Chapter 13

02204271

Java IO

Introduction

- I/O Stream นี่แทนความหมายของ Input source และ Output
- Stream เป็นเหมือน ”ท่อ”ส่งข้อมูลจากต้นทาง (source) ไปยังปลายทาง (sink)
- source เป็นตำแหน่งเริ่มต้นของstream เรียกว่าinput stream
- sink เป็นตำแหน่งสินสุดของstream เรียกว่าoutput stream

Introduction

- source หรือsink อาจเป็นฮาร์ดแวร์หรือซอฟต์แวร์เช่นไฟล์หน่วยความจำหรือ socket เป็นต้น

- ภาษาจาวาแบ่งstream ออกเป็น 2 ชนิด
  - Byte stream
  - Character stream

- คำว่าstream โดยทั่วไปจะหมายถึ งbyte stream

- reader และwriter จะหมายถึง character stream

Package java.io

- คลาสที่เกี่ยวกับอินพุตและเอาท์พุตจะถูกกำหนดโดยJava APIในแพคเก็จ java.io ซึ่งจะมีคลาสพื้นฐานอยู่4 คลาสคือ
  - InputStream เป็นคลาสที่ใช้ในการสร้างออปเจ็คที่เป็น stream ในการรับชนิดข้อมูลแบบ byte
  - Output Streamเป็นคลาสที่ใช้ในการสร้างออปเจ็คที่เป็น stream ในการรับชนิดข้อมูลแบบ byte
  - Reader เป็นคลาสที่ใช้ในการสร้างออปเจ็คที่เป็น stream ในการรับชนิดข้อมูลแบบ char
  - Writer เป็นคลาสที่ใช้ในการสร้างออปเจ็คที่เป็น stream ในการรับชนิดข้อมูลแบบ char

