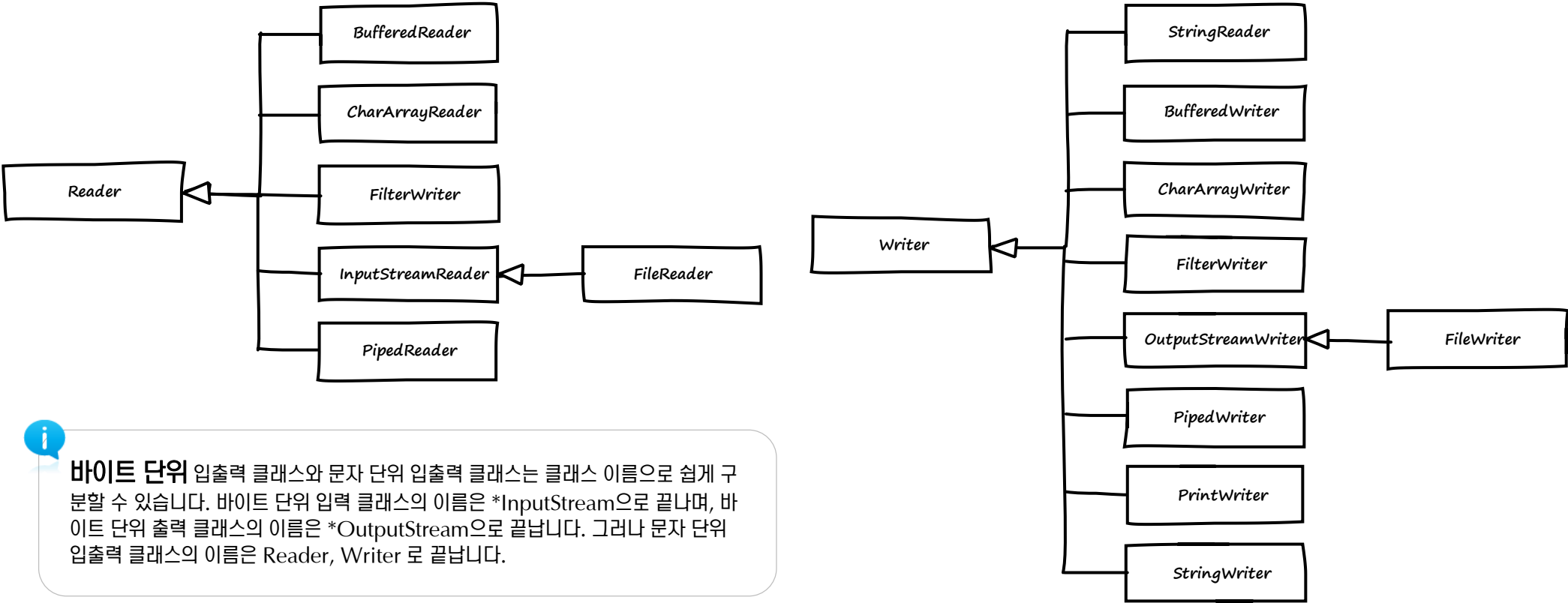
[Line3] 성적을 조회할 학번을 입력 받습니다.
[Line5] 학번에 해당하는 성적 데이터 파일을 나타내는 파일 객체를 생성합니다.
[Line6~9] 성적 데이터 파일이 존재하지 않는 경우, 메시지를 출력하고 기능을 종료합니다.
[Line11] 성적 데이터 파일의 내용을 스트림으로 읽기 위하여 FileInputStream 클래스를 생성합니다. 스트림의 내용을 기본 데이터형으로 읽기 위하여 DataInputStream 객체를 생성합니다.
[Line13~16] 데이터 입력 스트림 객체를 통해 스트림의 내용을 기본 데이터형으로 읽습니다.
[Line23] 입력 스트림을 종료합니다.

