입출력 I / O

FileInputStream

파일 입력을 받기 위한 스트림

- FileInputStream(String name)
- FileInputStream(File file)
- FileInputStream(FileDescriptor fdObj)

FileOutputStream

파일에 출력하기 위한 스트림

- FileOutputStream(String name)
- FileOutputStream(String name, Boolean append)
  Append는 기존에 추가하는 모드
- FileOutputStream(File file)
- FileOutputStream(File file, boolean append)
- FileOutputStream(FileDescriptor fdObj)

```
import java.io.FileInputStream;
import java.io.IOException;

public class FileViewer {

    public static void main(String[] args) throws IOException {
        FileInputStream fis = new FileInputStream(args[0]);
        int data = 0;

        while ((data = fis.read()) != -1) {
            char c = (char) data;
            System.out.print(c);
        }
    }
}
```

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

public class FileCopy {
   public static void main(String[] args) {
      try {
         FileInputStream fis = new FileInputStream(args[0]);
         FileOutputStream fos = new FileOutputStream(args[1]);

         int data = 0;
         while ((data = fis.read()) != -1) {
            fos.write(data);
         }

         fis.close();
         fis.close();
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
}
```

FilterInputStream

InputStream / OutputStream 을 포함하고 있음

FilterOutputStream

이것을 상속받아 기능을 추가한 InputStream과 OutputStream을 만들수 있음

BufferedInputStream, DataInputStream 등이 이 클래스를 상속받아 구현됨

BufferedInputStream

여러 바이트를 한꺼번에 입력받기위해 사용

- BufferedInputStream(InputStream in)
- BufferedInputStream(InputStream in, int size)

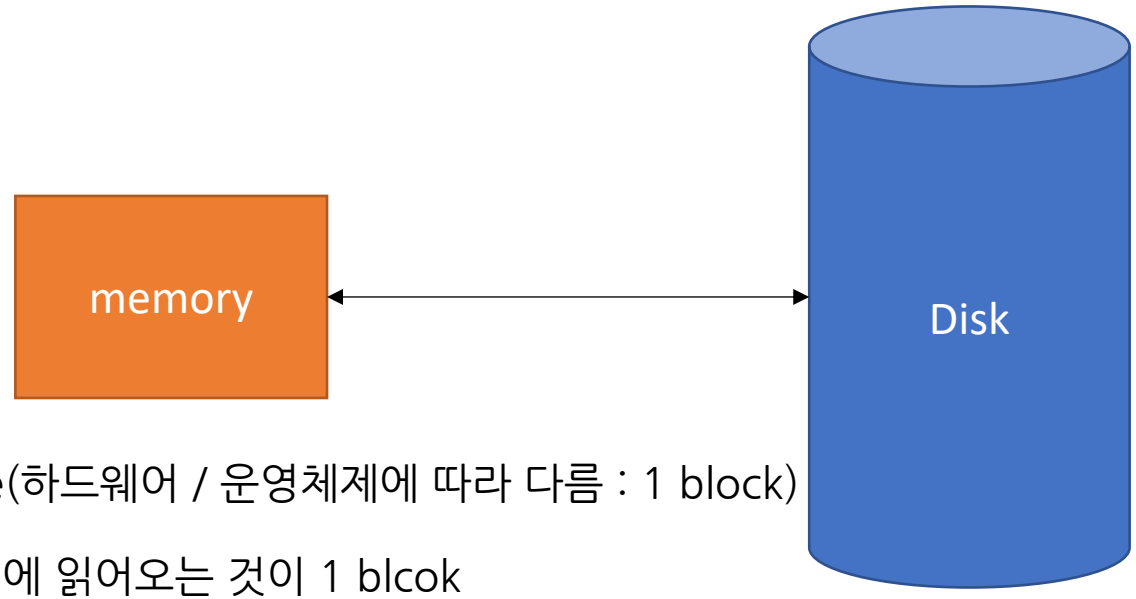
BufferedOutputStream

파일에 출력하기 위한 스트림

- BufferedOutputStream(OutputStream out)
- BufferedOutputStream(OutputStream out, int size)
- flush()
- Close()

버퍼하는 이유

한 바이트씩 읽어오는 것이 너무 느리다.



1byte ~ 512byte(하드웨어 / 운영체제에 따라 다름 : 1 block) 동일

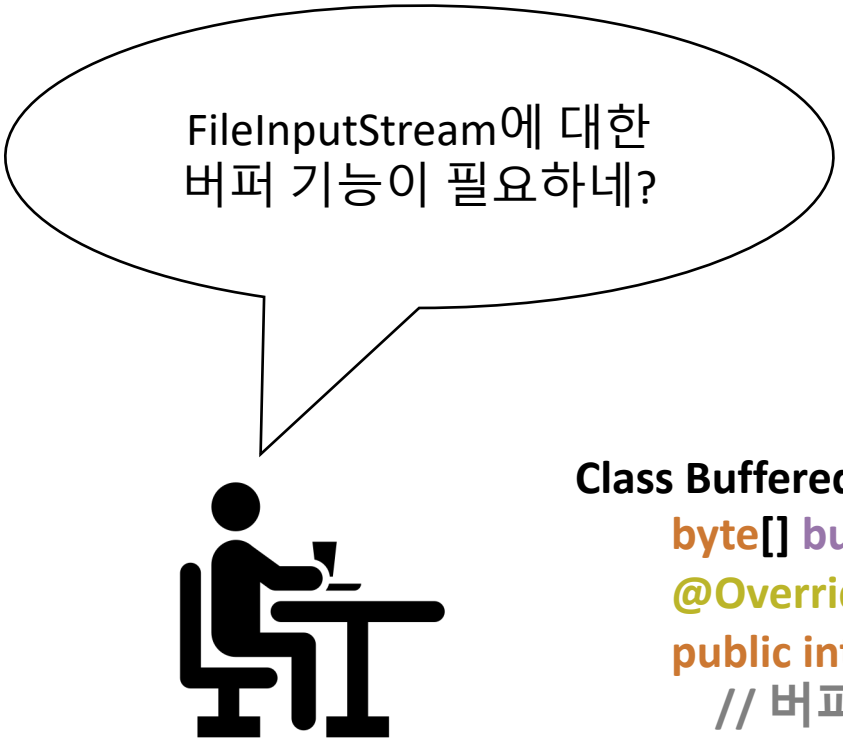
-> 한번에 읽어오는 것이 1 blcok

원래 한 칸 읽던 것을

-> System.*arraycopy*(buffer, 0, nbuf, 0, pos);

로 변경

어떻게?