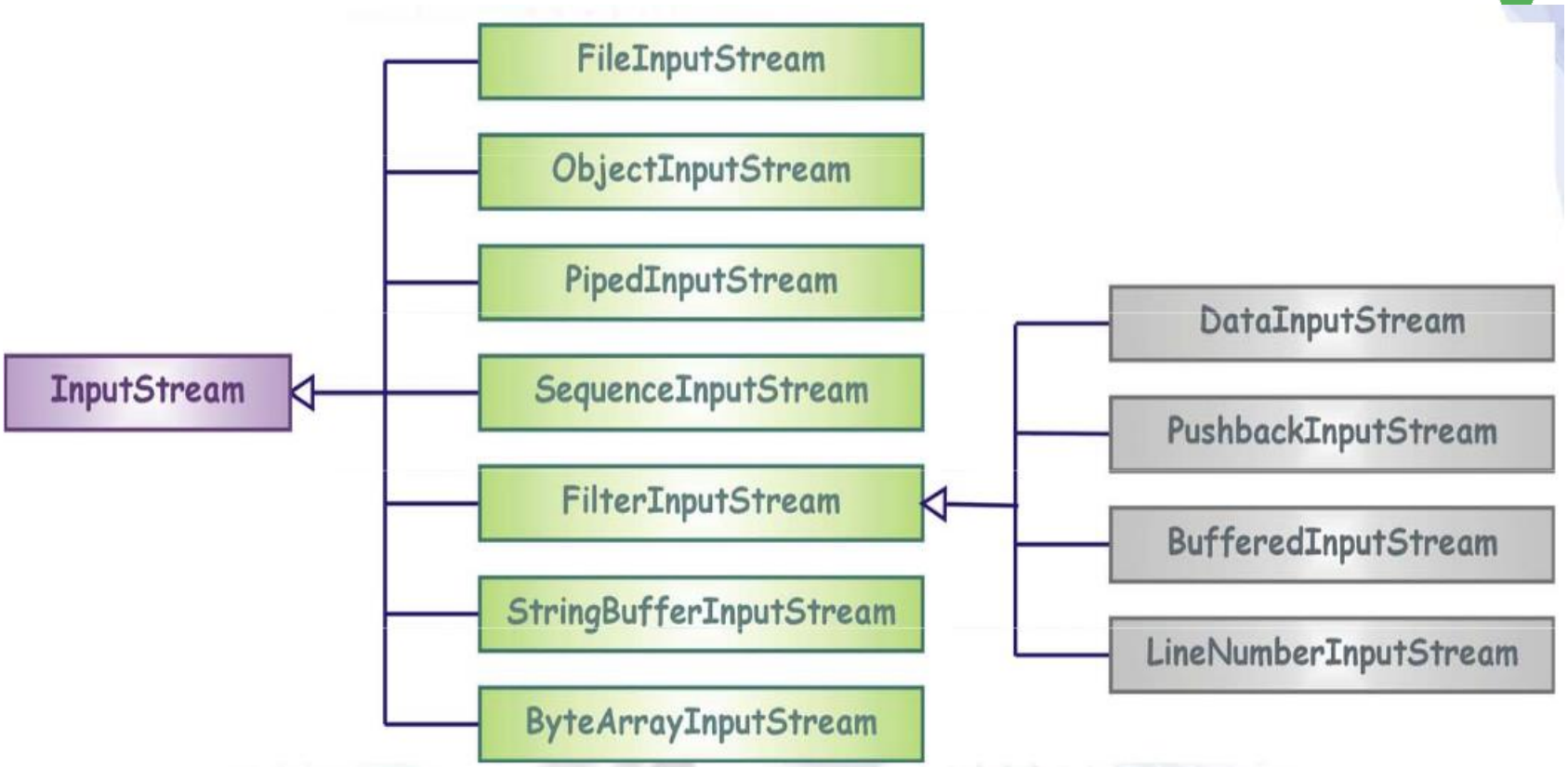
File

- คลาสที่ชื่อ File เป็นคลาสที่อยู่ในแพคเก็จ java.io โดยเป็นคลาสที่ใช้ในการสร้างออปเจ็คที่เป็นไฟล์หรือไดเร็กทอรี่
- คลาส File จะมีเมธอดในการจัดการกับไฟล์หรือไดเร็กทอรี่และเมธอดในการสืบค้นข้อมูลต่างๆอยู่หลายเมธอด
- ออปเจ็คของคลาสFileจะสร้างมาจากconstructor ที่มีรูปแบบดังนี้
  - public File(String name)
  - public File(String dir, String name)
  - public File(File dir, String name)

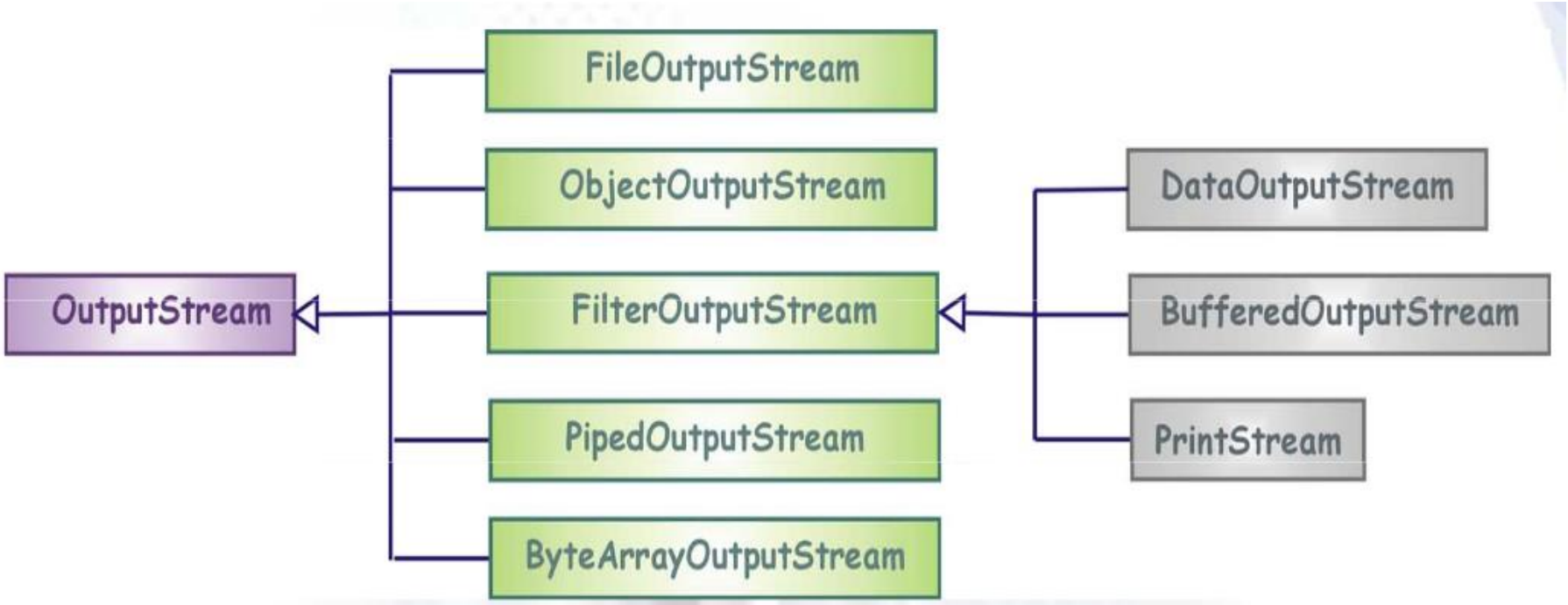
File (Cont.)

