제 46강

입출력,자바의 IO 패키지2

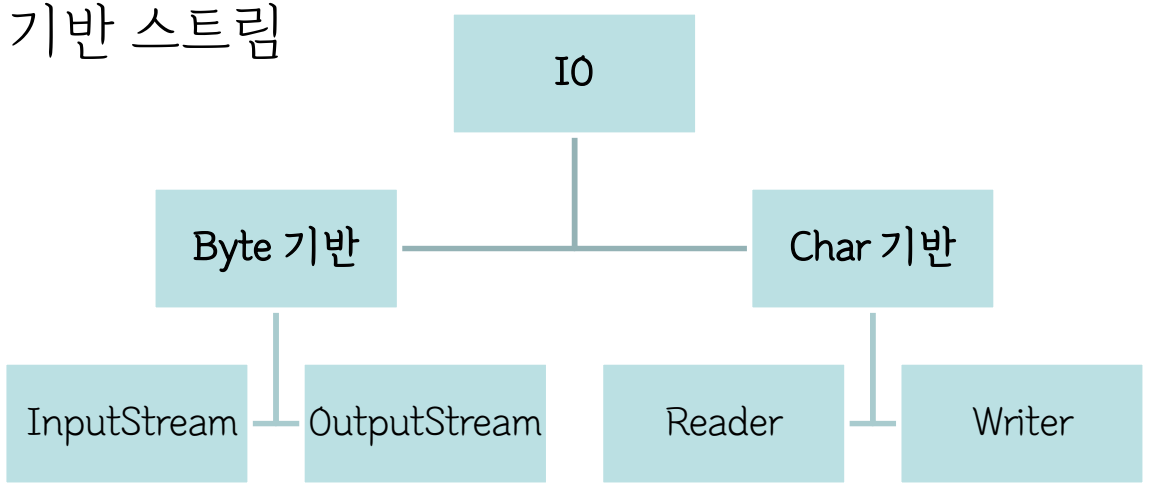
교재: p263~270

목차

복습

스트림(Stream)

- 단 방향이다.
- FIFO 구조를 갖고 있다.
- 출력단위에 따라 바이트 기반과 문자단위로 분류된다.
  > 출력단위: 1byte-> 바이트 기반 스트림
  문자 -> 문자 기반 스트림



1. 문자 기반 스트림

문자 기반 스트림?

16bit의 문자나 문자열을 읽고 쓰는 스트림

[1] 문자 기반 스트림의 활용

자바에서 사용하는 chat 형 변수는 2byte 이므로 바이트 기반 스트림은 불편하므로, 문자 단위로 입출력을 다루는 문자 기반 스트림을 제공

Reader	Writer
int read( )	void write(int c)
int read(char [] cbuf)	void write(char [] cbuf)
abstract int read(char [] cbuf,int off, int len)	abstract void write(char [] cbuf,int off, int len) void write(String str) void write(String str,int off, int len)

1. 문자 기반 스트림

- 대상의 따른 스트림 종류

입력 스트림	출력 스트림	대상
FileReader	FileWriter	파일
PipedReader	PipedWriter	메모리
CharArrayReader	CharArrayWriter	프로세스

사용 예시

```
FileReader fr = null;
fr = new FileReader( "파일 경로");
```

2. 문자 기반 보조 스트림

문자 기반 보조 스트림?

문자 기반 스트림의 성능을 향상시키는 역할

[1] 문자 기반 보조 스트림의 활용

입력 보조 스트림	출력 보조 스트림	대상
FilterReader	FilterWriter	필터를 이용한 문자 입출력
BufferedReader	BufferedWriter	버퍼를 이용한 문자 입출력

사용 예시

```
FileReader fr = null;
fr = new FileReader( "파일 경로");
BufferedReader br = new BufferedReader(fr);
```

3. 스트림의 예외 처리

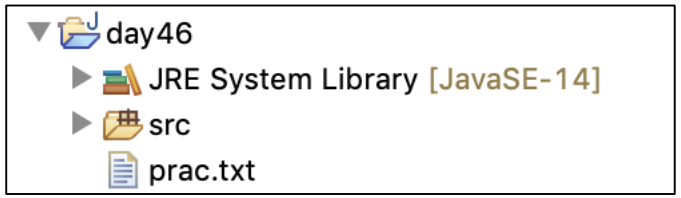
입출력의 모든 메서드는 IOException이 발생할 가능성이 높으므로 예외 처리가 필요하다.