```
Class BufferedFileInputStream extends FileInputStream {
    byte[] buffer;
    @Override
    public int read(byte[] b) {
      // 버퍼에 한번에 저장해놓기

    }
```

어떻게?

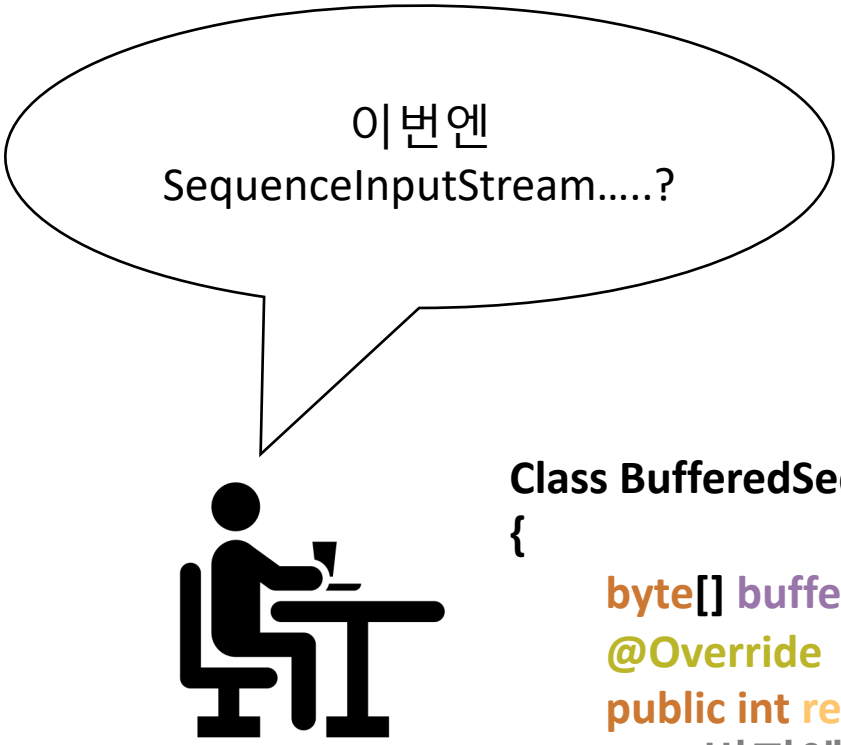
이번엔 ObjectInputStream 에 대한 버퍼 기능이 필요하네?

