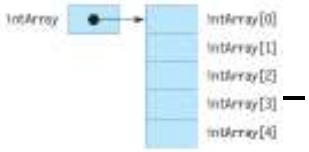
기초문법
배열, 문자열, 입출력

514770-1
2017년 봄학기
3/22/2017
박경신

Array



- 배열(array)
  - 여러 개의 데이터를 같은 이름으로 활용할 수 있도록 해주는 자료 구조
    - 인덱스(Index, 순서 번호)와 인덱스에 대응하는 데이터들로 이루어진 자료 구조
    - 배열을 이용하면 한 번에 많은 메모리 공간 선언 가능
  - 배열은 같은 타입의 데이터들이 순차적으로 저장되는 공간
    - 원소 데이터들이 순차적으로 저장됨
    - 인덱스를 이용하여 원소 데이터 접근
    - 반복문을 이용하여 처리하기에 적합한 자료 구조(주로 for 또는 for-each 반복문과 많이 사용됨)
  - 배열 인덱스
    - 0부터 시작
    - 인덱스는 배열의 시작 위치에서부터 데이터가 있는 상대 위치

Array

- 자바 배열의 필요성과 모양

Array

- 배열 선언과 배열 생성의 두 단계 필요
  - 배열 선언

```
int     intArray[];
char    charArray[];
```
또는
```
int[]   intArray;
char[]  charArray;
```

  - 배열 생성

```
intArray = new int[10];
charArray = new char[20];
```
또는
```
int intArray[] = new int[10];
char charArray[] = new char[20];
```

  - 선언과 초기화
    - 배열 생성과 값 초기화

```
// 총 10개의 정수 배열 생성 및 값 초기화
int intArray[] = {0,1,2,3,4,5,6,7,8,9};
```

  - 잘못된 배열 선언

```
//int intArray[10]; // 컴파일 오류. 배열의 크기를 지정할 수 없음
```

Array

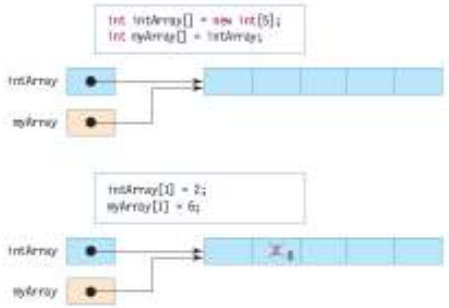
- 배열 선언과 생성

Array

- 배열을 초기화하면서 생성한 결과

Array

- 배열 참조
  - 생성된 1개의 배열을 다수의 레퍼런스가 참조 가능



Array

- 배열 원소 접근
  - 반드시 배열 생성 후 접근

```
int intArray []; // 배열 선언
intArray[4] = 8; // 오류, intArray 배열의 메모리가 할당되지 않았음
```

  - 배열 변수명과 [] 사이에 원소의 인덱스를 적어 접근
    - 배열의 인덱스는 0부터 시작
    - 배열의 마지막 항목의 인덱스는 (배열 크기 – 1)

```
int[] intArray;
intArray = new int[10];

intArray[3]=6; // 배열에 값을 저장
int n = intArray[3]; // 배열로부터 값을 읽음
```

Array

- 배열 인덱스
  - 인덱스는 0부터 시작하며 마지막 인덱스는 (배열 크기 -1)
  - 인덱스는 정수 타입만 가능

```
int intArray [] = new int[5]; // 인덱스는 0~4까지 가능
int n = intArray[-2];  // 실행 오류. -2는 인덱스로 적합하지 않음
int m = intArray[5];  // 실행 오류. 5는 인덱스의 범위(0~4)를 넘었음
```

- 배열의 크기
  - 배열의 크기는 배열 레퍼런스 변수를 선언할 때 결정되지 않음
    - 배열의 크기는 배열 생성 시에 결정되며, 나중에 바꿀 수 없음
  - 배열의 크기는 배열의 length 필드에 저장

```
int size = intArray.length;
```