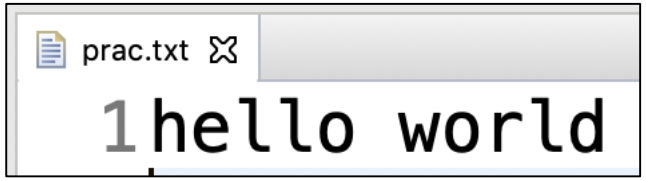
<실습> Exam101.java

파일 복사 프로그램의 예외처리코드, 문자 기반 스트림 활용

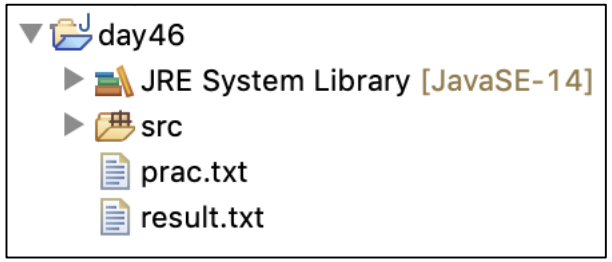
```java
public class File02 {
    public static void main(String[] args) {
//      파일 복사 프로그램
        FileReader fr=null;
        FileWriter fw =null;

        BufferedReader br= null;
        BufferedWriter bw= null;

        try {
            fr=new FileReader("prac.txt");
            fw=new FileWriter("result.txt");
            br=new BufferedReader(fr);
            bw=new BufferedWriter(fw);
            String readLine;
            while((readLine=br.readLine())!=null){
                bw.write(readLine);
            }
            br.close();
            bw.close();
        }
        catch(IOException e) {}
    }
}
```

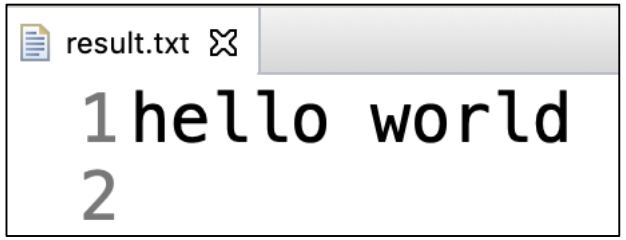
<실행 전>





<실행 후>





4. 객체 직렬화

직렬화?

: 자바 시스템 내부에서 사용하는 객체나 데이터를 외부의 시스템에서 사용할 수 있도록 byte 단위의 데이터로 변환시키는 기술과 바이트로 변환된 데이터를 다시 객체로 변환하는 기술

[1] 객체 전송의 단계

1) 직렬화된 객체를 바이트 단위로 분해한다(marshalling)
2) 직렬화 되어 분해된 데이터를 순서에 따라 전송
3) 전송받은 데이터를 복구(unmarshalling)

4. 객체 직렬화

[2] 마샬링(marshalling)

: 데이터를 바이트(byte) 단위의 데이터로 변환시키는 작업

```
class A implements Serializable{
                                                    …
        }

        A a = new A( );
```

- **마샬링을 적용할 수 있는 데이터**

- 기본 자료형: boolean, char, byte, short, int, long, float, double …
- Serializable 인터페이스를 구현한 클래스로 만들어진 객체

*** 사용할 클래스: ObjectOutputStream**

4. 객체 직렬화

[3] 직렬화 (Serializable)

: Serializable 인터페이스는 메서드 없이 JVM에게 정보 전달 기능만 존재

[4] 언마샬링(unmarshalling)