```
Class BufferedObjectInputStream extends ObjectInputStream {
    byte[] buffer;
    @Override
    public int read(byte[] b) {
      // 버퍼에 한번에 저장해놓기

    }
```

이전에 만든 BufferedFileInputStream 내용 복붙하니 되는군!

어떻게?




```
Class BufferedSequenceInputStream extends SequenceInputStream
{
        byte[] buffer;
        @Override
        public int read(byte[] b) {
           // 버퍼에 한번에 저장해놓기

        }
```

어떻게?

모든 InputStream마다 상속받아서 Buffered 클래스 작성?

```
byte[] buffer;
@Override
public int read(byte[] b) {
   // 읽기
}
```

문제

1. **InputStream이 추가될 때마다 새로운 클래스 작성**
2. **Buffered의 요구사항이 바뀔 때마다 모두 수정**

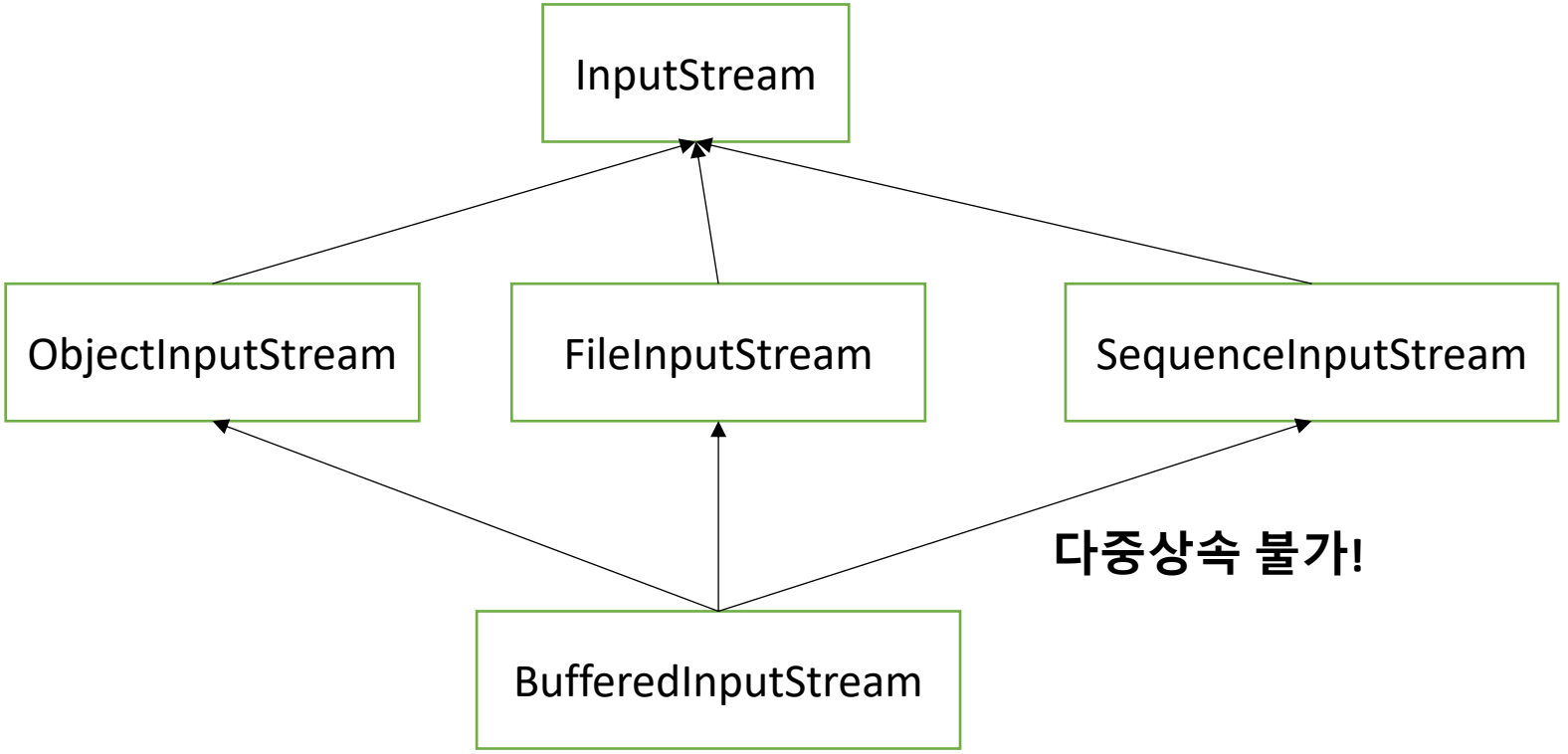
ByteArrayInputStream
FileInputStream
ObjectInputStream
PipedInputStream
SequenceInputStream

BufferedByteArrayInputStream
BufferedFileInputStream
BufferedObjectInputStream
BufferedPipedInputStream
BufferedSequenceInputStream

어떻게?

요구사항

- **버퍼 기능이 있을 것**
- **다른 InputStream에 추가적인 기능으로 동작할 수 있을 것**
- **앞으로 추가되는 InputStream에도 적용될 수 있을 것**
- **스스로도 InputStream 일것**



어떻게?

버퍼 기능이 있을 것