Array & For-each

- For-each 문
  - 배열이나 나열(enumeration)의 각 원소를 순차적으로 접근하는데 유용한 for 문

```
int[] num = { 1,2,3,4,5 };
int sum = 0;
// 반복될 때마다 k는 num[0], num[1], ..., num[4] 값으로 설정
for (int k : num)
  sum += k;
System.out.println("합은 " + sum);
```

합은 15

```
String names[] = { "사과", "배", "바나나", "체리", "딸기", "포도" } ;
// 반복할 때마다 s는 names[0], names[1], ..., names[5] 로 설정
for (String s : names)
  System.out.print(s + " ");
```

사과 배 바나나 체리 딸기 포도

메소드에서 배열 리턴

- 메소드의 배열 리턴
  - 배열의 레퍼런스만 리턴
- 메소드의 리턴 타입
  - 메소드가 리턴하는 배열의 타입은 리턴 받는 배열 타입과 일치
  - 리턴 타입에 배열의 크기를 지정하지 않음

Return type

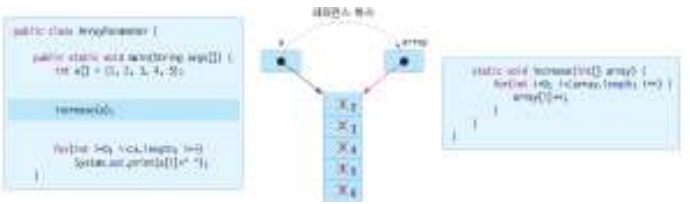
Method name

```
int[] makeArray() {
  int temp[] = new int[4];
  return temp;
}
```

Array return

매개 변수에 배열이 전달되는 경우

- 매개변수에 배열이 전달되는 경우는 배열의 reference가 복사



23456

Parameter Passing – Primitive Type

- 자바의 인자 전달 방식(Parameter Passing)
  - **값에 의한 호출(Pass-by-value)**
  - 기본 타입(Primitive Type: 예를 들어 int, double,)의 값을 전달하는 경우
    - 값이 복사되어 전달
    - 메소드의 매개 변수가 변경되어도 호출한 실제 인자 값은 변경되지 않음

```
void sum(int m, int n) {
    return m + n;
}

int a = 5;
int b = 7;
int c = sum(a, b);
System.out.println(c);
```

m 5
n 7
a 5
b 7

Parameter Passing – Reference Type

- 자바의 인자 전달(Parameter Passing) 방식
  - 객체(class object) 혹은 배열(array)을 전달하는 경우
    - 객체나 배열의 레퍼런스만 전달**(Pass-by-reference)**
      - 객체 혹은 배열이 통째로 복사되어 전달되는 것이 아님
    - 메소드의 매개 변수와 호출한 실인자가 객체나 배열을 공유하게 됨

```
void assignArray(int[] values) {
    for (int i=0; i<values.length; i++) {
        values[i] = i; // values 배열에 0~9까지 값을 넣어줌
    }
}

int[] intArray = new int[5]; // create 5 integer array
assignArray(intArray);
for (int v : intArray) {
    System.out.println(v);
}
```

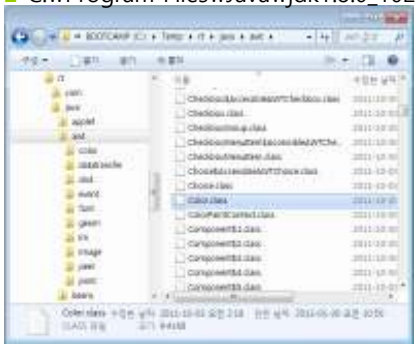
values
intArray
intArray[0]
intArray[1]
intArray[2]
intArray[3]
intArray[4]

패키지의 특징

- 패키지의 특징
  - 패키지 계층구조
    - 클래스나 인터페이스가 너무 많아지면 관리의 어려움
    - 관련된 클래스 파일을 하나의 패키지로 계층화하여 관리 용이
  - 패키지별 접근 제한
    - default로 선언된 클래스나 멤버는 동일 패키지 내의 클래스들이 자유롭게 접근하도록 허용
  - 동일한 이름의 클래스와 인터페이스의 사용 가능
    - 서로 다른 패키지에 이름이 같은 클래스와 인터페이스 존재 가능
  - 높은 소프트웨어 재사용성
    - 오라클에서 제공하는 자바 API는 패키지로 구성되어 있음
    - java.lang, java.io 등의 패키지들 덕분에 일일이 코딩하지 않고 입출력 프로그램을 간단히 작성할 수 있음