- เมธอดของคลาสFileที่ใช้ในการสืบค้นข้อมูลหรือจัดการกับไฟล์ที่สำคัญมีดังนี้
  - boolean exists()
  - boolean isFile()
  - boolean isDirectory()
  - String getName()
  - String []list()
  - boolean canWrite()
  - boolean mkdir()
  - boolean renameTo(File newName)

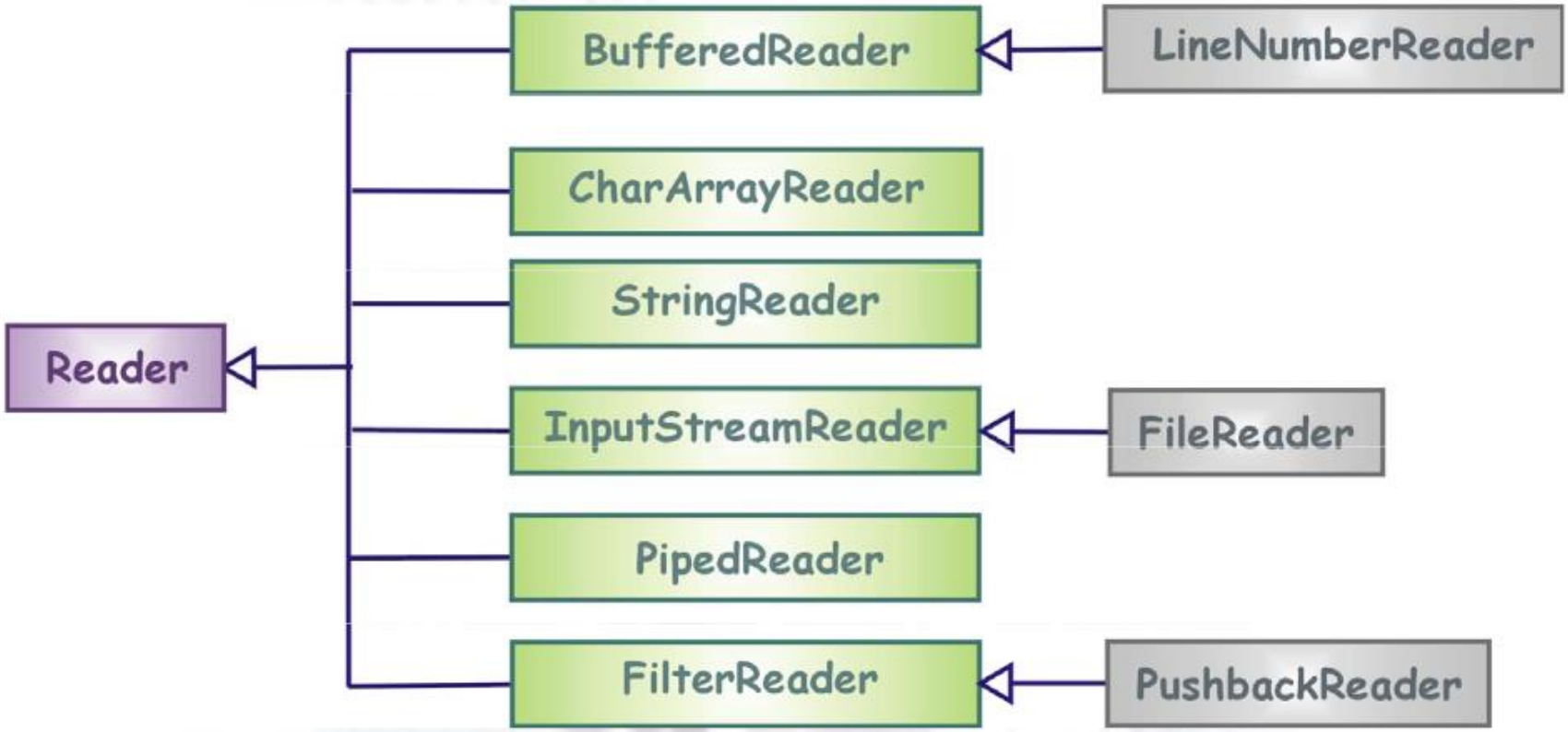