```
byte[] buffer;

public read() {
        //한 번에 읽어 buffer에 저장
}
```

어떻게?

스스로도 InputStream 일것
InputStream들에 손쉽게 적용할 수 있을 것
앞으로 추가되는 InputStream에도 적용될 수 있을 것

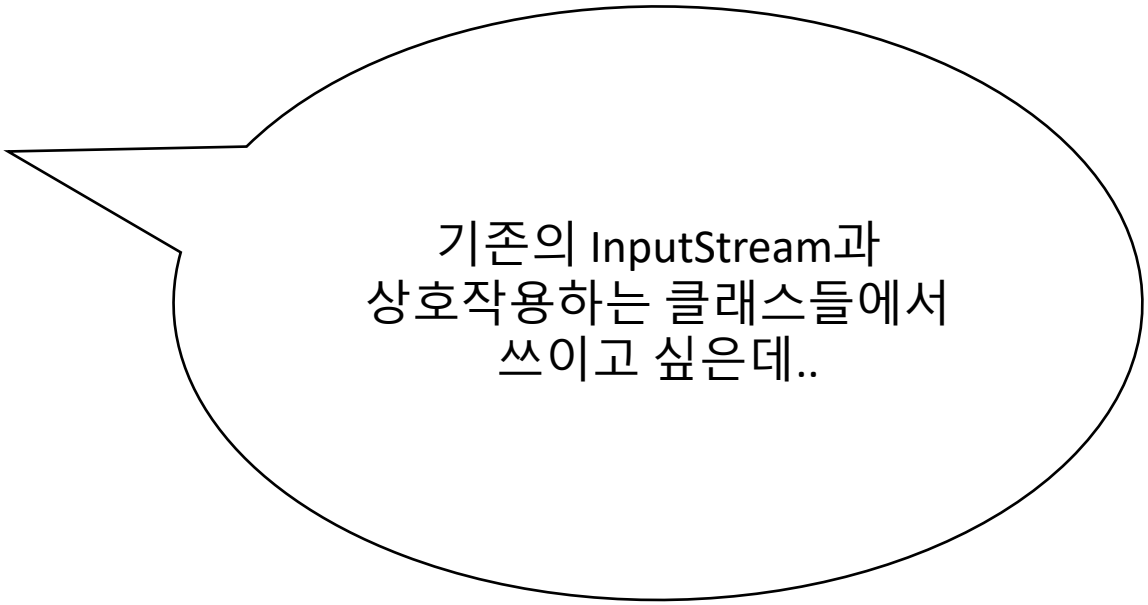
```
public class BufferedInputStream {
   byte[] buffer;
   InputStream inputStream;

