입출력 스트림

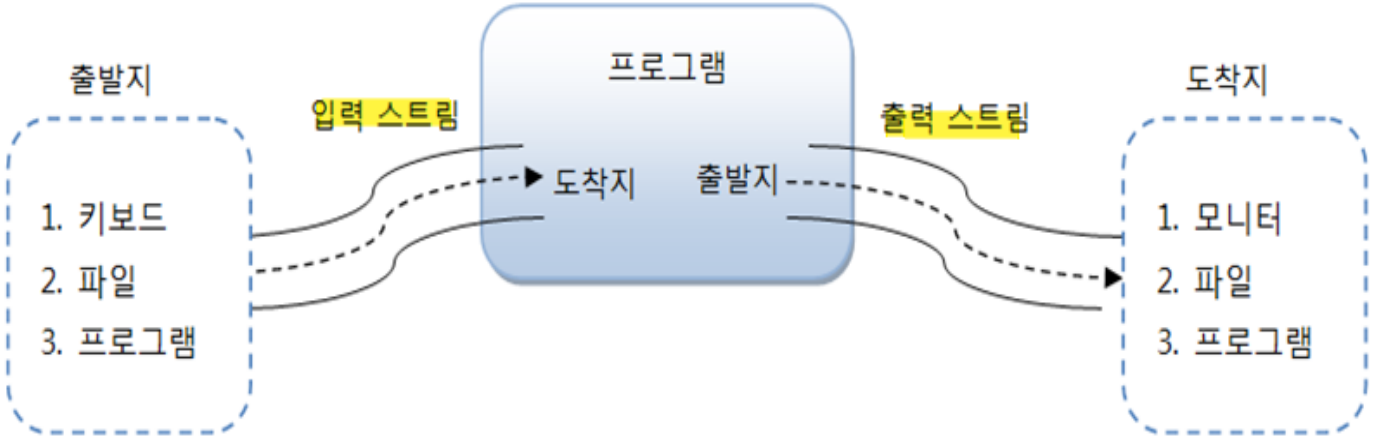
안 화 수

입출입 스트림

- 스트림(Stream)

  스트림은 단일 방향으로 연속적으로 흐르는 데이터의 흐름을 의미한다.

- 입력 스트림과 출력 스트림



입출입 스트림

❖ 입출력 스트림의 종류

- **바이트 기반 스트림**

  1Byte 단위로 데이터를 전송하는 스트림
  그림, 멀티미디어 등의 바이너리 데이터를 읽고 출력할 때 사용

- **문자 기반 스트림**

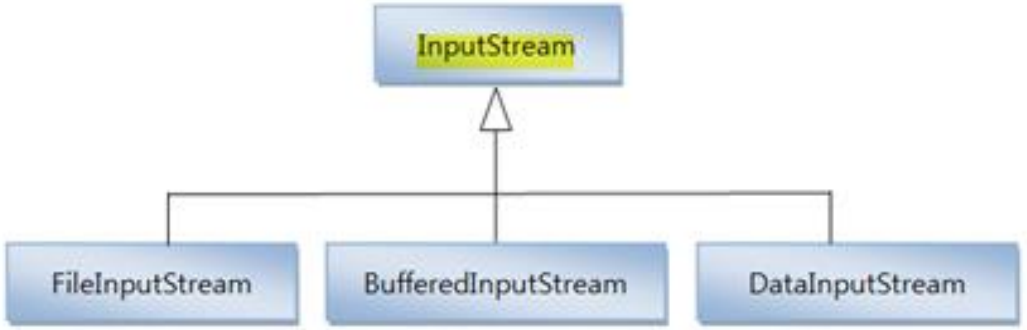
  2Byte 단위로 데이터를 전송하는 스트림 (한글 가능)
  문자 데이터를 읽고 출력할 때 사용

구분	바이트 기반 스트림		문자 기반 스트림	
	입력 스트림	출력 스트림	입력 스트림	출력 스트림
최상위 클래스	InputStream	OutputStream	Reader	Writer
하위 클래스 (예)	XXXputStream (FileInputStream)	XXXOutputStream (FileOutputStream)	XXXReader (FileReader)	XXXWriter (FileWriter)

