บทที่ 9

คลาสอินพุตและเอาท์พุต

(Input/Output Classes)

วัตถุประสงค์

- อธิบายความหมายและประเภทของ stream
- อธิบายองค์ประกอบที่สำคัญของแพคเก็จ `java.io`
- แนะนำคลาสและเมธอดของคลาส `InputStream` และ `OutputStream`
- แนะนำคลาสและเมธอดของคลาส `Reader` และ `Writer`
- อธิบายความแตกต่างระหว่างคลาส `Reader` และ `Writer` กับ stream
- อธิบายการสร้างและใช้ stream แบบต่าง ๆ

วัตถุประสงค์

- แนะนำคลาสและเมธอดของคลาส `File`
- อธิบายการอ่าน เขียนและอัพเดทข้อมูลของคลาส `RandomAccessFile`
- อธิบายการใช้อินเตอร์เฟสที่ชื่อ `Serializable`
- อธิบายวิธีการเขียนและอ่านข้อมูลของออปเจ็คผ่าน stream

Stream

- เปรียบเสมือนท่อส่งข้อมูลจากต้นทาง (source) ไปยังปลายทาง (sink)
- stream ช่วยทำให้ผู้เขียนโปรแกรม ไม่จำเป็นต้องทราบรายละเอียดการติดต่อกับฮาร์ดแวร์หรือซอฟต์แวร์ที่ใช้ในการส่งข้อมูล เพียงแต่ต้องรู้เมธอดที่ใช้ในการรับหรือส่งข้อมูล
- source เป็นตำแหน่งเริ่มต้นของ stream เรียกว่า input stream
- sink เป็นตำแหน่งสิ้นสุดของ stream เรียกว่า output stream