   public BufferedInputStream(InputStream inputStream) {
      this.inputStream = inputStream;
   }

   public read() {
      // 한 바이트를 요청했지만, 한번에 읽어서 buffer에 저장함
      inputStream.read(buffer);
   }
}
```

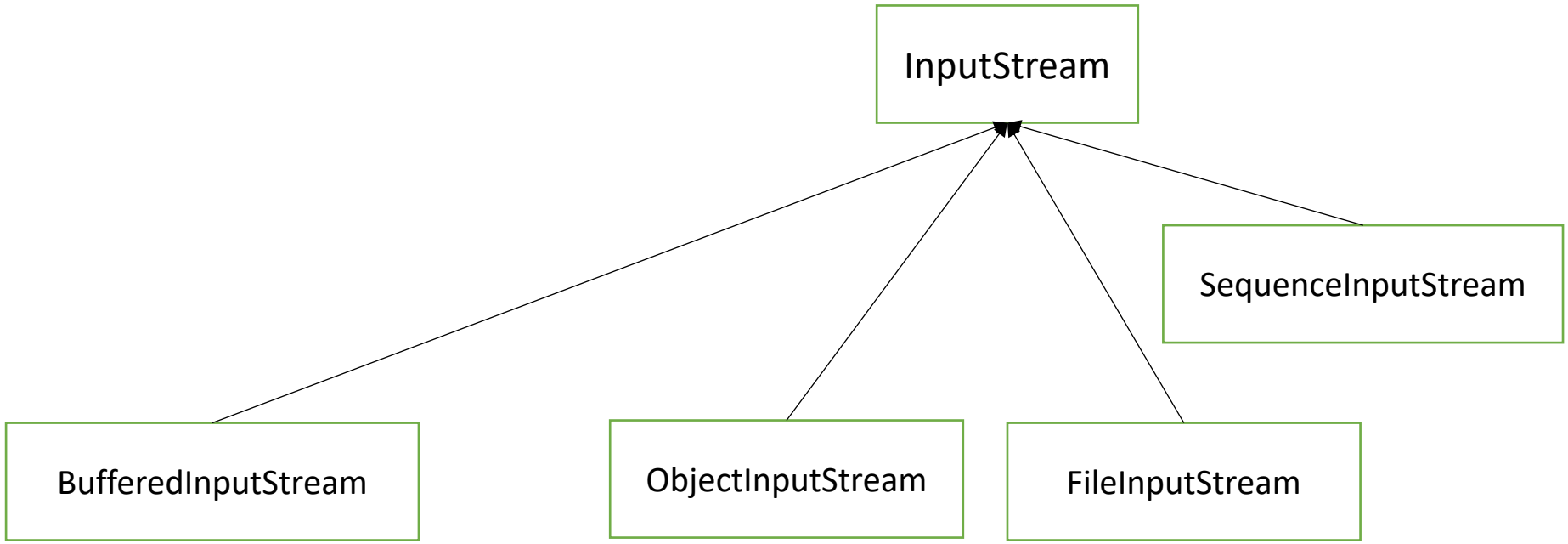
어떻게?

•**Scanner**
public Scanner(InputStream source)



어떻게?

스스로도 InputStream 일것



어떻게?

스스로도 InputStream 일것
InputStream들에 손쉽게 적용할 수 있을 것
앞으로 추가되는 InputStream에도 적용될 수 있을 것
스스로도 InputStream 일것

