Java I/O Stream Ⅲ - 문자 Stream

문자를 주고받으려면 무조건 문자Stream을 써야 하는가?

Q. ByteStream을 사용할까, 문자 Stream을 사용할까?

A. ByteStream을 사용할 수도 있고, 문자 Stream을 사용할 수도 있다. 기준은 어디로 전송하는가에 따라 나눈다.

- Java Program은 문자를 2Byte 유니코드 기준으로 인코딩/디코딩 한다.
- OS 위에서 Java Program이 동작하는데, OS의 문자코드 Set이 유니코드가 아니더라도 상관없이, Java Program은 유니코드 기준으로 문자를 코딩한다.

바이트스트림과 문자스트림 구분하기

바이트 스트림

- 단순히 Java 프로그램에서 문자를 파일에 저장하고 다시 Java프로그램으로 읽어들이는 것은 ByteStream 만으로도 충분하다. 대다수는 바이트 스트림으로 데이터를 주고받는다.

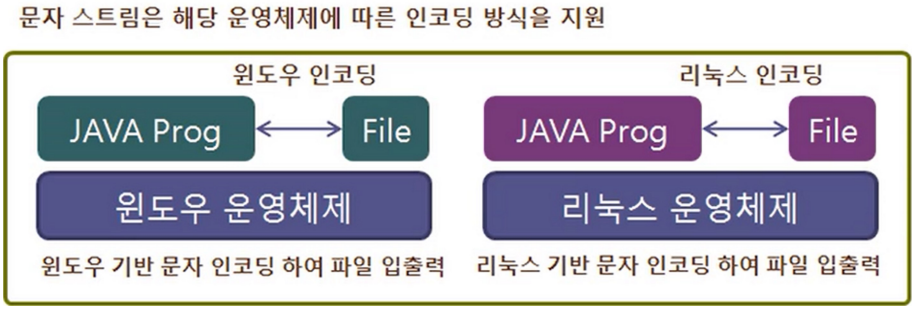
(Windows 상의) Java Program	← ByteStream →	(Linux 상의) Java Program

문자스트림

- Java Program과 OS간의 문자 인코딩/디코딩 기준이 다를 때 문제가 생긴다!
  Java Progmam의 문자 코딩 방식으로 쓰여진 파일을 OS에서 읽을 때,
  OS의 문자 코딩 방식이 다를 경우 문제가 발생한다. (OS에서 제대로 읽을 수가 없다.)

Java Program	← 문자Stream →	OS

- 그러므로 Java Progrma은 OS의 문자 코딩방식에 맞게 데이터를 주고받아야 한다.
- 문자스트림은 OS의 기준방식으로 문자를 인코딩/ Java의 기준방식으로 디코딩하는 스트림이다!



Reader And Writer

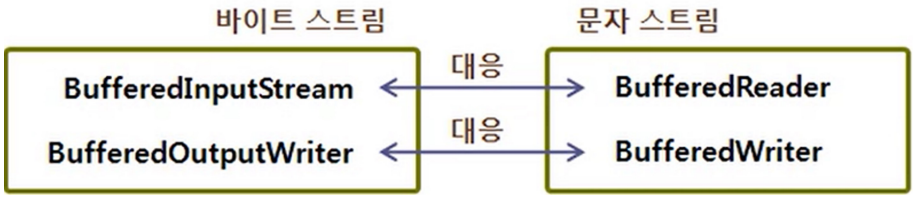
- **Reader의 대표적인 메소드**
  - **public int read() throws IOException**
  - **public abstract int read(char[] cbuf, int off, int len)**

- **Writer의 대표적인 메소드**
  - **public void write(int c) throws IOException**
  - **public abstract void write(char[] cbuf, int off, int len)**

```
public class ReadAndWriterStreamDemo {
     private static final String FILE_PATH = "D:/Documents/text.txt";

     public static void main(String[] args) throws IOException {
          char ch1 = 'A';
          char ch2 = 'B';
          char[] cbuf = new char[512];

          Writer writer = new FileWriter(FILE_PATH);
          Reader reader = new FileReader(FILE_PATH);

          // OS 기준으로 인코딩 후 쓰기
          writer.write(ch1);
          writer.write(ch2);

          writer.close();

          // Java 기준으로 디코딩 후 읽기
          reader.read(cbuf, 0, cbuf.length);

          for(int i=0; i<cbuf.length;i++)
               System.out.print(cbuf[i]);

          reader.close();
     }

}
```

문자열 입출력을 위한 BufferedReader And BufferedWriter

문자열의 입출력을 위해서 버퍼링을 해주는 클래스



- **문자열의 읽기 (BufferedReader)**

public String readLine() throws IOException

- **문자열의 쓰기 (BufferedWriter)**

  public void write(String str) throws IOException

```
public class BufferedReadAndWriterDemo {
     private static final String FILE_PATH = "D:/Documents/text.txt";

     public static void main(String[] args) throws IOException{
          String str = "";

          Writer writer = new FileWriter(FILE_PATH);
          BufferedWriter bw = new BufferedWriter(writer);

          bw.write("The Rain");
          bw.newLine(); // OS마다 개행정보가 다르므로 newLine() Method로 개행입력
          bw.newLine();

          bw.write("비가 그친지 몇 날이 지났음에도"); bw.newLine();
          bw.write("나는 아직도 이렇게 젖어있어요"); bw.newLine();
          bw.write("추억이라는 얕은 시간 속에"); bw.newLine();
          bw.write("여전히 내리는 이별의 비는"); bw.newLine();
          bw.write("나를 다 집어 삼켜"); bw.newLine();

          bw.close();
          writer.close();

          Reader reader = new FileReader(FILE_PATH);
          BufferedReader br = new BufferedReader(reader);

          while(true){
               str = br.readLine();
               if(str == null)
                    break;

               // 개행정보는 문자열의 구분자로 사용되므로 readLine() 호출시 버려진다.
               // 그러므로 println()으로 호출해야 개행된 출력을 볼 수 있다.
               System.out.println(str);
          }

          br.close();
          reader.close();
     }
}
```