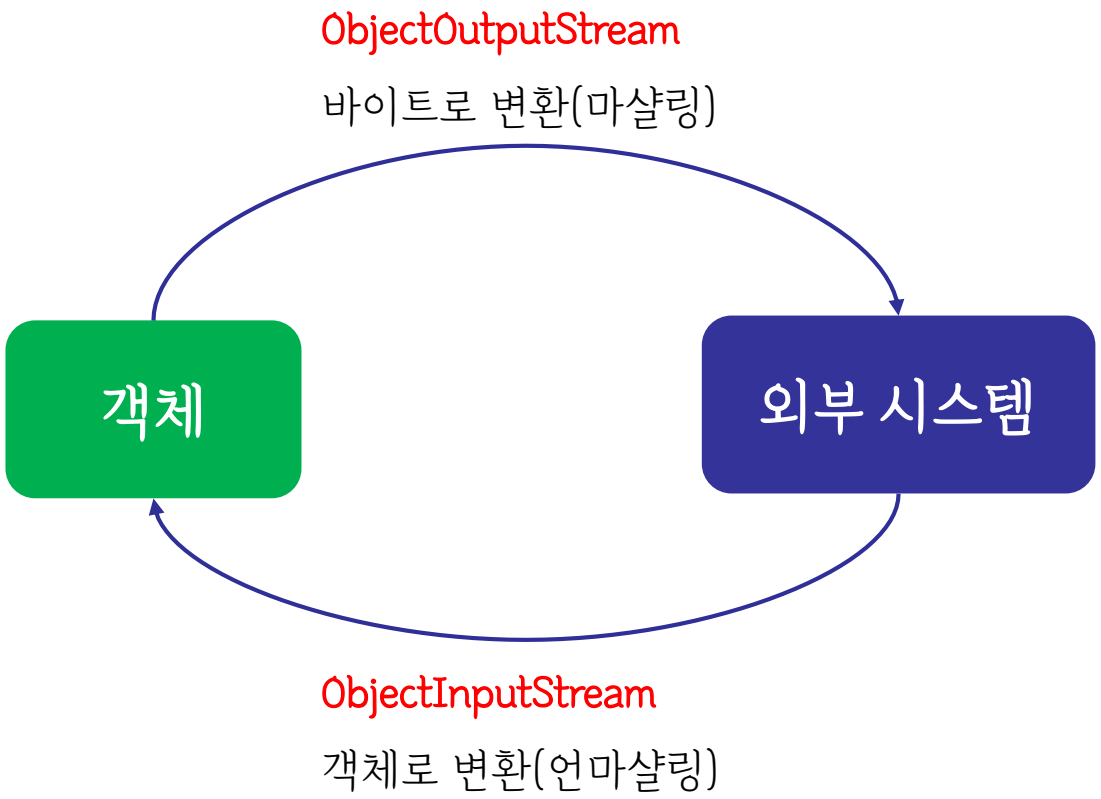
: 객체 스트림을 통해 전달된 바이트(Byte) 단위의 데이터를 원래의 객체로 복구하는 작업

* 언마샬링 시 주의할 점!
: 어떤 객체 형태로 복구할 지 형 변환을 정확하게 명시

* 사용할 클래스: ObjectInputStream

4. 객체 직렬화

[정리]


ObjectOutputStream
바이트로 변환(마샬링)
객체
외부 시스템
ObjectInputStream
객체로 변환(언마샬링)

4. 객체 직렬화

<실습> SerializableEx.java

User 클래스를 마샬링을 통해 바이트로 변환 하여, 파일에 저장하고,
파일에 저장된 객체를 언마샬링을 통해 객체의 정보를 프로그램에 출력하는 코드

마샬링 가능한 객체

```
class User implements Serializable{
    private String id;
    private  String pw;
    private String name;
    public User(String id, String pw, String name) {
        this.id = id;
        this.pw = pw;
        this.name = name;
    }
    @Override
    public String toString() {
        return "User [id=" + id + ", name=" + name + "]";
    }
}
```

```
직렬화 완료
User [id=test1234, name=가길동]
```

4. 객체 직렬화

<실습> SerializableEx.java

User 클래스를 마샬링을 통해 바이트로 변환 하여, 파일에 저장하고,
파일에 저장된 객체를 언마샬링을 통해 객체의 정보를 프로그램에 출력하는 코드

```
public static void marshalling() {
    try {
    FileOutputStream fos= new FileOutputStream("user.ser");
    BufferedOutputStream bos=new BufferedOutputStream(fos);
    ObjectOutputStream out = new ObjectOutputStream(bos);

    User u1=new User("test1234","1234","가길동");

    out.writeObject(u1);
    out.close();
    System.out.println("직렬화 완료");
}
    catch(Exception e) {
        e.printStackTrace();}
    }
```

ObjectOutputStream의 매개변수 생성자

ObjectOutputStream(OutputStream out)
: out 스트림에 마샬링한 객체 전달

out.writeObject(객체);

객체를 마샬링하여 바이트로 변환 후 버퍼스트림(bos)에 전달

<실행결과>

```
직렬화 완료
User [id=test1234, name=가길동]
```

4. 객체 직렬화

<실습> SerializableEx.java

User 클래스를 마샬링을 통해 바이트로 변환 하여, 파일에 저장하고,
파일에 저장된 객체를 언마샬링을 통해 객체의 정보를 프로그램에 출력하는 코드

```
public static void unmarshalling() {
    try {
        FileInputStream fis=new FileInputStream("user.ser");
        BufferedInputStream bis=new BufferedInputStream(fis);
        ObjectInputStream in=new ObjectInputStream(bis);

        User u1= (User)in.readObject();
        System.out.println(u1.toString());

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

}
```

ObjectInputStream의 매개변수 생성자
ObjectInputStream(InputStream in)
: 언마샬링을할 파일 연결

새로운객체= (클래스)in.readObject( );

- 정확한 타입 지정(형변환)
- 언마샬링된 객체를 읽어와 새로운 객체에 저장

<실행결과>

```
직렬화 완료
User [id=test1234, name=가길동]
```