```
public class BufferedInputStream extends InputStream {
   byte[] buffer;
   InputStream inputStream;

   public BufferedInputStream(InputStream inputStream) {
      this.inputStream = inputStream;
   }

   public read() {
      // logic
      inputStream.read(buffer);
   }
}
```

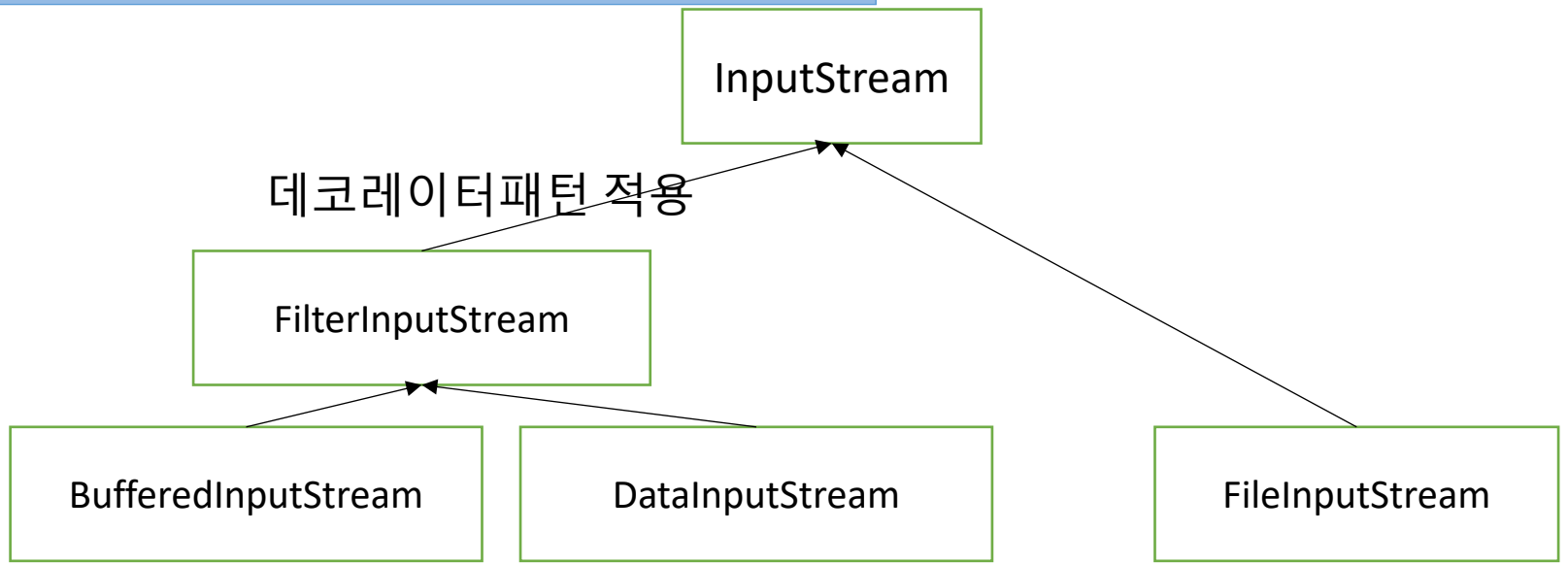
데코레이터 패턴

추가적인 기능을 확장하는 방법!

FilteredInputStream

```
public class FilterInputStream extends InputStream {
        InputStream inputStream;
}
```

버퍼 외의 다른 기능도 추가하고 싶음



```java
import java.io.BufferedOutputStream;
import java.io.FileOutputStream;
import java.io.FilterInputStream;
import java.io.IOException;

public class BufferedOutputStreamEx1 {

   public static void main(String[] args) {
      try {
         FileOutputStream fos = new FileOutputStream("123.txt");
         BufferedOutputStream bos = new BufferedOutputStream(fos, 5);
         for (int i = '1'; i <= '9'; i++) {
            bos.write(i);
         }
         fos.close();
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
}
```

DataInputStream

DataOutputStream

기본 자료형을 읽고 쓸수 있게함

- readBoolean()
- readByte()
- readChar()
- readInt()
- readFloat()
- readDouble()
- readUnsignedByte()
- Write…

```java
import java.io.DataInputStream;
import java.io.FileInputStream;
import java.io.IOException;

public class DataInputStreamEx1 {

    public static void main(String[] args) {
        try {
            FileInputStream fis = new FileInputStream("sample.dat");
            DataInputStream dis = new DataInputStream(fis);

