파일 입출력

1. 파일 입출력

    A. Java에서 파일의 입출력을 위해 제공되는 클래스들은 다음의 것들을 포함한다.

        i. **바이트 단위 (바이너리) 입출력(Byte I/O)**: FileInputStream, FileOutputStream, BufferedInputStream, BufferedOutStream

        ii. **문자 단위 (텍스트) 입출력(Character I/O)**: FileReader, FileWriter, BufferedReader, BufferedWriter, PrintWriter

        iii. **바이트-문자 변환(Byte-to-Character conversion)**: InputStreamReader, OutputStreamWriter

2. (텍스트 파일 생성 및 쓰기) 다음 프로그램을 입력하고 실행하시오.

   A. 실행 후 C:/Temp/data.txt의 파일 크기(바이트)와 그 내용을 확인해 보시오. (이클립스 프로젝트 properties 창에서 Text file encoding 속성 값을 UTF-8 혹은 MS949로 변경 후 각각에 대해 확인)

   B. C:/Temp/data.txt를 Z:/Temp/data.txt로 변경하여 실행해 보시오. w.close();를 생략하고 실행 후 C:/Temp/data.txt의 내용을 확인해 보시오.

```
public class Test {
	public static void main(String[] args) {
		try {
			FileWriter w=new FileWriter("C:/Temp/data.txt");
			w.write("Hi");
			w.write("안녕");
			w.close();
		} catch (Exception e) {
			System.out.println(e.getMessage());
		}
	}
}
```

3. (**실습**) C:/Temp/info.txt 파일을 새롭게 만들어 다음 내용을 저장하는 프로그램을 작성하시오.

빼앗긴
들에도
봄은
오는가

4. 다음 프로그램을 입력하고 실행하시오. 반복 실행하면서 각 실행 후 C:/Temp/memo.txt의 내용을 확인해 보시오. 개선의 필요가 없는가?

```
public class Test {
	public static void main(String[] args) {
		try {
			String		v=JOptionPane.showInputDialog("메모 입력:");
			FileWriter w=new FileWriter("C:/Temp/memo.txt");
			w.write(v+"\n");
			w.close();
		} catch (Exception e) {
			System.out.println(e.getMessage());
		}
	}
}
```

5. 다음 프로그램을 입력하고 실행하시오. 반복 실행하면서 각 실행 후 C:/Temp/memo.txt의 내용을 확인해 보시오. JOptionPane.showInputDialog()의 입력창에서 취소 버튼을 클릭(null이 반환됨) 후 v 및 memo.txt의 내용을 확인해 보시오. 개선의 필요가 없는가?

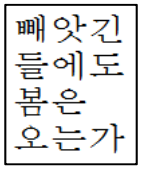
```
public class Test {
	public static void main(String[] args) {
		try {
			String		v=JOptionPane.showInputDialog("메모 입력:"); // 취소 버튼 클릭 시 null이 반환됨
			FileWriter w=new FileWriter("C:/Temp/memo.txt", true); // true인 경우 파일의 마지막 위치에 데이터 출력됨 (append 모드)
			w.write(v+"\n");
			w.close();
		} catch (Exception e) {
			System.out.println(e.getMessage());
		}

	}
}
```

6. (**실습**) 이전 프로그램을 수정하여 한 번 실행에서 메모를 반복적으로 입력할 수 있도록 하시오.

7. (**실습**) 다음 명령문을 실행하면 C:/Temp/memo.txt 파일의 마지막 위치에 **중간고사 끝**이라는 새로운 행을 추가하여 저장한다. LineAdd 프로그램을 작성하시오.

   java LineAdd C:/Temp/memo.txt "중간고사 끝" [enter]

8. (텍스트 파일 읽기) 메모장에서 우측 박스의 텍스트를 입력하여 C:/Temp/info.txt 파일에 저장한 후, 다음 프로그램을 입력하고 실행하시오. 아래 try 절의 첫 문장을 두 개 문장으로 작성해 보시오.




```
public class Test {
    public static void main(String[] args) {
        try {
            Scanner iF=new Scanner(new FileReader("C:/Temp/info.txt"));
            while(iF.hasNext()){ // 읽어 들일 토큰이 있으면 불리언 true 반환
                System.out.println(iF.nextLine()); // 입력 소스로부터 다음 행을 문자열로 반환
            }
            iF.close();
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

9. (**실습**) 이전 프로그램을 수정하여 C:/Temp/info.txt 파일의 세 번째 행만 출력하도록 하시오.

10. (**실습**) 다음 명령문을 실행하면 C:/Temp/info.txt 파일의 2번째 행을 출력한다. LineExtract 프로그램을 작성하시오.

    java LineExtract C:/Temp/info.txt 2 [enter]

11. (디렉토리 생성) 다음 프로그램을 입력하고 실행하시오. 실행 후 C:/MyTemp/MyData 디렉토리 아래 data.txt 파일이 생성되었는지 확인하시오.