InputStream Class



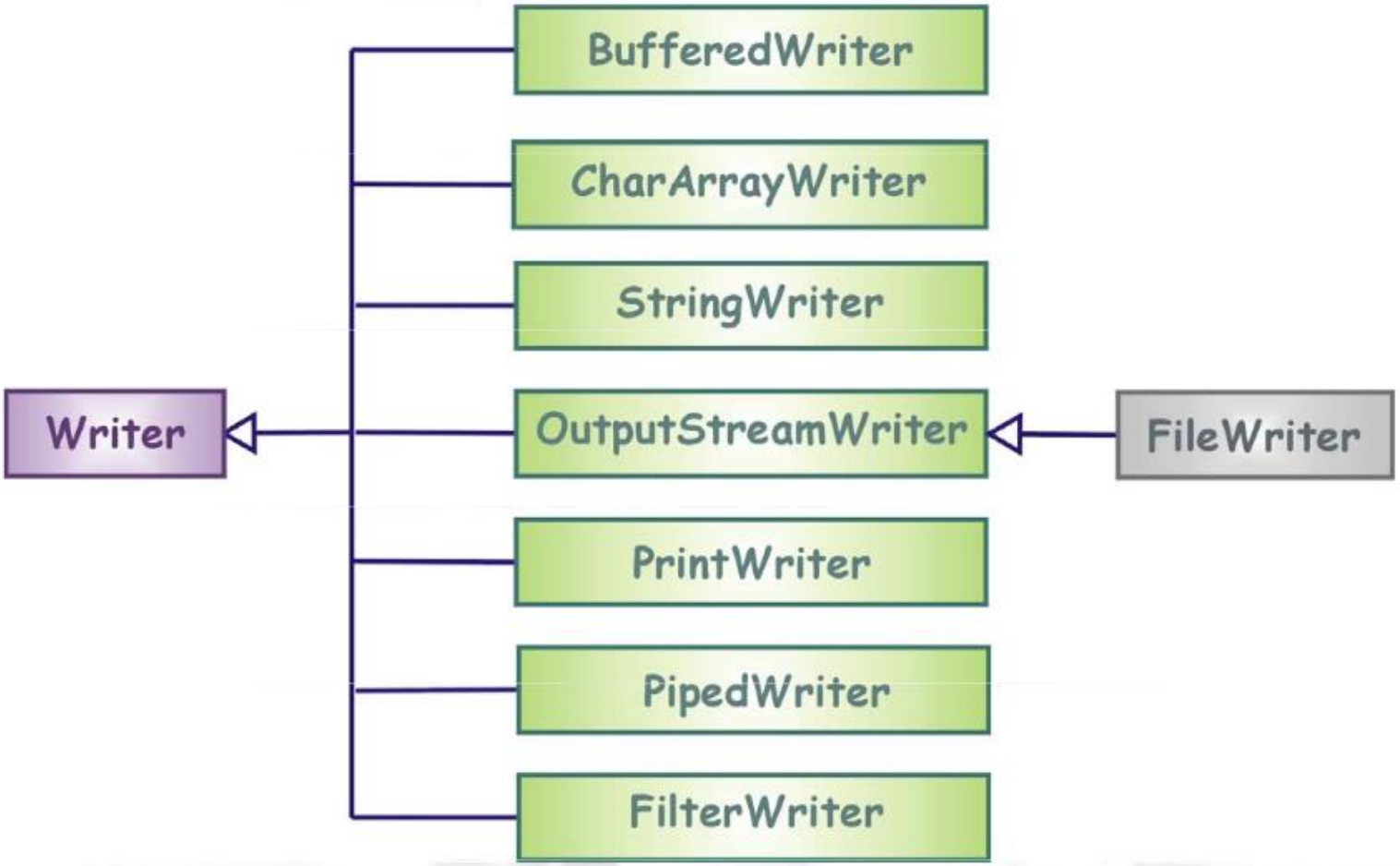
OutputStream Class



Reader Class



Writer Class



ByteStream

- ภาษาจาวาจะมีคลาสพื้นฐานในการจัดการกับอินพุตและเอาท์พุตที่เป็นชนิดข้อมูลแบบ byte อยู่สองคลาสที่คู่กันคือ InputStream และ OutputStream

