파일 입출력

파일 입출력

❖ FileInputStream

- 파일로부터 바이트 단위로 읽어 들일 때 사용
  - 그림, 오디오, 비디오, 텍스트 파일 등 모든 종류의 파일을 읽을 수 있음
- 객체 생성 방법
  - FileInputStream 객체가 생성될 때 파일과 직접 연결
  - 만약 파일이 존재하지 않으면 FileNotFoundException 발생
  - try-catch문으로 예외 처리

```
//첫번째 방법
FileInputStream fis = new FileInputStream("C:/Temp/image.gif");

//두번째 방법
File file = new File("C:/Temp/image.gif");
FileInputStream fis = new FileInputStream(file);
```

❖ FileInputStream

- InputStream 하위 클래스 - 사용 방법이 InputStream과 동일

```
FileInputStream fis = new FileInputStream("C:/Temp/image.gif");
int readByteNo;
byte[ ] readByes = new byte[100];
while ( (readByteNo = fis.read(readBytes) ) != -1 ) {
  //읽은 바이트 배열(readBytes)을 처리
}
fis.close();
```

❖ 텍스트 파일을 읽고 출력: FileInputStreamExample.java

```
import java.io.FileInputStream;

public class FileInputStreamExample {
    public static void main(String[] args) {
        try {
            FileInputStream fis = new FileInputStream(
                                        "C:/temp/FileInputStreamExample.java");

            int data;
            while ( (data = fis.read() ) != -1 ) { // 1byte씩 읽고 콘솔에 출력
                System.out.write(data);
            }
            fis.close();
        } catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```

❖ FileOutputStream

- 파일에 바이트 단위로 데이터를 저장할 때 사용
  - 그림, 오디오, 비디오, 텍스트 등 모든 종류의 데이터를 파일로 저장
- 객체 생성 방법
  - 파일이 이미 존재할 경우, 데이터를 출력하게 되면 파일을 덮어쓰는 단점

```
//방법1
FileOutputStream fis = new FileOutputStream("C:/Temp/image.gif");

//방법2
File file = new File("C:/Temp/image.gif");
FileOutputStream fis = new FileOutputStream(file);
```

- 기존 파일 내용 끝에 데이터를 추가할 경우

```
FileOutputStream fis = new FileOutputStream("C:/Temp/data.txt", true);
FileOutputStream fis = new FileOutputStream(file, true);
```

❖ FileOutputStream

- OutputStream 하위 클래스 - 사용 방법이 OutputStream과 동일

```
FileOutputStream fos = new FileOutputStream("C:/Temp/image.gif");
byte[ ] data = ...;
fos.write(data);
fos.flush();
fos.close();
```

❖ 파일 복사: FileOutputStreamExample.java

```
import java.io.FileInputStream;
import java.io.FileOutputStream;

public class FileOutputStreamExample {
    public static void main(String[] args) throws Exception {
        String originalFileName = "C:/temp/house.jpg";
        String targetFileName = "C:/Temp/house_copy.jpg";

        FileInputStream fis = new FileInputStream(originalFileName);
        FileOutputStream fos = new FileOutputStream(targetFileName);

        int readByteNo;
        byte[] readBytes = new byte[100];
        while( (readByteNo = fis.read(readBytes)) != -1 ) {
            fos.write(readBytes, 0, readByteNo);
        }
```

❖ 파일 복사: FileOutputStreamExample.java

```
        fos.flush();
        fos.close();
        fis.close();

        System.out.println("복사가 잘 되었습니다.");
    }
}
```

❖ FileReader

- 텍스트 파일로부터 데이터를 읽어 들일 때 사용
  - 문자 단위로 읽음
  - 텍스트가 아닌 그림, 오디오, 비디오 등의 파일은 읽을 수 없음
- 객체 생성 방법

```
//방법 1
FileReader fr = new FileReader("C:/Temp/file.txt");

//방법 2
File file = new File("C:/Temp/file.txt");
FileReader fr = new FileReader(file);
```

- FileReader 객체가 생성될 때 파일과 직접 연결
- 만약 파일이 존재하지 않으면 FileNotFoundException 발생
- try-catch문으로 예외 처리

❖ FileReader

- Reader 하위 클래스 - 사용 방법 Reader와 동일

```
FileReader fr = new FileReader("C:/Temp/file.txt");
int readCharNo;
char[ ] cbuf = new char[100];
while ((readCharNo=fr.read(cbuf)) != -1) {
  //읽은 문자 배열(cbuf)를 처리
}
fr.close();
```

❖ 텍스트 파일 읽기: FileReaderExample.java

```
import java.io.FileReader;

public class FileReaderExample {
    public static void main(String[] args) throws Exception {
        FileReader fr = new FileReader("C:/temp/FileReaderExample.java");

        int readCharNo;
        char[] cbuf = new char[100];
        while ((readCharNo=fr.read(cbuf)) != -1) {
            String data = new String(cbuf, 0, readCharNo);
            System.out.print(data);
        }
        fr.close();
    }
}
```

❖ FileWriter

- 텍스트 파일에 문자 데이터를 저장할 때 사용
  - 텍스트가 아닌 그림, 오디오, 비디오 등의 데이터를 파일로 저장 불가

- 객체 생성 방법
  - 파일이 이미 존재할 경우, 데이터를 출력하게 되면 파일을 덮어쓰게 됨.
  - 파일 존재여부 따라 분기

```
FileWriter fw = new FileWriter("C:/temp/file.txt");
```

- 기존 파일 내용 끝에 데이터를 추가할 경우

```
FileWriter fw = new FileWriter("C:/temp/file.txt", true);
FileWriter fw = new FileWriter(file, true);
```

❖ FileWriter

- Writer 하위 클래스 - 사용 방법이 Writer와 동일

```
FileWriter fw = new FileWriter("C:/Temp/file.txt");
String data = "저장할 문자열";
fw.write(data);
fw.flush();
fw.close();
```

❖ 문자열을 파일에 저장: FileWriterExample.java

```
import java.io.File;
import java.io.FileWriter;

public class FileWriterExample {
    public static void main(String[] args) throws Exception {
        File file = new File("C:/Temp/file.txt");
        FileWriter fw = new FileWriter(file, true);

        fw.write("FileWriter는 한글로된 " + "\r\n");
        fw.write("문자열을 바로 출력할 수 있다." + "\r\n");

        fw.flush();
        fw.close();

        System.out.println("파일에 저장되었습니다.");
    }
}
```