            System.out.println(dis.readInt());
            System.out.println(dis.readFloat());
            System.out.println(dis.readBoolean());
            dis.close();

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

```java
public class DataInputStreamEx2 {
    public static void main(String[] args) {
        int sum = 0;
        int score = 0;

        FileInputStream fis = null;
        DataInputStream dis = null;

        try {
            fis = new FileInputStream("score.dat");
            dis = new DataInputStream(fis);

            while (true) {
                score = dis.readInt();
                System.out.println(score);
                sum += score;
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                if (dis != null) { dis.close(); }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

```java
import java.io.DataInputStream;
import java.io.FileInputStream;
import java.io.IOException;

public class DataInputStreamEx3 {
    public static void main(String[] args) {
        int sum = 0;
        int score = 0;

        try (FileInputStream fis = new FileInputStream("score.dat");
             DataInputStream dis = new DataInputStream(fis);) {

            while (true) {
                score = dis.readInt();
                System.out.println(score);
                sum += score;
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```java
public class DataOutputStream {

   public static void main(String[] args) {
      ByteArrayOutputStream bos = null;
      java.io.DataOutputStream dos = null;

      byte[] result = null;

      try {
         bos = new ByteArrayOutputStream();
         dos = new java.io.DataOutputStream(bos);

         dos.writeInt(10);
         dos.writeFloat(20.0f);
         dos.writeBoolean(true);

         result = bos.toByteArray();
         String[] hex = new String[result.length];
         for (int i = 0; i < result.length; i++) {
            if (result[i] < 0) { hex[i] = String.format("%02x", result[i] + 256); } else {
               hex[i] = String.format("%02x", result[i]);
            }
         }
         System.out.println("Arrays.toString(result) = " + Arrays.toString(result));
         System.out.println("Arrays.toString(hex) = " + Arrays.toString(hex));
         dos.close();
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
}
```

```java
import java.io.DataOutputStream;
import java.io.FileOutputStream;
import java.io.IOException;

public class DataOutputStreamEx1 {
    public static void main(String[] args) {
        FileOutputStream fos = null;
        DataOutputStream dos = null;

        try {
            fos = new FileOutputStream("sample.dat");
            dos = new DataOutputStream(fos);
            dos.writeInt(10);
            dos.writeFloat(20.0f);
            dos.writeBoolean(true);
            dos.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```java
import java.io.DataOutputStream;
import java.io.FileOutputStream;
import java.io.IOException;

public class DataOutputStreamEx3 {
    public static void main(String[] args) {
        int[] score = { 100, 90, 95, 85, 50 };
        try {
            FileOutputStream fos = new FileOutputStream("score.dat");
            DataOutputStream dos = new DataOutputStream(fos);

            for (int i = 0; i < score.length; i++) {
                dos.writeInt(score[i]);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

SequenceInputStream

여러개의 InputStream을 연속적으로 연결

```
public class SequentialInputStreamEx {
   public static void main(String[] args) {
      byte[] arr1 = {0, 1, 2};
      byte[] arr2 = {3, 4, 5};
      byte[] arr3 = {6, 7, 8};
      byte[] outSrc = null;

      Vector v = new Vector();
      v.add(new ByteArrayInputStream(arr1));
      v.add(new ByteArrayInputStream(arr2));
      v.add(new ByteArrayInputStream(arr3));

      SequenceInputStream input = new SequenceInputStream(v.elements());
      ByteArrayOutputStream output = new ByteArrayOutputStream();

      int data = 0;
      try {
         while((data = input.read()) != -1) {
            output.write(data);
         }
      } catch (IOException e) {}

      outSrc = output.toByteArray();

      System.out.println("Arrays.toString(arr1) = " + Arrays.toString(arr1));
      System.out.println("Arrays.toString(arr2) = " + Arrays.toString(arr2));
      System.out.println("Arrays.toString(arr3) = " + Arrays.toString(arr3));
      System.out.println("Arrays.toString(outSrc) = " + Arrays.toString(outSrc));
   }
}
```