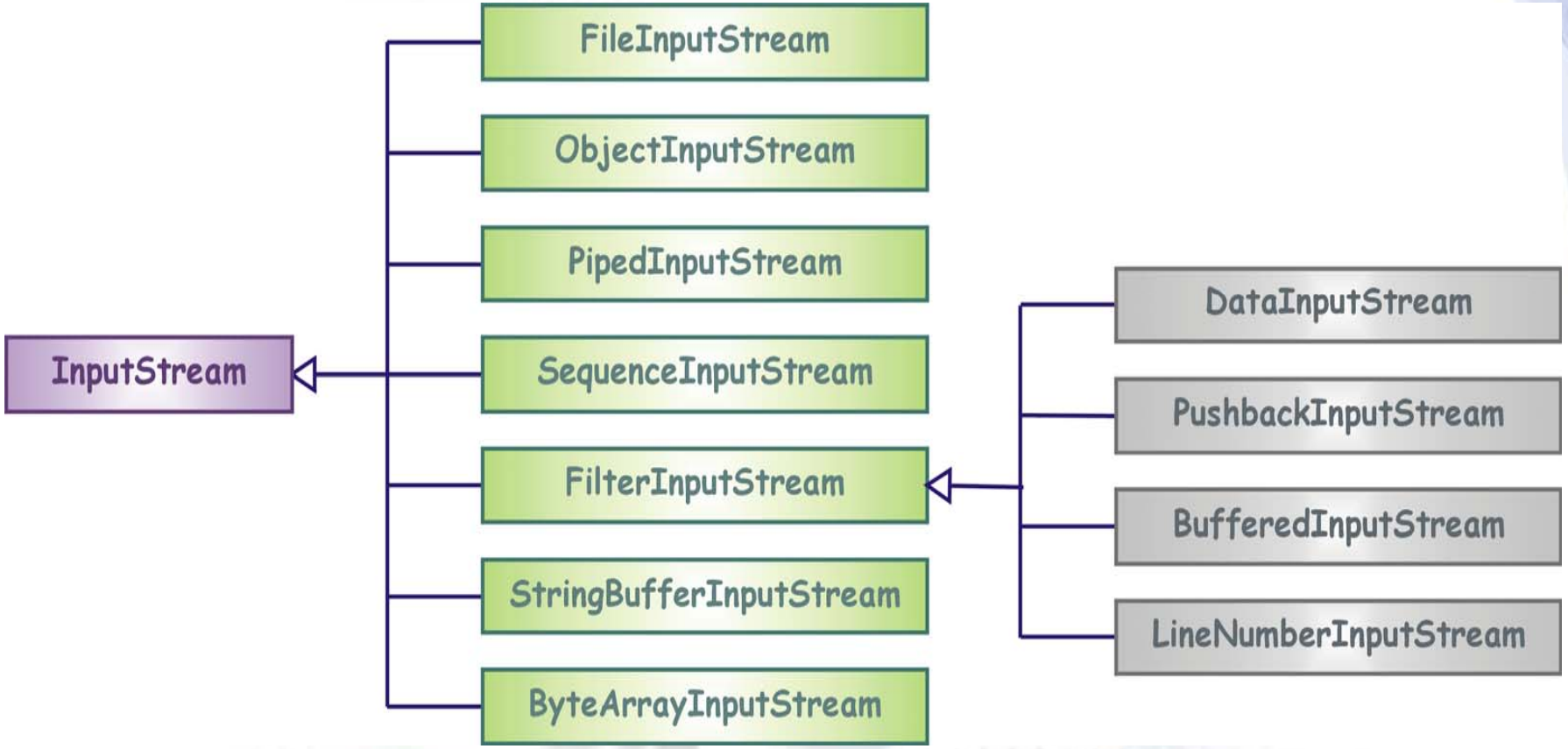
Stream

- source หรือ sink อาจเป็นฮาร์ดแวร์หรือซอฟต์แวร์ เช่น ไฟล์ หน่วยความจำ หรือ socket เป็นต้น
- ภาษาจาวาแบ่ง stream ออกเป็น
  - byte stream
  - character stream
- คำว่า stream โดยทั่วไปจะหมายถึง byte stream
- reader และ writer จะหมายถึง character stream