실행결과

```
>> 2
Enter student number : 20011601
- StudentNo : 20011601
- Grade : C
- Score : 72.9
- Rank : 65
```

3.3 문자 스트림 (1/6) – 개요

- ✓ Java는 문자를 유니코드로 처리합니다. 2바이트 이상의 유니코드 문자를 바이트 기반 I/O로 처리하는 것은 불편합니다.
- ✓ Reader, Writer와 같은 문자 기반 입출력 클래스를 사용하면, 바이트 대신 문자 단위로 입출력을 처리할 수 있습니다.
- ✓ 모든 문자 기반 입출력 클래스는 추상 클래스인 Reader와 Writer 클래스를 상속합니다.
- ✓ InputStreamReader, OutputStreamWriter 클래스는 입출력 스트림 객체를 문자 단위 입출력 객체로 변환합니다.



3.3 문자 스트림 (2/6) – 추상클래스

- ✓ Stream 계열 클래스가 바이트 단위로 처리하는 반면, Reader, Writer 추상클래스는 문자 단위로 입출력 합니다.
- ✓ Reader의 read() 메소드는 문자 단위로 데이터를 읽습니다. 이때, 읽을 데이터가 없으면 차단됩니다.
- ✓ Reader는 차단(blocking) 없이 읽을 수 있는 데이터가 있는지만 확인하는 ready() 메소드를 제공합니다.
- ✓ Writer는 문자 단위로 데이터를 쓰는 write() 메소드와, 체인형식으로 쓸 수 있는 append() 메소드를 제공합니다.

Reader 주요 메소드	설명
boolean ready()	스트림에 읽을 수 있는 데이터가 있는지 여부를 반환합니다.
int read()	입력 스트림에서 하나의 문자를 읽어 int 형으로 반환합니다. 더 이상 읽을 내용이 없을 경우, -1을 반환합니다.
int read(char cbuf[])	입력 스트림에서 cbuf[] 크기만큼 읽어 cbuf에 저장하고 읽은 문자 수를 반환합니다. 더 이상 읽을 내용이 없을 경우, -1을 반환합니다.
long skip(long n)	n으로 지정된 문자만큼을 무시하고, 실제로 무시된 문자의 수를 반환합니다.

Writer 주요 메소드	설명
void flush()	버퍼에 남은 출력 스트림을 출력합니다.
void write(int c)	하나의 문자를 출력합니다. (int 형의 하위 16비트만 사용합니다.)
void write(char cbuf[])	cbuf 배열의 내용을 출력합니다.
void write(char cbuf[], int off, int len)	cbuf 배열의 off 위치부터 len 만큼의 문자를 출력합니다.
Writer append(CharSequence csq)	csq 에 지정된 문자열을 출력합니다. (내부적으로는 write() 메소드 호출과 동일함) 출력이 끝나면 Writer 객체를 다시 리턴하여, 메소드 체인을 구성할 수 있게 합니다.

3.3 문자 스트림 (3/6) – 바이트 스트림 변환

- ✓ InputStreamReader와 OutputStreamWriter는 바이트 스트림 객체를 문자 스트림 객체로 변환하는 클래스입니다.
- ✓ InputStreamReader 생성자를 통해, 바이트 스트림을 문자로 읽어들이기 때 사용할 문자 인코딩을 설정합니다.
- ✓ OutputStreamWriter 생성자를 통해, 문자를 바이트 스트림으로 출력할 때 사용할 문자 인코딩을 설정합니다.
- ✓ 문자 인코딩을 설정하지 않으면, 시스템의 디폴트 문자 인코딩을 사용합니다.



InputStreamReader 생성자	설명
InputStreamReader(InputStream in)	in 객체를 받아 Reader 객체를 생성합니다.
InputStreamReader(InputStream in, String charsetName)	in 객체와 charsetName 을 받아, 해당 문자 인코딩을 사용하는 Reader 객체를 생성합니다. 문자 인코딩 설정은 인코딩 이름을 나타내는 문자열이나 Charset, CharsetDecoder 객체를 사용할 수 있습니다.

OutputStreamWriter 생성자	설명
OutputStreamWriter(OutputStream out)	out 객체를 받아 Writer 객체를 생성합니다.
OutputStreamWriter (OutputStream out, String charsetName)	out 객체와 charsetName 을 받아, 해당 문자 인코딩을 사용하는 Reader 객체를 생성합니다. 문자 인코딩 설정은 인코딩 이름을 나타내는 문자열이나 Charset, CharsetEncoder 객체를 사용할 수 있습니다.