```
public class Test {
	public static void main(String[] args) {
		try {
			File	dir=new File("C:/MyTemp/MyData/");
			if(dir.exists()==false) dir.mkdirs();
			FileWriter w=new FileWriter("C:/MyTemp/MyData/data.txt");
			w.write("안녕");
			w.close();
		} catch (Exception e) {
			System.out.println(e.getMessage());
		}
	}
}
```

12. (**실습**) C:/Java001/Data/ 디렉토리를 새롭게 생성하고 크기 0의 C:/Java001/Data/Hello.txt 파일을 생성하는 코드를 작성하시오.

13. (**실습**) 다음 명령문을 실행하면 크기 0의 C:/Temp/A/B/C/data.txt 파일을 생성한다. Touch 프로그램을 작성하시오.

java Touch C:/Temp/A/B/C/data.txt [enter]

14. (시간 날짜 처리) 다음 프로그램을 입력하고 실행하시오.

```
public class Test {
	public static void main(String[] args) {
		long	curTime=System.currentTimeMillis();
		System.out.println(curTime);
		Date	date=new Date(curTime);
		System.out.println(date);
		SimpleDateFormat		sdf=new SimpleDateFormat("yyyy-MM-dd HH:mm:ss (E)");
		String			datetime=sdf.format(date);
		System.out.println(datetime);
	}
}
```

15. (**실습**) JOptionPane.showInputDialog()를 통해 입력 받은 한 줄 메모(예: 시작이 반이다)의 앞에 현재 시각 정보를 부착한 문자열(아래 예 참조)을 C:/Temp/memo.txt의 마지막에 추가 저장하는 프로그램을 작성하시오. 현재 시각 정보와 메모의 구분자로 하나의 탭 문자를 사용하시오.

2018-05-08 15:26:26 (화) *시작이 반이다.*

16. (한글 인코딩) 다음 프로그램을 입력하고 실행하시오.

```
public class Test {
	public static void main(String[] args) {
		// Reference: 오영은. 2012. 한글 인코딩의 이해 1편: 한글 인코딩의 역사와 유니코드 (https://d2.naver.com/helloworld/19187)
		// Reference: 오영은. 2012. 한글 인코딩의 이해 2편: 유니코드와 Java를 이용한 한글 처리 (https://d2.naver.com/helloworld/76650)
		try {
			char		c='가';	// char	c=0xAC00; test 힣
			System.out.println(Integer.toHexString(c));
			System.out.printf("%02X", c>>8 & 0xFF);
			System.out.printf("%02X", c & 0xFF);
			System.out.println();
			System.out.println(0xD7A3-0xAC00+1); // charmap.exe 가~힣

			for (byte b : "가".getBytes("EUC-KR")){ // 한글 2350, test 갘, 돽
				System.out.printf("%02X", b);
			}
			System.out.println();

			for (byte b : "가".getBytes("UTF-8")){ // test 갘, 돽
				System.out.printf("%02X", b);
			}
			System.out.println();

		} catch (Exception e) {
			e.printStackTrace();
		}
	}
}
```

17. 다음 프로그램을 입력하고 실행하시오. (바이트 단위 파일 쓰기)

A. 다음은 data.bin이라는 새 파일을 만들고, 문자열 "ABC₩n"에 해당하는 바이트들을 파일에 저장하는 코드이다.

i. 아래 코드에서 new FileOutputStream("data.bin", **true**)라고 하면 기존에 존재하는 파일 "data.bin"의 마지막 위치부터 쓰게 됨.

B. 윈도우 탐색기에서 이 프로그램의 이클립스 프로젝트 디렉토리를 열어 data.bin의 내용을 확인해 볼 것.