입출입 스트림

- InputStream

  바이트 기반 입력 스트림의 최상위 클래스로 추상 클래스



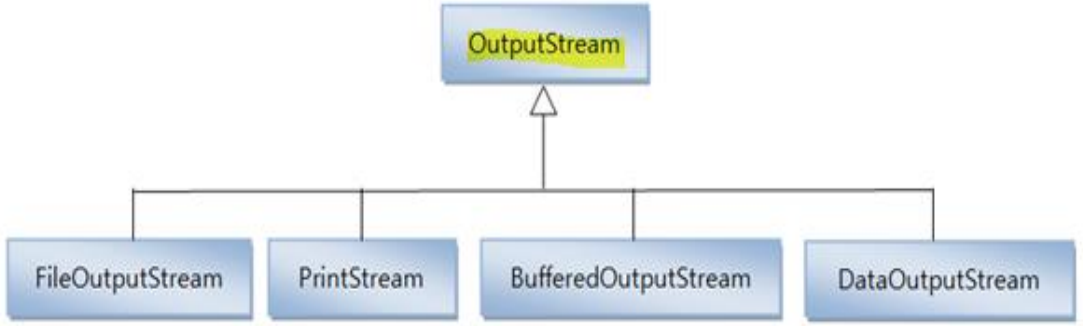
- InputStream 클래스의 주요 메소드

리턴타입	메소드	설명
int	read()	입력 스트림으로부터 1 바이트를 읽고 읽은 바이트를 리턴한다.
int	read(byte[] b)	입력 스트림으로부터 읽은 바이트들을 매개값으로 주어진 바이트 배열 b 에 저장하고 실제로 읽은 바이트 수를 리턴한다.
int	read(byte[] b, int off, int len)	입력 스트림으로부터 len 개의 바이트 만큼 읽고 매개값으로 주어진 바이트 배열 b[off] 부터 len 개까지 저장한다. 그리고 실제로 읽은 바이트 수인 len 개를 리턴한다. 만약 len 개를 모두 읽지 못하면 실제로 읽은 바이트 수를 리턴한다.
void	close()	사용한 시스템 자원을 반납하고 입력 스트림을 닫는다.

입출입 스트림

- OutputStream

  바이트 기반 출력 스트림의 최상위 클래스로 추상 클래스



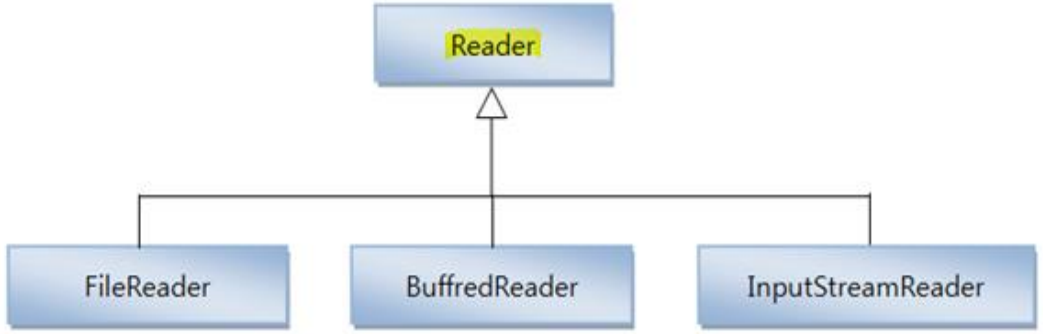
- OutputStream 클래스의 주요 메소드

리턴타입	메소드	설명
void	wrtie(int b)	출력 스트림으로 1 바이트를 보낸다.
void	write(byte[] b)	출력 스트림에 매개값으로 주어진 바이트 배열 b 의 모든 바이트를 보낸다.
void	write(byte[] b, int off, int len)	출력 스트림에 매개값으로 주어진 바이트 배열 b[off] 부터 len 개까지의 바이트를 보낸다.
void	flush()	버퍼에 잔류하는 모든 바이트를 출력한다.
void	close()	사용한 시스템 자원을 반납하고 출력 스트림을 닫는다.