3.3 문자 스트림 (4/6) – File Reader & Writer

- ✓ FileReader 클래스는 파일을 문자 단위로 읽는 클래스로 InputStreamReader 클래스를 상속합니다.
- ✓ FileWriter 클래스는 파일에 문자단위로 쓰는 클래스로 OutputStreamWriter 클래스를 상속합니다.
- ✓ 파일 Reader & Writer는 내부적으로는 파일 입출력 스트림을 Reader, Writer로 변환하여 처리합니다.
- ✓ 파일 Reader & Writer는 문자로만 구성된 텍스트 파일을 읽거나 저장하기에 적합합니다.



```
1 public class TextPrinter {
2     public static void main(String[] args) throws IOException {
3
4         System.out.print("Enter text filename : ");
5         String filename = new Scanner(System.in).nextLine();
6
7         FileReader reader = new FileReader(filename);
8         int ch = 0;
9         while ((ch = reader.read()) != -1) {
10             System.out.print((char) ch);
11         }
12
13         reader.close();
14     }
15 }
```

TextPrinter.java

코드설명

[Line5] 텍스트 파일명을 입력 받습니다.
[Line7] 파일에 대한 FileReader 객체를 생성합니다.
[Line9~11] 한 글자씩 읽습니다. 읽은 값이 EOF(-1)이 아닐 때 까지 반복하면서 읽은 값을 문자형으로 변환하여 화면에 출력합니다.
[Line13] 입력 스트림을 닫습니다.

실행결과

Enter text filename : poem.txt
서시 - 윤동주
죽는 날까지 하늘을 우러러
한점 부끄럼이 없기를
잎새에 이는 바람에도
나는 괴로워했다.
...

3.3 문자 스트림 (5/6) – Buffered Reader & Writer

- ✓ 입력과 출력 작업이 빠른 속도로 일어날 때, 속도차이가 발생하면 병목현상이 발생합니다.
- ✓ BufferedReader와 BufferedWriter는 입출력 시 병목현상을 줄이기 위해서 버퍼를 사용합니다.
- ✓ 네트워크 프로그래밍의 경우 write() 호출 후, flush() 를 호출하여 버퍼에 남은 데이터를 전송해야 합니다.
- ✓ flush() 는 버퍼에 남은 내용을 모두 출력한 후 버퍼를 비우는 메소드입니다.



```
1 public class BufferedTextPrinter {
2     public static void main(String[] args) throws IOException {
3
4         System.out.print("Enter text filename : ");
5         String filename = new Scanner(System.in).nextLine();
6
7         BufferedReader reader = new BufferedReader(new FileReader(filename));
8         String readLine = null;
9         while ((readLine = reader.readLine()) != null) {
10             System.out.println(readLine);
11         }
12
13         reader.close();
14     }
15 }
```

BufferedTextPrinter.java

코드설명

[Line5] 텍스트 파일명을 입력 받습니다.

[Line7] 파일에 대한 FileReader 객체를 생성합니다. 생성한 Reader 객체를 사용하여 BufferedReader 객체를 생성합니다.

[Line9~11] 한 라인씩 읽습니다. 읽은 문자열이 null (EOF)이 아닐 때 까지 반복하면서 읽은 문자열을 화면에 출력합니다.

[Line13] 입력 스트림을 닫습니다.

실행결과

Enter text filename : poem.txt

서시 - 윤동주

죽는 날까지 하늘을 우러러

한점 부끄럼이 없기를

앞새에 이는 바람에도

나는 괴로워했다.

...

3.3 문자 스트림 (6/6) – PrintWriter

- ✓ PrintWriter 클래스는 객체의 내용을 텍스트로 출력하는데 특화된 클래스입니다.
- ✓ PrintStream 클래스에서 제공하는 대부분의 출력 관련 메소드를 유사하게 제공합니다.
- ✓ 이 클래스의 모든 메소드는 예외를 발생시키지 않습니다. 예외발생 여부는 checkError() 메소드로 확인합니다.
- ✓ format()이나 printf() 메소드를 사용하면 다양한 형식으로 객체의 내용을 출력할 수 있습니다.

```
1 public class GugudanPrinter {
2
3     public static void main(String[] args) throws FileNotFoundException {
4
5         System.out.print("Enter number : ");
6         int dan = new Scanner(System.in).nextInt();
7
8         File output = new File("gugudan_" + dan + ".txt");
9         PrintWriter writer = new PrintWriter(output);
10
11         for ( int i = 1; i <= 9; i++ ) {
12             writer.printf("%d * %d = %d\n", dan, i, (dan * i));
13         }
14
15         System.out.println("gugudan file saved.");
16         writer.close();
17     }
18 }
```