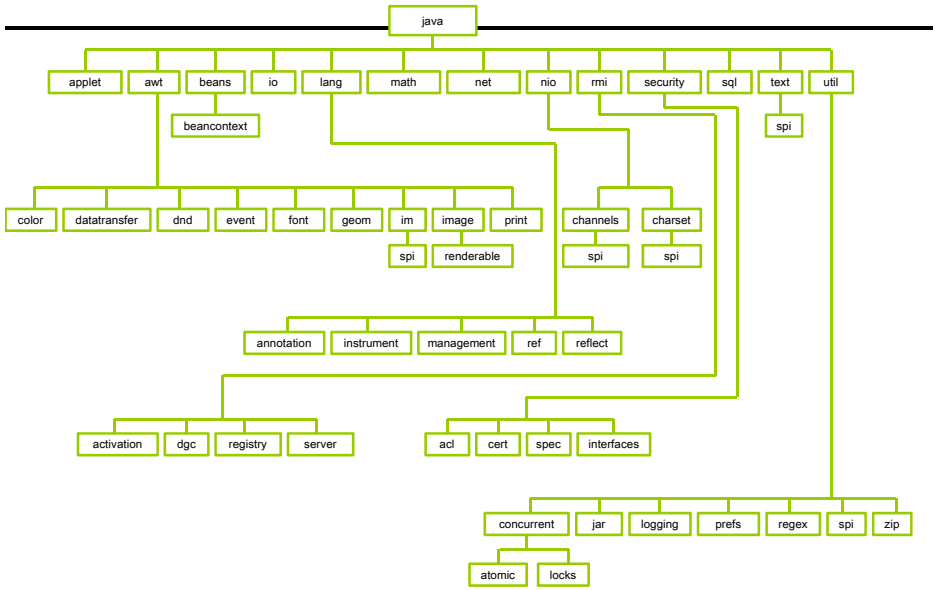
자바 JDK의 패키지 구조

- JDK 패키지
  - 자바에서는 관련된 클래스들을 표준 패키지로 묶어 사용자에게 제공
  - 자바에서 제공하는 패키지는 C언어의 표준 라이브러리와 유사
  - JDK의 표준 패키지는 rt.jar에 담겨 있음
    - C:₩Program Files₩Java₩jdk1.8.0_102₩jre₩lib₩rt.jar



rt.jar의 java.awt 패키지에 컴파일된 클래스들이 들어있다.

자바 패키지 구조



주요 패키지

- java.lang
  - 자바 language 패키지
    - 스트링, 수학 함수, 입출력 등 자바 프로그래밍에 필요한 기본적인 클래스와 인터페이스
  - 자동으로 import 됨 - import 문 필요 없음
- java.util
  - 자바 유틸리티 패키지
    - 날짜, 시간, 벡터, 해시맵 등과 같은 다양한 유틸리티 클래스와 인터페이스 제공
- java.io
  - 키보드, 모니터, 프린터, 디스크 등에 입출력을 할 수 있는 클래스와 인터페이스 제공
- java.awt
  - 자바 GUI 프로그래밍을 위한 클래스와 인터페이스 제공
- javax.swing
  - 자바 GUI 프로그래밍을 위한 스윙 패키지

Object 클래스

- 특징
  - java.lang 패키지에 포함
  - 자바 클래스 계 층 구조의 최상위에 위치
  - 모든 클래스의 수퍼 클래스
- 주요 메소드

메소드	설명
protected Object clone()	현 객체와 똑같은 객체를 만들어 리턴
boolean equals(Object obj)	obj가 가리키는 객체와 현재 객체가 비교하여 같으면 true 리턴
Class getClass()	현 객체의 런타임 클래스를 리턴
int hashCode()	현 객체에 대한 해시 코드 값 리턴
String toString()	현 객체에 대한 스트링 표현을 리턴
void notify()	현 객체에 대해 대기하고 있는 하나의 스레드를 깨운다.
void notifyAll()	현 객체에 대해 대기하고 있는 모든 스레드를 깨운다.
void wait()	다른 스레드가 깨울 때까지 현재 스레드를 대기하게 한다.