입출입 스트림

- Reader

  문자 기반 입력 스트림의 최상위 클래스로 추상 클래스



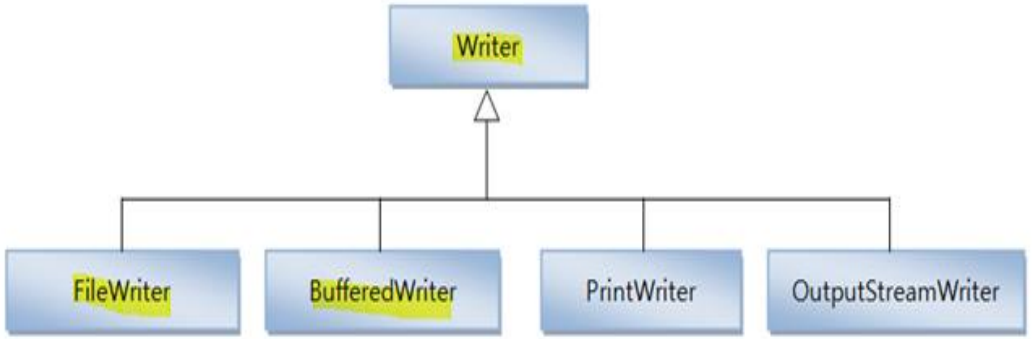
- Reader 클래스의 주요 메소드

메소드		설명
int	read()	입력 스트림으로부터 한개의 문자를 읽고 리턴한다.
int	read(char[] cbuf)	입력 스트림으로부터 읽은 문자들을 매개값으로 주어진 문자 배열 cbuf 에 저장하고 실제로 읽은 문자 수를 리턴한다.
int	read(char[] cbuf, int off, int len)	입력 스트림으로부터 len 개의 문자를 읽고 매개값으로 주어진 문자 배열 cbuf[off] 부터 len 개까지 저장한다. 그리고 실제로 읽은 문자 수인 len 개를 리턴한다.
void	close()	사용한 시스템 자원을 반납하고 입력 스트림을 닫는다.

입출입 스트림

- Writer

  문자 기반 출력 스트림의 최상위 클래스로 추상 클래스



- Writer 클래스의 주요 메소드

리턴타입	메소드	설명
void	wrtie(int c)	출력 스트림으로 매개값으로 주어진 한 문자를 보낸다.
void	write(char[] cbuf)	출력 스트림에 매개값으로 주어진 문자 배열 cbuf 의 모든 문자를 보낸다.
void	write(char[] cbuf, int off, int len)	출력 스트림에 매개값으로 주어진 문자 배열 cbuf[off] 부터 len 개까지의 문자를 보낸다.
void	write(String str)	출력 스트림에 매개값으로 주어진 문자열을 전부 보낸다.
void	write(String str, int off, int len)	출력 스트림에 매개값으로 주어진 문자열 off 순번부터 len 개까지의 문자를 보낸다.
void	flush()	버퍼에 잔류하는 모든 문자열을 출력한다.
void	close()	사용한 시스템 자원을 반납하고 출력 스트림을 닫는다.

입출입 스트림

- 입출력 스트림 예. InputStream

```
public class InputStreamTest {

    public static void main(String[] args) {
        InputStream is = System.in;
        int inputValue = 0;

        System.out.print("Input Data : ");

        try {
            // read() 메소드는 1바이트를 읽어와서 ascii 코드 값으로 casting
            inputValue = is.read();
        } catch (IOException io) {
            System.out.print(io.getMessage());
        }
        System.out.println("InputData is " + inputValue);
        System.out.println("InputData is " + (char) inputValue);
    }
}
```

입출입 스트림

- 입출력 스트림 예. InputStreamReader

```
public class InputStreamReaderTest {
    public static void main(String[] args) {

        InputStream is = System.in;
        InputStreamReader isr = new InputStreamReader(is);
//      InputStreamReader isr = new InputStreamReader(System.in);

        int inputValue = 0;
        System.out.print("Input Value : ");
        try {
            // 문자 스트림으로 2바이트를 읽어온다. 한글은 1자만 입력 받을 수 있다.
            inputValue = isr.read();    // 입력받은 값은 유니코드로 처리된다.
        } catch (IOException io) {
            System.out.print(io.getMessage());
        }
        System.out.println("Input Result : " + inputValue);
        System.out.println("Input Result : " + (char) inputValue);
    }
}
```