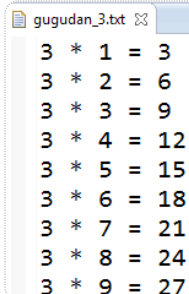
GugudanPrinter.java

코드설명

[Line6] 출력할 단을 입력합니다.
[Line8] 구구단을 출력할 파일 객체를 생성합니다.
[Line9] 파일에 텍스트로 출력하는 PrintWriter 객체를 생성합니다.
[Line11~13] 구구단 형식에 맞게 출력합니다.
[Line16] 출력 스트림을 닫습니다.

실행결과

Enter number : 3
gugudan file saved.



```
gugudan_3.txt
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
```

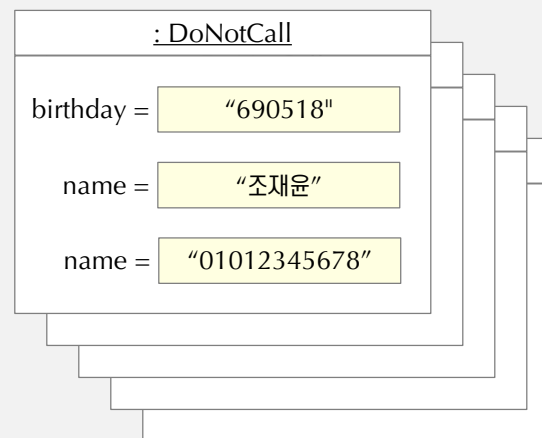
퀴즈~

✓ DNC 파일에서 데이터 읽기

등록된 DNC 목록

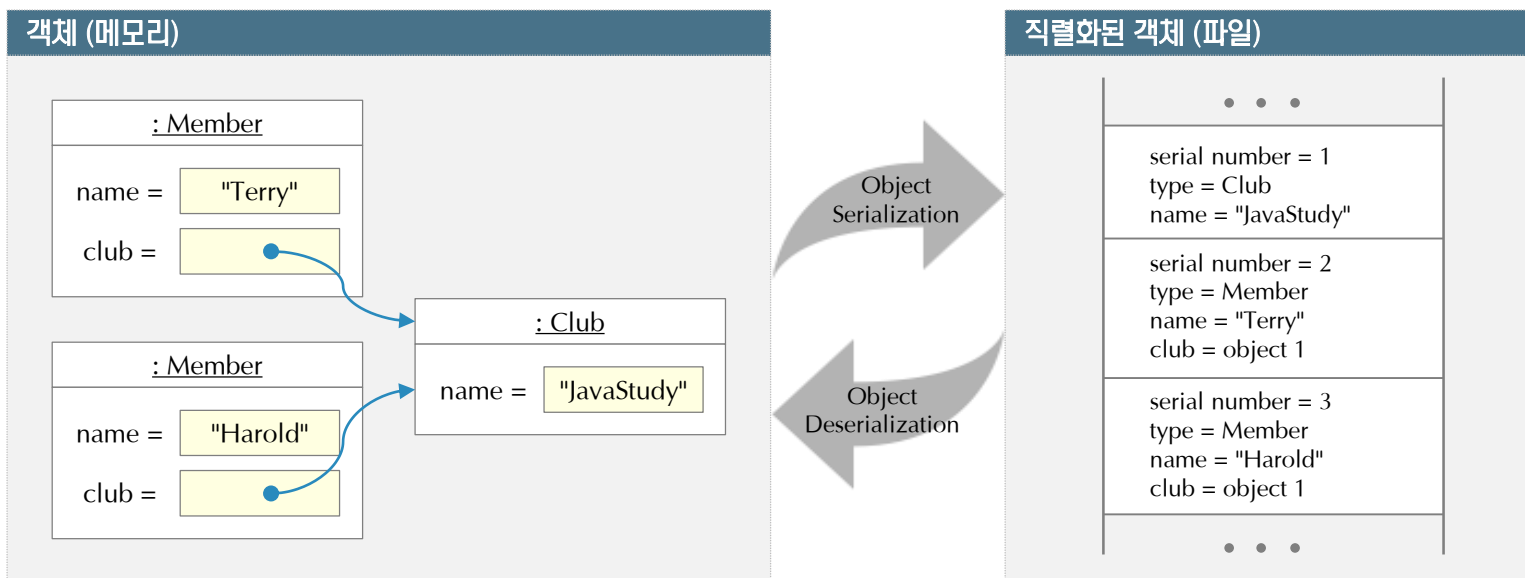
690518|조재윤|01012345678
901004| |0162931234
870910|최귀화|01094834823
710123|임형준|024857342
900918|진선규|01058371295
581001|홍기준|024857342
871010|최귀화|01094834823
491009|허동원|01098864834
710123|진선규|01058371295
710123| |0319572386
581001|홍기준|024857342
710123|민경진|01029475593
710123|이동윤|01057603452