String의 생성과 특징

- String - java.lang.String
  - String 클래스는 하나의 스트링만 표현

```
// 스트링 리터럴로 스트링 객체 생성
String str1 = "abcd";

// String 클래스의 생성자를 이용하여 스트링 생성
char data[] = {'a', 'b', 'c', 'd'};
String str2 = new String(data);
String str3 = new String("abcd"); // str2와 str3은 모두 "abcd" 스트링
```

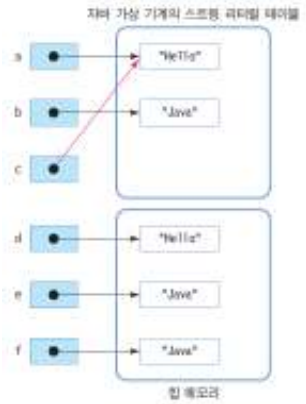
  - String 생성자

생성자	설명
String()	빈 스트링 객체 생성
String(char[] value)	문자 배열에 포함된 문자들을 스트링 객체로 생성
String(String original)	인자로 주어진 스트링과 똑같은 스트링 객체 생성
String(StringBuffer buffer)	스트링 버퍼에 포함된 문자들을 스트링 객체로 생성

스트링 리터럴과 new String()

- 스트링 생성
  - 단순 리터럴로 생성, String s = "Hello";
    - JVM이 리터럴 관리, 응용프로그램 내에서 공유됨
  - String 객체로 생성, String t = new String("Hello");
    - 힙에 String 객체 생성

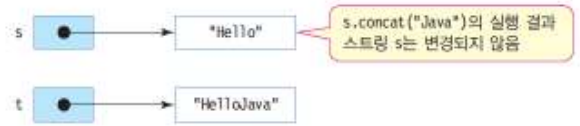
```
String a = "Hello";
String b = "Java";
String c = "Hello";
String d = new String("Hello");
String e = new String("Java");
String f = new String("Java");
```



스트링 객체의 주요 특징

- 스트링 객체는 수정 불가능

```
String s = new String("Hello");
String t = s.concat("Java"); // 스트링 s에 "Java"를 덧붙인 스트림을 리턴함.
```



- ==과 equals()
  - 두 스트링을 비교할 때 반드시 equals()를 사용하여야 함
    - equals()는 내용을 비교하기 때문

String 클래스 주요 메소드

메소드	설명
char charAt(int index)	지정된 index 인덱스에 있는 문자 값 리턴
int codePointAt(int index)	지정된 index 인덱스에 있는 유니코드 값 리턴
int compareTo(String anotherString)	두 스트링을 사전적 순서를 기준으로 비교. 두 스트링이 같으면 0, 현 스트링이 지정된 스트링보다 사전적으로 먼저 나오면 음수, 아니면 양수를 리턴
String concat(String str)	str 스트링을 현재 스트링 뒤에 덧붙인 스트링 리턴
boolean contains(CharSequence s)	s에 지정된 일련의 문자들을 포함하고 있으면 true 리턴
int length()	스트링의 길이 리턴
String replace(Charsequece target, Charsequence replacement)	target이 지정하는 일련의 문자들을 replacement가 지정하는 문자들로 변경한 스트링 리턴
String[] split(String regex)	정규식 regex에 일치하는 부분을 중심으로 스트링을 분리하고 분리된 스트링을 배열에 저장하여 리턴
String subString(int beginIndex)	beginIndex 인덱스부터 시작하는 서브 스트링 리턴
String toLowerCase()	스트링을 소문자로 변경한 스트링 리턴
String toUpperCase()	스트링을 대문자로 변경한 스트링 리턴
String trim()	스트링 앞뒤의 공백 문자들을 제거한 스트링 리턴