입출입 스트림

- 입출력 스트림 예. InputStreamReader

```
public class InputStreamReaderTest2 {
    public static void main(String[] args) {

        InputStream is = System.in;
        InputStreamReader isr = new InputStreamReader(is);

        int inputValue = 0;
        char[] temp = new char[10]; // 배열의 크기: 10

        System.out.print("Input Value : ");

        try {
            inputValue = isr.read(temp); // 키보드로 입력한 값은 temp배열에 저장된다.
        } catch (IOException io) {
            io.printStackTrace();
        }
        System.out.println("InputValue : " + inputValue);

        for (int i = 0; i < inputValue; i++) {
            System.out.print(temp[i]);
        }
        // char[]을 String type으로 변환
        System.out.println("char[] -> String : " + new String(temp));
    }
}
```

입출입 스트림

- 입출력 스트림 예. BufferedReader

```
public class BufferedReaderTest {
    public static void main(String[] args) {

        InputStream is = System.in;
        InputStreamReader isr = new InputStreamReader(is);
        BufferedReader br = new BufferedReader(isr);

        // BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

        System.out.print("Input Data : ");

        try {
            String inputString = br.readLine(); // 입력한 한줄을 모두 읽어온다.
            System.out.println();
            System.out.println("Output String = " + inputString);
        } catch (IOException io) {
            System.out.println(io.getMessage());
        }
    }
}
```

파일 입출력

- 파일 입출력
  - 바이트 스트림

```
FileInputStream  file = new FileInputStream("read.txt")
FileOutputStream file = new FileOutputStream("read1.txt")
```

  - 문자 스트림

```
FileReader  file = new FileReader("data.txt")
FileWriter  file = new FileWriter("data1.txt")
```

read.txt

Java program opens a stream

data.txt

자바 프로그램은 스트림을 열어요

파일 입출력

- 파일 입출력 예. FileInputStream

```
public class FileInputStreamTest {
    public static void main(String[] args) {

        int inputValue = 0;
        FileInputStream file = null;

        try {
            file = new FileInputStream("read.txt");

            // file의 끝을 만날 때까지 데이터를 읽어 온다. 파일의 끝을 만나면 -1을 반환함
            while ((inputValue = file.read()) != -1) {
                System.out.print((char) inputValue);
            }
        } catch (Exception e) {
            System.out.println(e.toString());
        } finally {
            try {
                if (file != null) file.close();
            } catch (Exception e) {}
        }
    }
}
```

파일 입출력

- 파일 입출력 예. FileReader

```
public class FileReaderTest {
    public static void main(String[] args) {

        FileReader file = null;
        int inputValue = 0;

        try {
            file = new FileReader("data.txt");

            // file의 끝을 만날 때까지 데이터를 읽어 온다. 파일의 끝을 만나면 -1을 반환함
            while ((inputValue = file.read()) != -1) {
                System.out.print((char) inputValue);
            }

            file.close();
        } catch (Exception e) {
            System.out.println(e.toString());
        }
    }
}
```

파일 입출력

- 파일 입출력 예. FileOutputStream

```
public class FileOutputStreamTest {
    public static void main(String[] args) {
        try {
            FileInputStream fis = new FileInputStream("read.txt");     // 원본 파일
            FileOutputStream fos = new FileOutputStream("read1.txt"); // 복사본 파일
            int input = 0;

            while ((input = fis.read()) != -1) {
                System.out.print((char) input);
                fos.write(input);   // read1.txt 파일에 저장
            }
            fis.close();   // 입력 스트림 닫기
            fos.close();   // 출력 스트림 닫기
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

파일 입출력

- 파일 입출력 예. FileWriter

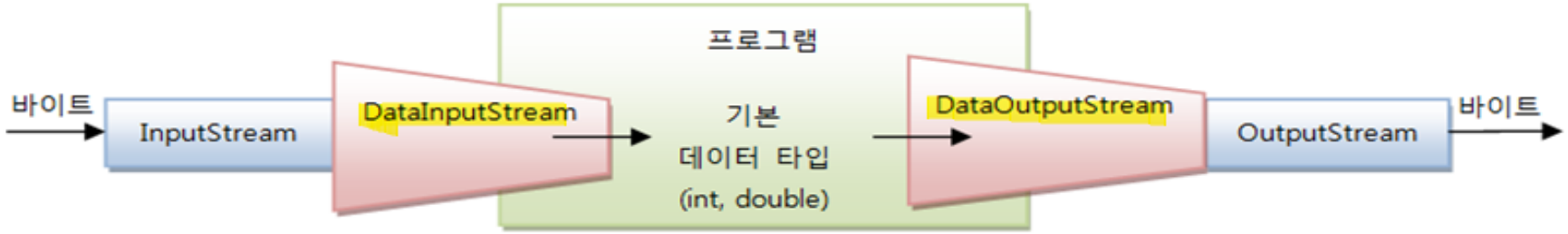
```
public class FileWriterTest {
    public static void main(String[] args) {
        try {
            FileReader fr = new FileReader("data.txt");  // 원본 파일
            FileWriter fw = new FileWriter("data1.txt"); // 복사본 파일
            int input = 0;