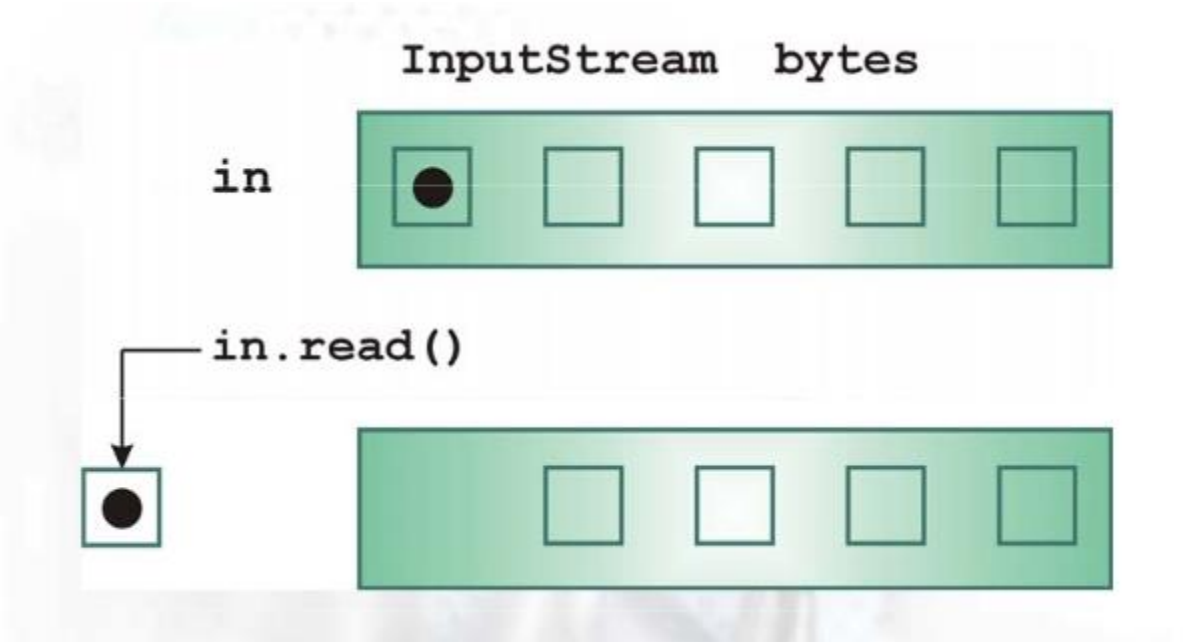
- คลาสทั้งสองเป็นคลาสแบบabstractซึ่งเราไม่สามารถที่จะสร้างออปเจ็คของคลาสทั้งสองได้แต่คลาสทั้งสองจะมีคลาสที่เป็นsubclass ซึ่งจะใช้ในการสร้างออปเจ็คสำหรับการรับและส่งข้อมูลแบบbyteของโหนดที่มีต้นทางและปลายทางแบบต่างๆอาทิเช่น
  - FileInputStream และ FileOutputStream
  - ByteArrayInputStream และ ByteArrayOutputStream

InputStream

- คลาส InputStream จะใช้ในการอ่านข้อมูลของ stream ที่เป็นชนิดข้อมูลแบบ byte

- คลาสInputStreamจะนำข้อมูลจากโหนดต้นทางเข้ามาในstream และการอ่านข้อมูลจากstream จะเป็นการลบข้อมูลที่อ่านออกจากstream โดยมีเมธอดที่ใช้สำหรับการอ่านข้อมูลที่เป็น byte หรืออะเรย์ของ byteเท่านั้นดังนี้
  - int read()
  - int read(byte []b)
  - int read(byte []b,int offset,int length)

InputStream (Cont.)