```
public class Test {
	public static void main(String[] args) {
		try {
			FileOutputStream	os=new FileOutputStream("data.bin");
			//	BufferedOutputStream	os=new BufferedOutputStream(new FileOutputStream("data.bin"));
			os.write("ABC\n".getBytes()); // 파일에 바이트열(byte sequence) 쓰기. os.write("ABC\n");라고 고쳐 실행해 볼 것. 오류 발생 이유는?
			os.close();
		} catch (Exception e1) {
			e1.printStackTrace();
		}
	}
}
```

18. 다음 프로그램을 입력하고 실행하시오. (바이트 단위 파일 읽기)

A. 다음은 이전 문제에서 작성한 data.bin이라는 파일의 내용을 한 바이트씩 읽고 그 바이트들을 16진수로 출력하는 코드이다.

```
public class Test {
	public static void main(String[] args) {
		try {
			FileInputStream		is=new FileInputStream("data.bin");
			//		BufferedInputStream  is=new BufferedInputStream(new FileInputStream("data.bin"));
			while(true){
				int		data=is.read();		// 오픈된 파일로부터 다음 한 바이트를 읽어 정수형 변수 data에 저장한다.
				if(data==-1) break; // 읽은 바이트 값이 -1(혹은 0보다 작으면)이면 파일 끝을 의미하므로 반복문을 탈출(break)한다.
				byte		byteData=(byte) data; // 읽은 바이트 값이 정수형 변수에 저장되어 있으므로 바이트 자료형으로 형 변환한다.
				System.out.printf("%02X\n", byteData);	// 바이트값을 16진수로 출력한다. 41 42 43 0A가 출력됨. 16진수 41은 'A', 0A는 '\n'
			}
			is.close();
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
}
```

19. (**실습**) 아래 예와 같이 명령행 인자로 주어진 파일의 내용을 한 행에 16바이트씩 각 바이트를 2자리 16진수로 출력하는 프로그램을 작성하시오.

java HexDump A.hwp [enter]

```
88 64 96 07 AB 61 45 98 C3 B8 09 EC AF 2E 10 C9
DF 0C 16 59 08 26 83 80 61 F2 3F 7D 25 58 16 AC
9E 81 03 4C B2 21 91 9C 60 92 09 4C 32 30 E2 23
19 7F 23 D8 4C FF C1 22 0C 00 01 06 00 46 EE 21
AE 0D 0A 65 6E 64 73 74 72 65 61 6D 0D 65 6E 64
6F 62 6A 0D 73 74 61 72 74 78 72 65 66 0D 0A 31
31 36 0D 0A 25 25 45 4F 46 0D 0A
```

20. 다음 프로그램을 입력하고 실행하시오. (버퍼 기반 바이트 단위 파일 읽기)

A. 다음은 이전 문제에서 작성한 data.bin이라는 파일의 바이트 크기를 출력하는 코드이다.

```
public class Test {
	public static void main(String[] args) {
		try {
			int		cnt=0;
			byte		buf[]=new byte[1024];
			FileInputStream	is=new FileInputStream("C:/Temp/data.bin");
			while(true){
				int		data=is.read(buf); // 파일로부터 최대 1024바이트까지 buf로 읽어 들임. 읽은 바이트 수 반환
				if(data==-1) break;
				cnt+=data;
			}
			is.close();
			System.out.println(cnt);
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
}
```

21. 다음 프로그램을 입력하고 실행하시오. (문자 단위 파일 쓰기)

A. 텍스트 파일은 문자(들)로 구성되어 있다. 텍스트 파일에 대해 읽기, 쓰기를 할 경우 바이트 단위 대신 문자 단위로 읽기 혹은 쓰기를 지원하는 클래스의 객체를 사용하는 것이 편리할 수 있다.

B. 다음은 문자 단위로 파일에 출력을 하는 코드의 예이다. 다음은 "data.txt"라는 새 텍스트 파일을 만들고 그 내용으로 "안녕하세요!₩nGood afternoon. ₩n", '반', '갑', '습', '니', '다', '.', '₩n','끝'의 데이터를 문자 단위로 저장하는 코드이다.

i. 아래 코드에서 new FileWriter("data.txt", **true**)라고 하면 기존에 존재하는 파일 "data.bin"의 마지막 위치부터 쓰게 됨.

C. 윈도우 탐색기에서 이 프로그램의 이클립스 프로젝트 디렉토리를 열어 data.txt의 내용을 확인해 볼 것.