String 연결

- + 연산자로 문자열 연결
  - + 연산의 피연산자에 문자열이 있는 경우
  - + 연산에 객체가 포함되어 있는 경우
    - 객체.toString()을 호출하여 객체를 문자열로 변환한 후 문자열 연결
  - 기본 타입 값은 문자열로 변환된 후에 연결

```
System.out.print("abcd" + 1 + true + 3.13e-2 + 'E'+ "fgh" );
// abcd1true0.0313Efgh  출력
```

- String concat(String str)를 이용한 문자열 연결

```
"abcd".concat("efgh");
// "abcdefg" 리턴
```

  - 기존 String 객체에 연결되지 않고 새로운 스트링 객체 생성 리턴

String 비교

- int compareTo(String anotherString)
  - 문자열이 같으면 0 리턴
  - 이 문자열이 anotherString 보다 사전에 먼저 나오면 음수 리턴
  - 이 문자열이 anotherString 보다 사전에 나중에 나오면 양수 리턴

```
String a = "java";
String b = "jasa";
int res = a.compareTo(b);
if(res == 0)
  System.out.println("the same");
else if(res < 0)
  System.out.println(a +"<"+b);
else
  System.out.println(a +">"+b);
```

"java" 가 "jasa" 보다 사전에 나중에 나오기 때문에 양수 리턴

```
java>jasa
```

- 비교 연산자 ==는 문자열 비교에는 사용할 수 없음

String 공백 제거, 문자열의 각 문자 접근

- 공백 제거
  - String trim()
    - 문자열 앞 뒤 공백 문자(tab, enter, space) 제거한 문자열 리턴

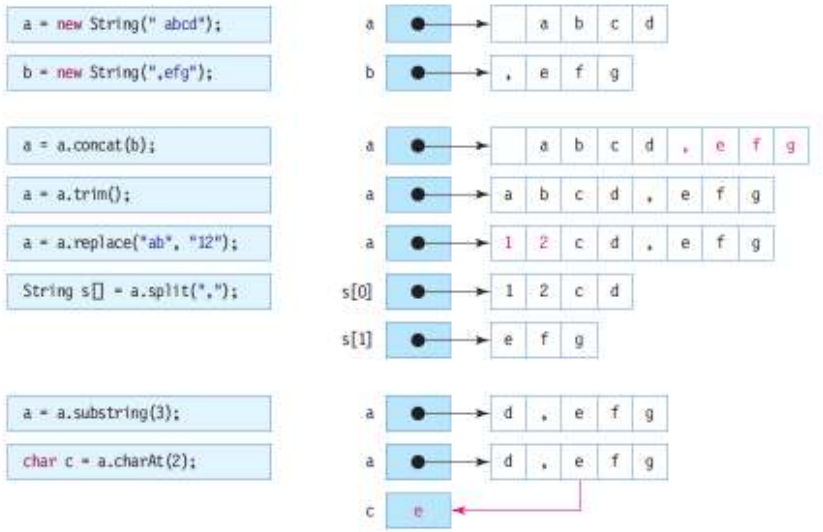
```
String a = " abcd def ";
String b = "₩txyz₩t";
String c = a.trim(); // c = "abcd def"
String d = b.trim(); // d = "xyz"
```

- 문자열의 문자
  - char charAt(int index)
    - 문자열 내의 문자 접근

```
String a = "class";
char c = a.charAt(2); // c = 'a'
```

```
// "class"에 포함된 's'의 개수를 세는 코드
int count = 0;
String a = "class";
// a.length()는 5
for(int i=0; i<a.length(); i++) {
  if(a.charAt(i) == 's')
    count++;
}
System.out.println(count); // 2 출력
```