객체 (메모리)



3.4 객체 직렬화와 객체 스트림 (1/3)

- ✓ Java에서는 객체를 스트림 즉, 일련의 바이트들로 변환할 수 있는 객체 직렬화(Serialization)를 제공합니다.
- ✓ 객체를 직렬화하는 과정을 마샬링이라 하고, 역직렬화하여 객체로 변환하는 것을 언마샬링이라 합니다.
- ✓ 직렬화된 객체는 파일이나 네트워크로 전송할 수 있고, 전송된 데이터는 역직렬화를 통해 다시 객체로 변환됩니다.
- ✓ `ObjectInputStream`과 `ObjectOutputStream` 클래스는 객체 직렬화 및 역직렬화를 위한 메소드를 제공합니다.



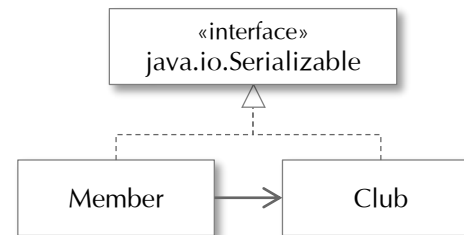
3.4 객체 직렬화와 객체 스트림 (2/3)

- ✓ 객체 직렬화는 기본 자료형이나 java.io.Serializable 인터페이스를 구현한 객체만 가능합니다.
- ✓ Serializable 인터페이스는 구현할 메소드가 없는 직렬화 대상을 나타내는 Mark 인터페이스입니다.
- ✓ ObjectInputStream의 readObject() 메소드는 입력 스트림으로부터 객체를 조회합니다.
- ✓ ObjectOutputStream의 writeObject() 메소드는 출력 스트림으로 객체를 직렬화하여 출력합니다.

```
1 Club club = new Club("JavaStudy");
2 Member member = new Member("Harold", club);
3
4 String filename = "object.out";
5
6 // save object to file
7 ObjectOutputStream oos =
8     new ObjectOutputStream(new FileOutputStream(filename));
9 oos.writeObject(member);
10
11 oos.close();
12
13 // read object from file
14 ObjectInputStream ois =
15     new ObjectInputStream(new FileInputStream(filename));
16 Member savedMember = (Member) ois.readObject();
17
18 System.out.println("member name : " + savedMember.getName());
19 System.out.println("club name : " + savedMember.getClub().getName());
20
21 ois.close();
```

ObjectInOut.java

객체 모델



코드설명

[Line1~2] 직렬화할 객체를 생성합니다.
[Line7~8] 객체를 파일로 출력하는 ObjectOutputStream 객체를 생성합니다.
[Line9] 객체를 파일로 저장합니다.
[Line11] 객체 출력 스트림을 닫습니다.
[Line14~15] 객체를 스트림으로 부터 조회하는 ObjectInputStream 객체를 생성합니다.
[Line16] 객체를 스트림에서 조회합니다.
[Line21] 객체 입력 스트림을 닫습니다.

실행결과

member name : Harold
club name : JavaStudy

3.4 객체 직렬화와 객체 스트림 (3/3)

- ✓ 복잡한 객체 그래프를 DB에 저장해야 하는 경우 객체 직렬화를 유용하게 사용할 수 있습니다.
- ✓ 직렬화하여 저장한 객체가 변경된 후 역 직렬화했을 때 변경 내용에 대한 추가 처리가 필요합니다.