```
public class Test {
	public static void main(String[] args) {
		try {
			FileWriter  w=new FileWriter("data.txt");
			//	BufferedWriter w=new BufferedWriter(new FileWriter("data.txt"));	// Buffering
			w.write("안녕하세요!\nGood afternoon. \n");		// 객체 w에 대해 문자열 "안녕하세요!\nGood afternoon. \n"을 write()한다.
			char	cbuf[]={'반', '갑', '습', '니', '다', '.', '\n'};
			w.write(cbuf);		// 객체 w에 대해 문자 배열 cbuf의 내용을 write()한다.
			w.write('끝');		// 객체 w에 대해 문자 '끝'을 write()한다.
			w.close();
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
}
```

22. 다음 프로그램을 입력하고 실행하시오. (문자 단위 파일 읽기)

A. 다음은 앞에서 만든 텍스트 파일 "data.txt"의 내용을 문자 단위로 읽어 화면에 출력하는 코드이다.

```
public class Test {
	public static void main(String[] args) {
		try {
			FileReader		r=new FileReader("data.txt");
			//	BufferedReader		r=new BufferedReader(new FileReader("data.txt"));
			while(true){
				int		data=r.read();		// 파일 객체 r로부터 하나의 문자를 읽어 정수형 변수 data에 저장한다.
				if(data==-1) break;	// 읽어 들인 문자의 값이 -1이면 파일의 마지막을 의미하므로 while 반복문을 탈출(break)한다.
				char		ch=(char)data;		// 정수형 변수 data에 읽어 들인 문자를 문자형 자료로 형 변환(type casting)한다.
				System.out.print(ch);	// 읽어 들인 문자를 화면에 출력한다.
			}
			r.close();
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
}
```

23. 다음 프로그램을 입력하고 실행하시오. (텍스트 파일 라인 입력 방법 I: **BufferedReader**)

A. 텍스트 파일은 행(라인)으로 이루어져 있으므로, 문자 단위보다 행(라인) 단위로 읽을 수 있다면 편리할 것이다.

```
public class Test {
	public static void main(String[] args) {
		try {
			BufferedReader	br=new BufferedReader(new FileReader("data.txt"));
			while(true){
				// BufferedReader 객체는 readLine() 메소드를 통해 라인단위 읽기 가능.
				String	line=br.readLine();
				if(line==null) break;  // 파일에서 읽어 들인 라인 문자열이 null이면 파일의 마지막을 의미하므로 while루프를 탈출한다.
				System.out.println(line);		// 읽어 들인 라인 문자열을 화면에 출력한다.
			}
			br.close();
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
}
```

24. 다음 프로그램을 입력하고 실행하시오. (텍스트 파일 라인 입력 방법 II: **Scanner**)

```
public class Test {
	public static void main(String[] args) {
		try {
			Scanner	scanner = new Scanner(new FileReader("data.txt"));
			while(scanner.hasNext()){ // scanner.hasNext() => 파일의 현재 위치에 더 읽을 내용이 남아 있으면 true, 그렇지 않으면 false
				String	line=scanner.nextLine();		// 파일의 현재 위치에서 다음 라인(문자열)을 읽어 line에 저장.
				System.out.println(line);		// 읽어 들인 라인 문자열을 화면에 출력.
			}
			scanner.close();
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
}
```

25. 다음 프로그램을 입력하고 실행하시오. (텍스트 파일 라인 출력 및 포맷 출력)

A. 텍스트 파일에 행 단위로 쓰거나 포맷된 문자열을 쓸 수 있다면 편리할 것이다.

```
public class Test {
	public static void main(String[] args) {
		try {
			PrintWriter pw=new PrintWriter("data.formated.txt");
			String	name="Obama";
			int	score=97;
			pw.println("Hi");
			pw.printf("%s\t%d\n", name, score);		// PrintWriter 객체가 제공하는 printf() 메소드를 통해 포맷된 문자열을 파일에 쓴다.
			pw.close();
		} catch (Exception e1) {
			e1.printStackTrace();
		}
	}
}
```

26. 다음 프로그램을 입력하고 실행하시오. (키보드 입력)

A. 다음은 키보드(표준입력)로부터의 입력을 처리하는 코드의 예이다. 아래의 System.in은 표준입력을 의미하며, 디폴트로 표준입력은 키보드입력을 의미한다.