String 예제



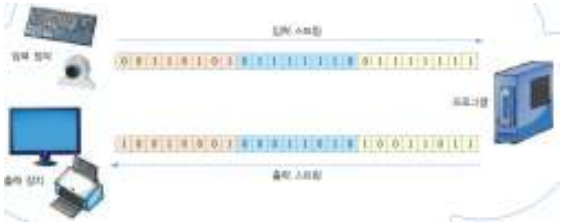
String 예제

- String 메소드
  - String substring(int beginIndex)는 beginIndex부터 나머지 부분을 string으로 반환
  - String substring(int beginIndex, int endIndex)는 beginIndex부터 endIndex까지 부분을 string으로 반환
  - boolean contains(CharSequence s)는 s를 가지고 있는지 여부를 true/false 반환
  - int lastIndexOf(String str)는 str을 가지고 있는 lastIndex 값을 반환

```
String dir = "C:\\JAVA";
String filename = "IMG1.jpg";
String fullPath = dir + "\\" + filename; // fullPath="C:\JAVA\IMG1.jpg"
if (filename.contains("jpg")) {
   String format = "png";
   String newName = fullPath.substring(0, fullPath.lastIndexOf('.')+1) + format;
   System.out.println("newName=" + newName); // newName="C:\JAVA\IMG1.png"
}
```

스트림

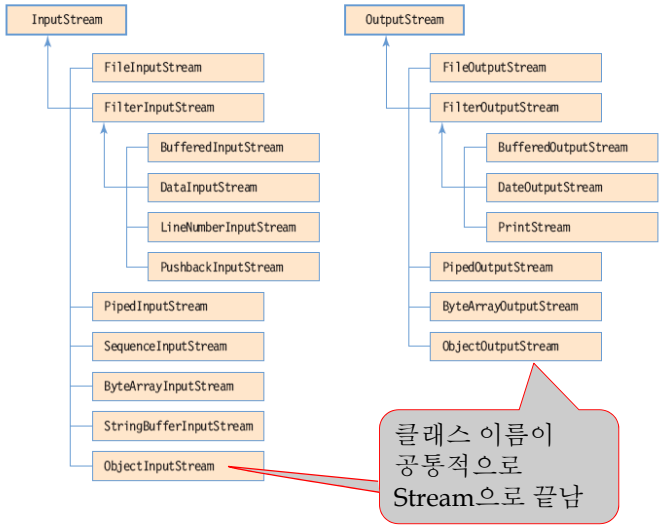
- 자바의 스트림
  - 자바 스트림은 입출력 장치와 자바 응용 프로그램 연결
  - 입출력 장치와 프로그램 사이의 데이터 흐름을 처리하는 소프트웨어 모듈
  - 입력 스트림
    - 입력 장치로부터 자바 프로그램으로 데이터를 전달하는 소프트웨어 모듈
  - 출력 스트림
    - 자바 프로그램에서 출력 장치로 데이터를 보내는 소프트웨어 모듈
- 입출력 스트림 기본 단위 : 바이트
- 자바 입출력 스트림 특징
  - 단방향 스트림, 선입선출 구조