            while ((input = fr.read()) != -1) {
                System.out.print((char) input);
                fw.write(input);  // data1.txt 파일에 저장
            }
            fr.close();
            fw.close();
        } catch (IOException io) {
            System.out.println(io);
        }
    }
}
```

입출입 스트림

- 기본 타입 입출력 스트림 : DataInputStream, DataOutputStream

  기본 데이터 타입을 유지 하면서 입.출력을 처리하는 스트림



DataInputStream		DataOutputStream	
boolean	readBoolean()	void	writeBoolean(boolean v)
byte	readByte()	void	writeByte(int v)
char	readChar()	void	writeChar(int v)
double	readDouble()	void	writeDouble(double v)
float	readFloat()	void	writeFloat(float v)
int	readInt()	void	writeInt(int v)
long	readLong()	void	writeLong(long v)
short	readShort()	void	writeShort(int v)
String	readUTF()	void	writeUTF(String str)

입출입 스트림

- 입출력 스트림 예. DataInputStream, DataOutputStream

```
public class DataIOTest {
    public static void main(String[] args) {
        try {
            FileOutputStream fos = new FileOutputStream("iodata.txt");
            DataOutputStream dos = new DataOutputStream(fos);
            dos.writeBoolean(true);
            dos.writeChar('j');
            dos.writeInt(1234);
            dos.writeFloat(3.14F);
            dos.writeDouble(123.5423);
            dos.writeUTF("자바");

            FileInputStream fis = new FileInputStream("iodata.txt");
            DataInputStream dis = new DataInputStream(fis);
            System.out.println(dis.readBoolean());
            System.out.println(dis.readChar());
            System.out.println(dis.readInt());
            System.out.println(dis.readFloat());
            System.out.println(dis.readDouble());
            System.out.println(dis.readUTF());
            dis.close();
            dos.close();
        } catch (IOException io) {
            System.out.println(io.toString());
        }
    }
}
```

입출입 스트림

- 입출력 스트림 예. File

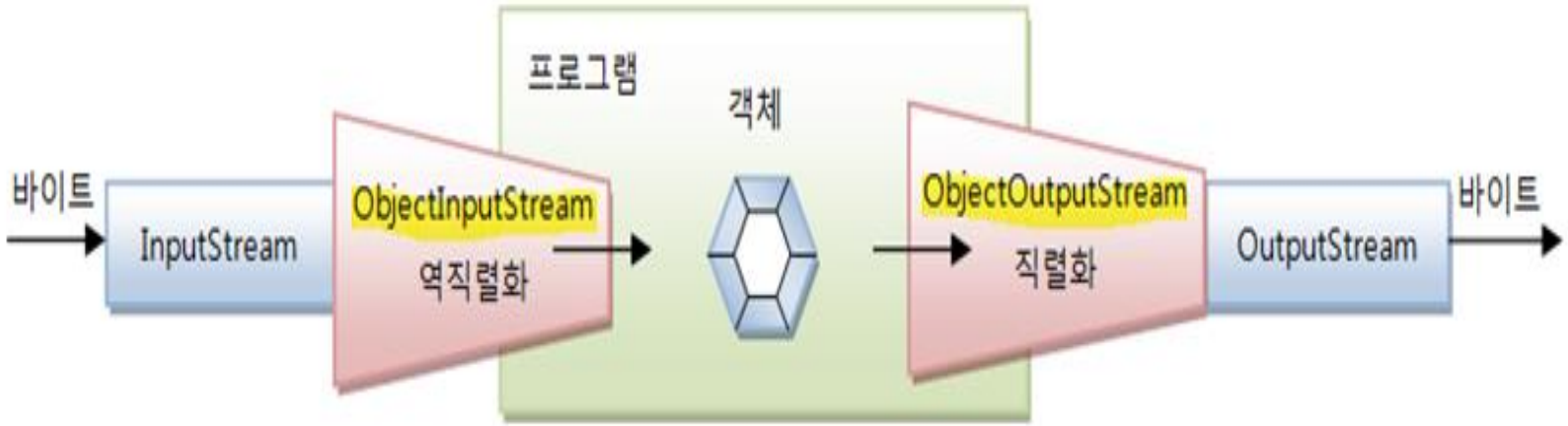
```
public class FileTest {
    public static void main(String[] args) {
        try {
            File temp = new File("c:/java01", "temp"); // c:/java01/temp 폴더 생성
            File tempFile = new File("test");          // test 폴더 생성