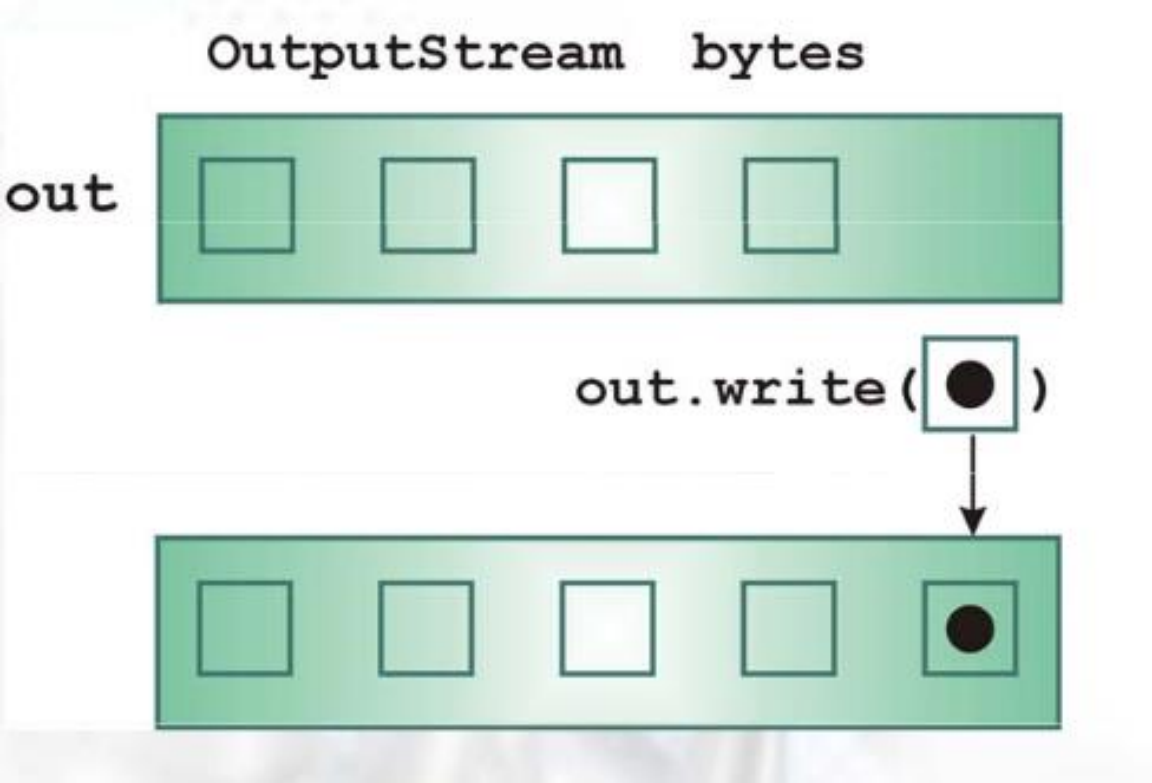
Example

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

public class InputStreamTest {

    public static void main(String[] args) throws IOException  {
        File file = new File("D:/javaTest.txt");
        InputStream inputStream = new FileInputStream(file);

        int content;
        while ((content = inputStream.read()) != -1) {
            // convert to char and display it
            System.out.print((char) content);
        }
    }
}
```

OutputStream

- คลาส OutputStream จะใช้การส่งข้อมูลของ stream ที่เป็นชนิดข้อมูลแบบ byte การส่งข้อมูลของออบเจ็คชนิด Outputstream จะเป็นการเพิ่มข้อมูลลงในstream
- คลาส OutputStreamจะมีเมธอดในการส่งข้อมูลชนิดbyteที่สอดคล้องกับเมธอดread()ในคลาสInputStreamโดยคลาสนี้จะมีเมธอด write()ที่เป็นเมธอดแบบabstractในรูปแบบต่างๆดังนี้
  - void write(int c)
  - void write(byte []b)
  - void write(byte []b,int offset,int length)

OutputStream (Cont.)



Example

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

public class OutputStreamTest {

    public static void main(String[] args) throws IOException {
        File file = new File("D:/javaTestOutput.txt");
        OutputStream outputStream = new FileOutputStream(file);
        String text = "Hello World output !";
        byte[] textByte = text.getBytes();
        outputStream.write(textByte);
        outputStream.close();
    }
}
```

Example (Cont.)

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

public class InputToOutputStream {
    public static void main(String[] args) throws IOException {
        File fileInput = new File("D:/javaTest.txt");
        InputStream inputStream = new FileInputStream(fileInput);
        String text = "";
        int content;
        while ((content = inputStream.read()) != -1) {
            text += (char)content;
        }