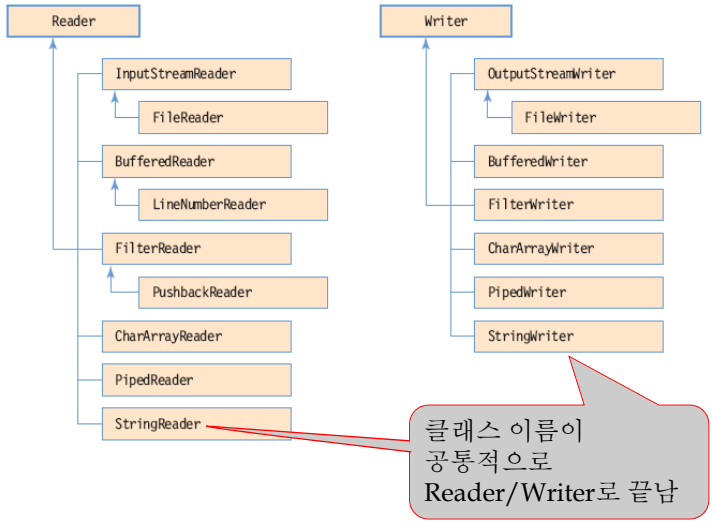
자바의 입출력 스트림 종류

- 바이트 입출력 스트림과 문자 입출력 스트림
  - 바이트 입출력 스트림
    - 입출력되는 데이터를 단순 바이트의 스트림으로 처리
    - 예) 바이너리 파일을 읽는 입력 스트림
  - 문자 입출력 스트림
    - 문자만 입출력하는 스트림
    - 문자가 아닌 바이너리 데이터는 스트림에서 처리하지 못함
    - 예) 텍스트 파일을 읽는 입력 스트림
- JDK는 입출력 스트림을 구현한 다양한 클래스 제공

JDK의 바이트 스트림 클래스 계층 구조



JDK의 문자 스트림 클래스 계층 구조



바이트 스트림 클래스

- 바이트 스트림
  - 바이트 단위의 바이너리 값을 읽고 쓰는 스트림
- 바이트 스트림 클래스
  - java.io 패키지에 포함
  - 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

FileInputStream을 이용한 파일 읽기

- 파일 전체를 읽어 화면에 출력하는 코드 샘플

```
FileInputStream fin = new FileInputStream("c:₩₩test.txt");

int c;

while((c = fin.read()) != -1) {

  System.out.print((char)c);

}

fin.close();
```

C:₩test.txt 파일을 열고 파일과 입력 바이트 스트림 객체 fin 연결

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

바이트 c를 문자로 변환하여 화면에 출력

스트림을 닫음. 파일도 닫힘. 스트림과 파일의 연결을 끊음. 더 이상 스트림으로부터 읽을 수 없음

문자 스트림 클래스

- 문자 스트림
  - 유니 코드로 된 문자를 입출력 하는 스트림
    - 문자로 표현되지 않는 데이터는 다루지 않음
    - 문자 스트림은 이미지, 동영상과 같은 바이너리 데이터는 입출력 할 수 없음 - 문자 스트림은 문자 데이터만 입출력 가능
- 문자 스트림을 다루는 클래스
  - Reader/Writer
    - java.io 패키지에 포함
    - 추상 클래스. 문자 스트림을 다루는 모든 클래스의 슈퍼 클래스
  - InputStreamReader/OutputStreamWriter
    - 바이트 스트림과 문자 스트림을 연결시켜주는 다리 역할
    - 지정된 문자집합 이용
    - InputStreamReader : 바이트를 읽어 문자로 인코딩
    - OutputStreamWriter : 문자를 바이트로 디코딩하여 출력
  - **FileReader/FileWriter**
    - 텍스트 파일에서 문자 데이터 입출력

FileReader를 이용한 텍스트 파일 읽기

- system.ini 파일 전체를 읽어 화면에 출력하는 코드 샘플

```
import java.io.*;
public class FileReaderEx {
  public static void main(String[] args) {
    FileReader in = null;
    try {
      // 파일로부터 문자 입력 스트림 생성
      in = new FileReader("c:₩₩windows₩₩system.ini");
      int c;
      while ((c = in.read()) != -1) { // 한 문자씩 읽는다.
            System.out.print((char)c);
      }
      in.close();
    }
    catch (IOException e) {
      System.out.println("입출력 오류");
    }    }  }
```

C:₩Windows₩system.ini 파일을 열고 문자입력스트림 연결

파일의 끝을 만나면 read()는 -1 리턴

버퍼 입출력 스트림 클래스

- 버퍼 스트림
  - 버퍼를 가진 스트림으로써 입출력 데이터를 일시적으로 저장하는 버퍼를 이용하여 입출력 효율 개선
  - 입출력 시 운영체제의 API 호출 횟수를 줄여 입출력 성능 개선
    - 출력 시 여러 번 출력되는 데이터를 버퍼에 모아두고 한 번에 장치로 출력
    - 입력 시 입력 데이터를 버퍼에 모아두고 한번에 프로그램에게 전달
- **바이트 버퍼 스트림 클래스**
  - BufferedInputStream와 BufferedOutputStream
  - 바이트 단위의 바이너리 데이터(Binary Data)를 처리하는 버퍼 스트림
- **문자 버퍼 스트림 클래스**
  - **BufferedReader와 BufferedWriter**
  - **유니코드의 문자 데이터(Text Data)만 처리하는 버퍼 스트림**