            // mkdirs()는 디렉토리를 생성하면 true를 반환함
            System.out.println("create directory state : " + temp.mkdirs());
            System.out.println("create directory state : " + tempFile.mkdirs());

            System.out.println("temp canRead : " + temp.canRead());
            System.out.println("temp canWrite : " + temp.canWrite());
            System.out.println("temp getAbsoluteFile : " + temp.getAbsoluteFile());
            System.out.println("temp getName : " + temp.getName());
            System.out.println("temp getParent : " + temp.getParent());
            System.out.println("temp getPath : " + temp.getPath());
            System.out.println("temp isDirectory : " + temp.isDirectory());
            System.out.println("temp isFile : " + temp.isFile());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

객체 직렬화

- 객체 직렬화
  1. 객체를 파일 또는 네트워크로 입출력할 수 있는 기능을 제공한다.
  2. 객체 직렬화를 지원하는 클래스 ObjectInputStream, ObjectOutputStream 가 있다.
  3. 객체 직렬화를 처리하기 위해서는 Serializable 인터페이스를 상속 받아서 구현할 수 있다. (객체가 중복으로 입출력하는 것을 방지 해준다.)



객체 직렬화

- 객체 직렬화 예. ObjectInputStream, ObjectOutputStream

```
public class PersonInformation implements Serializable {
    private String name;
    private int age;
    private String address;
    private String telephone;

    public PersonInformation(String name, int age, String address, String telephone) {
        this.name = name;
        this.age = age;
        this.address = address;
        this.telephone = telephone;
    }
    public String getName() {
        return name;
    }
    public int getAge() {
        return age;
    }
    public String getAddress() {
        return address;
    }
    public String getTelephone() {
        return telephone;
    }
}
```

객체 직렬화

- 객체 직렬화 예. ObjectInputStream, ObjectOutputStream (1/2)

```
public class ObjectStreamTest {

    PersonInformation gemini;
    PersonInformation johnharu;
    Date d;

    public ObjectStreamTest() {
        gemini = new PersonInformation("gemini", 10, "seoul", "02-321-3234");
        johnharu = new PersonInformation("johnharu", 20, "seoul", "02-473-4232");
        d = new Date();
    }
    public void writeObjectFile() {
        try {
            FileOutputStream fos = new FileOutputStream("person.dat");
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject(gemini);
            oos.writeObject(johnharu);
            oos.writeObject(d);
        } catch (Exception e) {
            System.out.println(e.toString());
        }
    }
```

객체 직렬화

- 객체 직렬화 예. ObjectInputStream, ObjectOutputStream (2/2)

```
    public void readObjectFile() {
        try {
            FileInputStream fis = new FileInputStream("person.dat");
            ObjectInputStream ois = new ObjectInputStream(fis);
            Object o = null;
            while ((o = ois.readObject()) != null) {
                // if( 변수 instanceof 클래스명)
                // : 변수가 이 클래스의 instance인지 아닌지를 체크함
                if (o instanceof PersonInformation) {
                    System.out.print(((PersonInformation) o).getName() + " : ");
                    System.out.print(((PersonInformation) o).getAge() + " : ");
                    System.out.print(((PersonInformation) o).getAddress() + " : ");
                    System.out.print(((PersonInformation) o).getTelephone());
                    System.out.println();
                } else {
                    System.out.println(o.toString());
                }
            }
        } catch (Exception e) {
        }
    }
    public static void main(String[] args) {
        ObjectStreamTest ost = new ObjectStreamTest();
        ost.writeObjectFile();
        ost.readObjectFile();
    }
}
```