        File fileOutput = new File("D:/javaTestOutput.txt");
        OutputStream outputStream = new FileOutputStream(fileOutput);
        byte[] textByte = text.getBytes();
        outputStream.write(textByte);
        outputStream.close();
    }
}
```

Character Stream

- ภาษาจาวากำหนดคลาสพื้นฐานในการจัดการกับอินพุตและเอาท์พุตที่เป็นชนิดข้อมูลแบบ char อยู่สองคลาสคือ Reader และ Writer

- คลาสทั้งสองเป็นคลาสแบบabstractโดยมีsubclass ที่สืบทอดมาเพื่อใช้ในการสร้างออปเจ็คสำหรับจัดการกับโหนดต้นทางและปลายทางในรูปแบบต่างๆเช่นไฟล์ หน่วยความจำ และ ไบท์ เป็นต้น

Reader

- Readerเป็นคลาสที่ใช้ในการอ่านข้อมูลของstream ที่เป็นชนิดข้อมูลแบบchar

- Readerจะมีเมธอดที่เหมือนกับคลาส InputStream และมีหลักการทำงานที่สอดคล้องกันแต่จะรับข้อมูลหรือ argument ที่เป็นชนิดข้อมูลchar โดยมีเมธอดต่างๆดังนี้
  - int read()
  - int read(char []c)
  - int read(char []c,int offset,int length)
  - void close()

Example

```
import java.io.FileReader;
import java.io.IOException;
import java.io.Reader;

public class FileReaderTest {

    public static void main(String[] args) throws IOException {
        Reader reader = new FileReader("D:/javaTest.txt");
        int data;
        while ((data = reader.read()) != -1) {
            System.out.print((char) data);
        }
        reader.close();
    }
}
```

Writer

- Writer เป็นคลาสที่ใช้ในการเขียนข้อมูลของ stream ที่เป็นชนิดข้อมูลแบบchar

- Write จะมีเมธอดที่เหมือนกับคลาส OutputStream และมีหลักการทำงานที่สอดคล้องกันแต่จะรับข้อมูลหรือ argument ที่เป็นชนิดข้อมูลchar โดยมีเมธอดต่างๆ ดังนี้
  - int write()
  - int write(char []c)
  - int write(char []c,int offset,int length)
  - void close()
  - Void flush()

Writer (Cont.)

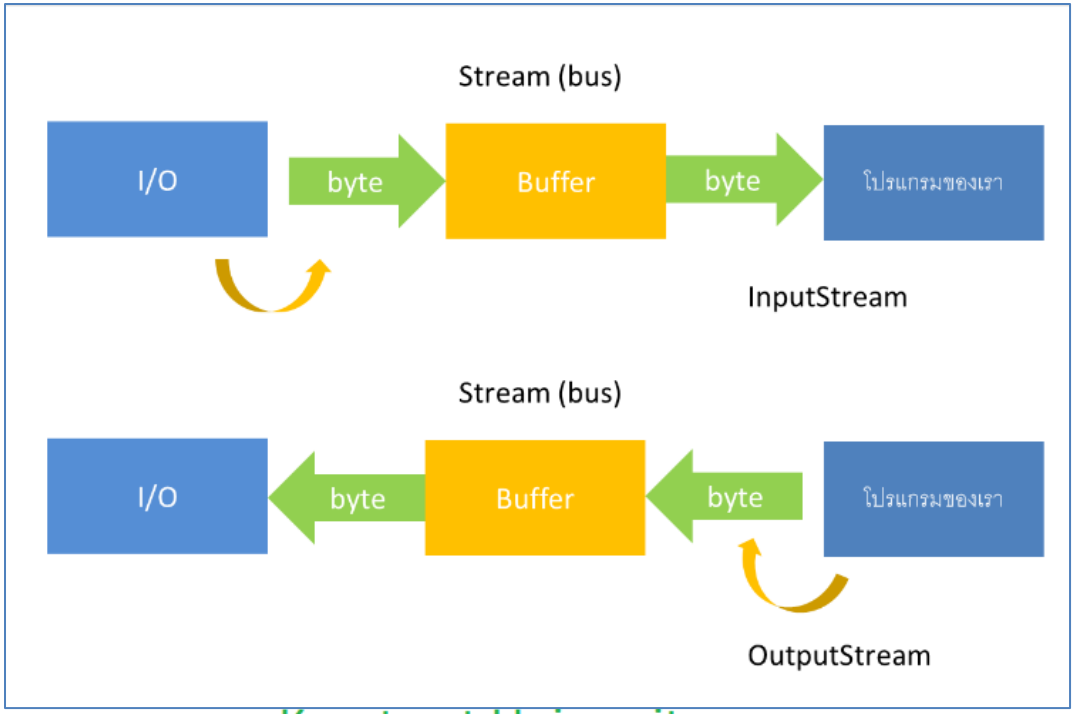
- เมธอดเพิ่มเติมเพื่อเขียนชนิดข้อมูลที่เป็นString
  - int write(String s)
  - int write(String s,int offset,int length)

Example