텍스트 파일 읽기

- BufferedReader 클래스를 사용한 텍스트 파일 read

```
import java.io.*;
public class BufferedReaderExample {
   public static void main(String args[])throws Exception{
        FileReader fr = new FileReader("C:/test.txt");
        BufferedReader br = new BufferedReader(fr);
        int i;
        while((i=br.read())!=-1){
        System.out.print((char)i);
        }
        br.close();
        fr.close();
   }
}
```

파일 전체를 읽어 화면에 출력

텍스트 파일 읽기

- BufferedReader 클래스를 사용한 텍스트 파일 readLine
  (파일을 한 줄씩 읽어서 lines 배열에 저장)

```
import java.io.*;
public class BufferedReaderExample2 {
   static String[] lines = new String[10]; // 10개의 라인 배열

   public static void main(String args[])throws Exception{
        BufferedReader br = new BufferedReader(new FileReader("C:/test.txt"));
        int i = 0;
        String line = "";
        while ((line=br.readLine())!= null){
           lines[i++] = line;
        }
        br.close();
        for (String l : lines) System.out.println(l); // lines 배열 출력
   }
}
```

한 줄씩 읽어 lines 배열에 저장

File 클래스

- File 클래스
  - 파일의 경로명을 다루는 클래스
    - java.io.File
    - 파일과 디렉터리 경로명의 추상적 표현
  - 파일 이름 변경, 삭제, 디렉터리 생성, 크기 등 파일 관리
    - File 객체는 파일 읽고 쓰기 기능 없음
  - 파일 입출력은 파일 입출력 스트림 이용

File 클래스 생성자와 주요 메소드

생성자	설명
File(File parent, String child)	parent 디렉터리의 child 이름의 디렉터리나 파일을 나타내는 File 객체 생성
File(String pathname)	pathname이 나타내는 File 객체 생성
File(String parent, String child)	parent 디렉터리의 child 이름의 디렉터리나 파일을 나타내는 File 객체 생성
File(URI uri)	file:URI를 추상 경로명으로 변환하여 File 객체 생성

메소드	설명
boolean mkdir()	새로운 디렉터리 생성
String[] list()	디렉터리 내의 파일과 디렉터리 이름의 문자열 배열 리턴
File[] listFiles()	디렉터리 내의 파일 이름의 File 배열 리턴
boolean renameTo(File dest)	dest가 지정하는 파일 이름 변경
boolean delete()	파일 또는 디렉터리 삭제
long length()	파일의 크기 리턴. 디렉터리나 장치 파일인 경우 0 리턴
String getPath()	파일 경로명 전체를 문자열로 변환하여 리턴
String getName()	파일 또는 디렉터리 이름을 문자열로 리턴
boolean isFile()	일반 파일이면 true 리턴
boolean isDirectory()	디렉터리이면 true 리턴
long lastModified()	파일이 마지막으로 변경된 시간 리턴
boolean exists()	파일 또는 디렉터리가 존재하면 true 리턴

File 클래스 예제

파일 객체 생성

```
File f = new File("c:₩₩test.txt");
```

파일인지 디렉터리인지 구분

```
File f = new File("c:₩₩windows₩₩system.ini");
String res;
if(f.isFile()) // 파일 타입이면
  res = "파일";
else // 디렉터리 타입이면
  res = "디렉터리";
System.out.println(f.getPath() + "은 " + res + "입니다.");
```

```
c:\windows\system.ini은 파일입니다.
```

서브 디렉터리 리스트 얻기

```
File f = new File("c:₩₩tmp₩₩java_sample");
String[] filenames = f.list(); // 파일명 리스트 얻기
for (int i=0; i<filenames.length; i++) {
  File sf = new File(f, filenames[i]);
  System.out.print(filenames[i]);
  System.out.print("₩t파일 크기: " + sf.length());
}
```