```
public class Test {
	public static void main(String[] args) {
		Scanner	si=new Scanner(System.in);
		while(true){
			String	line=si.nextLine();
			if(line.equals("bye")) break;	// 키보드에서 bye[enter]를 입력하면 while 루프를 탈출한다.
			System.out.println(line.length()); // 키보드에서 읽은 한 라인 문자열의 문자 개수를 계산하여 화면에 출력한다.
		}
		si.close();
	}
}
```

27. 다음 프로그램을 입력하고 실행하시오. (UTF-8 파일 출력)

A. UTF-8 파일→PrintWriter(문자들을 버퍼를 통해 출력)→OutputStreamWriter(문자를 바이트로 변환)→FileOutputStream(바이트를 출력)

B. OutputStreamWriter에서는 문자를 바이트로 변환할 때 사용할 문자표를 지정할 수 있으며 아래 코드의 예는 출력 파일 data.UTF-8를 UTF-8로 인코딩하여 출력하기 위해 UTF-8 문자표를 사용한 것임.

C. 자바SE 지원 문자 인코딩 리스트: http://docs.oracle.com/javase/7/docs/technotes/guides/intl/encoding.doc.html

```
public class Test {
	public static void main(String[] args) {
		try {
			// 아래 코드의 UTF-8을 EUC-KR로 변경하여 실행해 볼 것
			PrintWriter oF=new PrintWriter(new OutputStreamWriter(new FileOutputStream("data.UTF-8"), "UTF-8"));
			oF.println("동해물과 백두산이 마르고 닳도록");
			oF.println("하느님이 보우하사 우리나라 만세");
			oF.flush();
			oF.close();
		} catch (Exception e) {
		}
	}
}
```

28. 다음 프로그램을 입력하고 실행하시오. (UTF-8 파일 입력)

A. UTF-8 파일→FileInputStream(바이트 단위 입력)→InputStreamReader(바이트(들)을 문자로 변환)→BufferedReader(문자들을 버퍼를 통해 입력)

B. InputStreamReader에서는 바이트를 문자로 변환할 때 사용할 문자표를 지정할 수 있으며 아래 코드 예에서는 입력 파일 data.UTF-8이 UTF-8로 인코딩된 것이므로 UTF-8 문자표를 사용하였음.

C. 자바SE 지원 문자 인코딩 리스트: http://docs.oracle.com/javase/7/docs/technotes/guides/intl/encoding.doc.html

```
public class Test {
	public static void main(String[] args) {
		try {
			// 아래 코드의 UTF-8을 EUC-KR로 변경하여 실행해 볼 것
			BufferedReader	iF=new BufferedReader(new InputStreamReader(new FileInputStream("data.UTF-8"), "UTF-8"));
			while(true){
				String	line=iF.readLine();
				if(line==null) break;
				System.out.println(line);
			}
			iF.close();
		} catch (Exception e) {
		}
	}
}
```

29. 실습 과제 I: 다음 예와 같이 명령행 인자로 주어진 임의 파일의 바이트 단위 크기를 출력하는 프로그램 Size.java를 작성하시오.

    java Size A.jpg [enter]

    753

30. 실습 과제 II: 다음 예와 같이 명령행 인자로 주어진 임의 텍스트 파일의 라인수, 토큰수, 문자수를 출력하는 프로그램 WordCount.java을 작성하시오. 토큰은 공백, ₩t, ₩r, ₩n 문자 이외 문자(들)의 나열로 정의한다.

    java WordCount A.txt [enter]

    4 5 31

31. (파일복사) 실습 과제 III: 다음 예와 같이 명령행 인자로 주어진 임의 파일을 복사하는 프로그램 Copy.java를 작성하시오. A.hwp.copy가 만들어져야 함.

    java Copy A.hwp [enter]

32. (바이트변환) 실습 과제 IV: 명령행 인자로 주어진 임의 파일을 다음과 같이 암호화하는 프로그램을 작성하시오. HalfByteSwap은 파일의 각 바이트에 대해 상위 4비트와 하위 4비트를 교환하여 저장한다. 예를 들어 바이트 $AC_{(16)}$는 $CA_{(16)}$로 변환 저장됨.

    java HalfByteSwap A.hwp [enter] ➔ A.hwp.enc가 만들어져야 함.

    java HalfByteSwap A.hwp.enc [enter] ➔ hwp.enc.enc가 만들어져야 하며 이 파일은 원래 파일인 A.hwp와 같아야 함.

References

김윤명. (2008). 뇌를 자극하는 Java 프로그래밍. 한빛미디어.

남궁성. 자바의 정석. 도우출판.

김윤명. (2010). 뇌를 자극하는 JSP & Servlet. 한빛미디어.