```
import java.io.FileWriter;
import java.io.IOException;
import java.io.Writer;

public class FileWriterTest {

    public static void main(String[] args) throws IOException {
        String text = "Hello World Output!";
        Writer writer =  new FileWriter("D:/javaTestOutput.txt");
        writer.write(text);
        writer.close();
    }
}
```

Buffered Stream

- Buffered Stream คือ การอ่าน/เขียนข้อมูลลงใน Buffer ของ Stream นั้น ๆ ก่อน แล้วจึงส่งไปยังปลายทางอีกที เหมาะสำหรับการเขียน/อ่านไฟล์ที่มีขนาดใหญ่



Buffered Stream (Buffer Reader)

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.io.Reader;

public class BufferReaderTest {

    public static void main(String[] args) throws IOException {
        Reader reader = new FileReader("D:/javaTestLarge.txt");
        BufferedReader bufferedReader = new BufferedReader(reader);
        String data;
        while ((data = bufferedReader.readLine()) != null) {
            System.out.println(data);
        }
        bufferedReader.close();
        reader.close();
    }
}
```

Buffered Stream (Buffer Writer)

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

public class BufferWriterTest {
    public static void main(String[] args) throws IOException{
        String text = "Hello Buffer Writer!";
        File file = new File("D:/javaBufferWriter.txt");
        FileWriter fw = new FileWriter(file);
        BufferedWriter bw = new BufferedWriter(fw);
        bw.write(text);
        bw.close();
    }
}
```

ObjectOutputStream

- Java object Serialization is an API provided by Java Library stack as a means to serialize Java objects.

- Serialization is a process to convert objects into a writable byte stream. Once converted into a byte-stream, these objects can be written to a file

Example

```
public class Person implements Serializable {

        private static final long serialVersionUID = 1L;
        private String name;
        private int age;
        private String gender;

        Person() {
        };

        Person(String name, int age, String gender) {
                this.name = name;
                this.age = age;
                this.gender = gender;
        }

        @Override
        public String toString() {
                return "Name:" + name + "\nAge: " + age + "\nGender: " + gender;
        }
}
```

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

        Person p1 = new Person("John", 30, "Male");
        Person p2 = new Person("Rachel", 25, "Female");

        try {
                FileOutputStream f = new FileOutputStream(new File("myObjects.txt"));
                ObjectOutputStream o = new ObjectOutputStream(f);

                // Write objects to file
                o.writeObject(p1);
                o.writeObject(p2);

                o.close();
                f.close();

            FileInputStream fi = new FileInputStream(new File("myObjects.txt"));
            ObjectInputStream oi = new ObjectInputStream(fi);

            // Read objects
            Person pr1 = (Person) oi.readObject();
            Person pr2 = (Person) oi.readObject();

            System.out.println(pr1.toString());
            System.out.println(pr2.toString());

            oi.close();
            fi.close();

    } catch (FileNotFoundException e) {
            System.out.println("File not found");
    } catch (IOException e) {
            System.out.println("Error initializing stream");
    } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
```

```
Name:John
Age: 30
Gender: Male
Name:Rachel
Age: 25
Gender: Female
```

Appending ObjectOutputStream

```
public class AppendingObjectOutputStream extends ObjectOutputStream {

  public AppendingObjectOutputStream(OutputStream out) throws IOException {
    super(out);
  }

  @Override
  protected void writeStreamHeader() throws IOException {
    // do not write a header, but reset:
    // this line added after another question
    // showed a problem with the original
    reset();
  }

}
```

AppendingObjectOutputStream

```
Person[] p = { new Person("Test_1", 10,"male"),
        new Person("Test_2", 20,"female") };

File f = new File("DataObject.txt");
boolean exist = f.exists();
FileOutputStream fw = new FileOutputStream(f,true);
ObjectOutputStream oos = null;
if(!f.exists()) {
     oos = new ObjectOutputStream(fw)
    ;
}else {
    oos = new AppendingObjectOutputStream(fw) ;
}
for(Person pp : p) {
    oos.writeObject(pp);

}
fw.close();
```

AppendingObjectOutputStream

```
FileInputStream fi = new FileInputStream("DataObject.txt");
ObjectInputStream ois = new ObjectInputStream(fi);

while (true) {

    try {
        Person p1 = (Person) ois.readObject();
        System.out.println(p1);
    } catch (EOFException | ClassNotFoundException e) {

        break;
    }
}
```