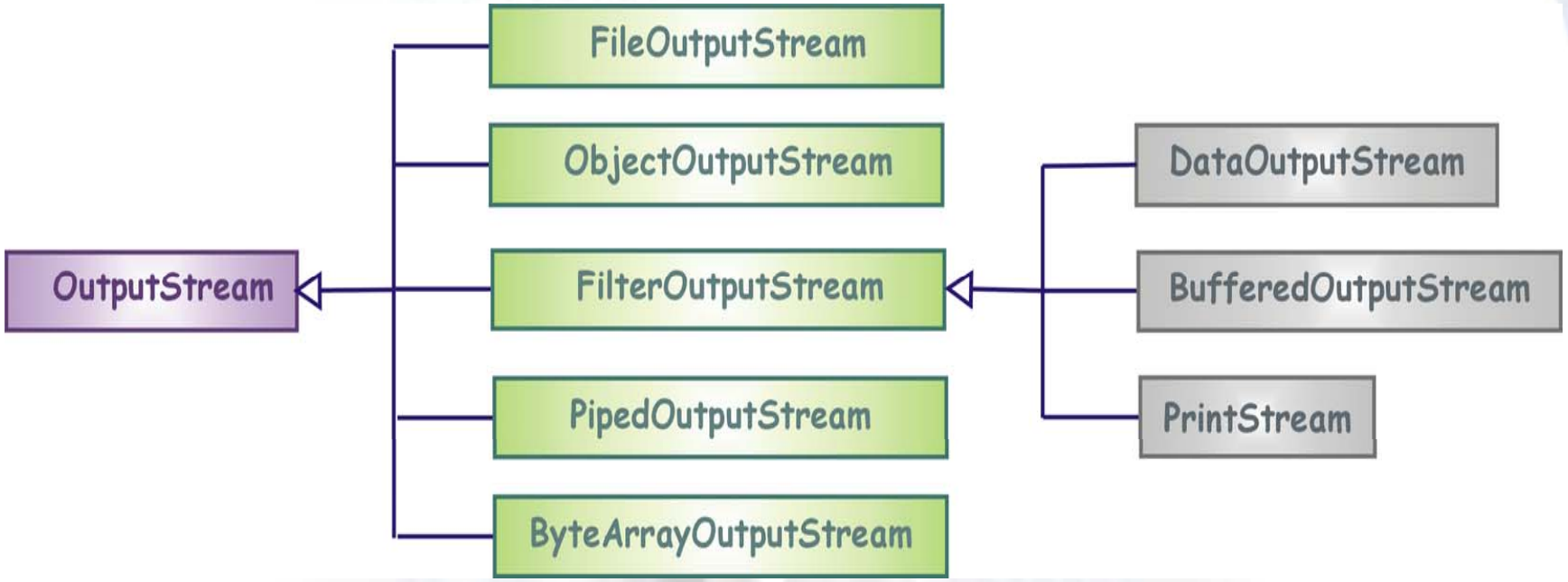
แพคเก็จ java.io

- คลาสที่เกี่ยวกับอินพุตและเอาท์พุตจะถูกกำหนดโดย Java API ในแพคเก็จ java.io ซึ่งจะมีคลาสพื้นฐานอยู่ 4 คลาสคือ
  - **InputStream**
    เป็นคลาสที่ใช้ในการสร้างออปเจ็คที่เป็น stream ในการรับชนิดข้อมูลแบบ byte
  - **OutputStream**
    เป็นคลาสที่ใช้ในการสร้างออปเจ็คที่เป็น stream ในการส่งชนิดข้อมูลแบบ byte
  - **Reader**
    เป็นคลาสที่ใช้ในการสร้างออปเจ็คที่เป็น stream ในการรับชนิดข้อมูลแบบ char
  - **Writer**
    เป็นคลาสที่ใช้ในการสร้างออปเจ็คที่เป็น stream ในการส่งชนิดข้อมูลแบบ char

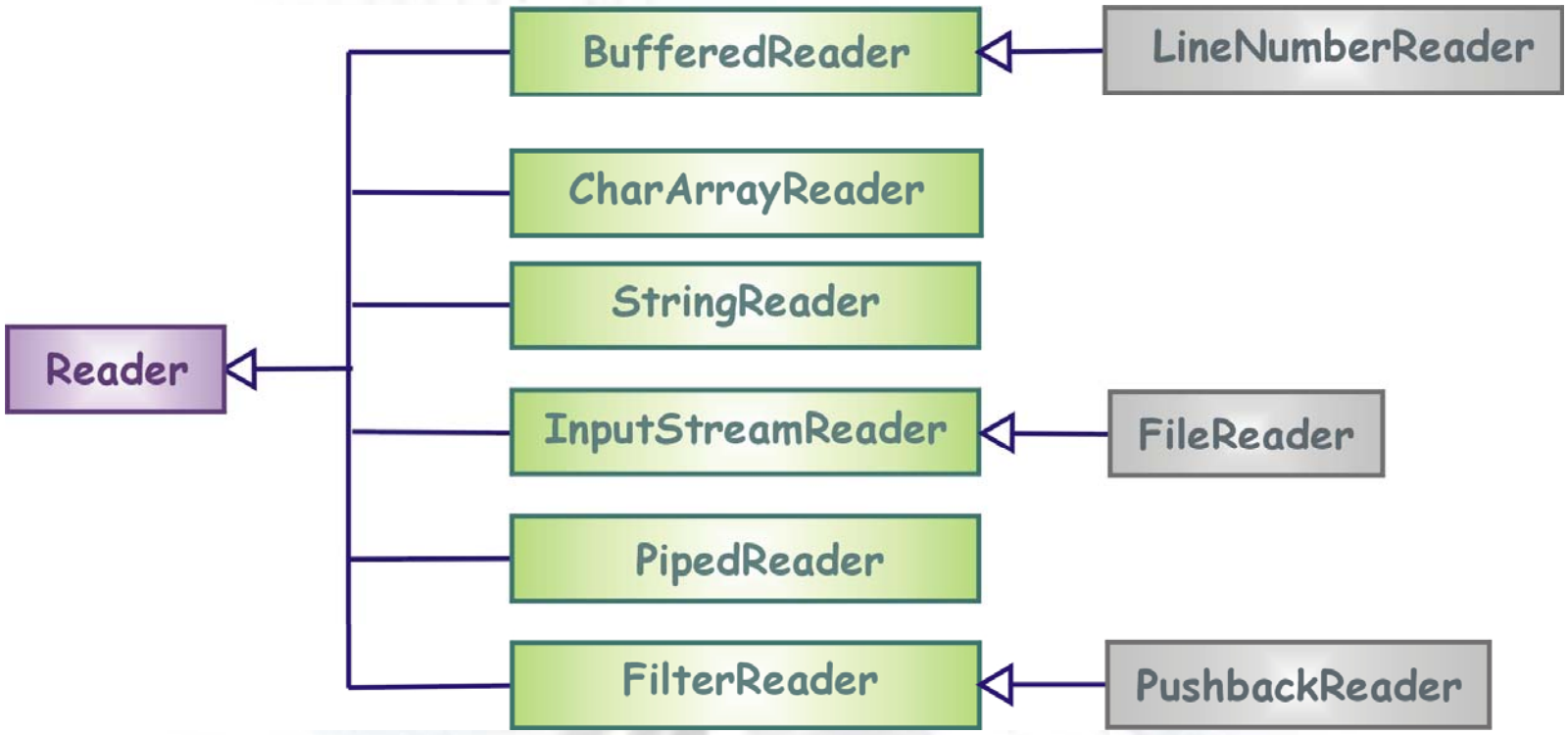
คลาสประเภท InputStream



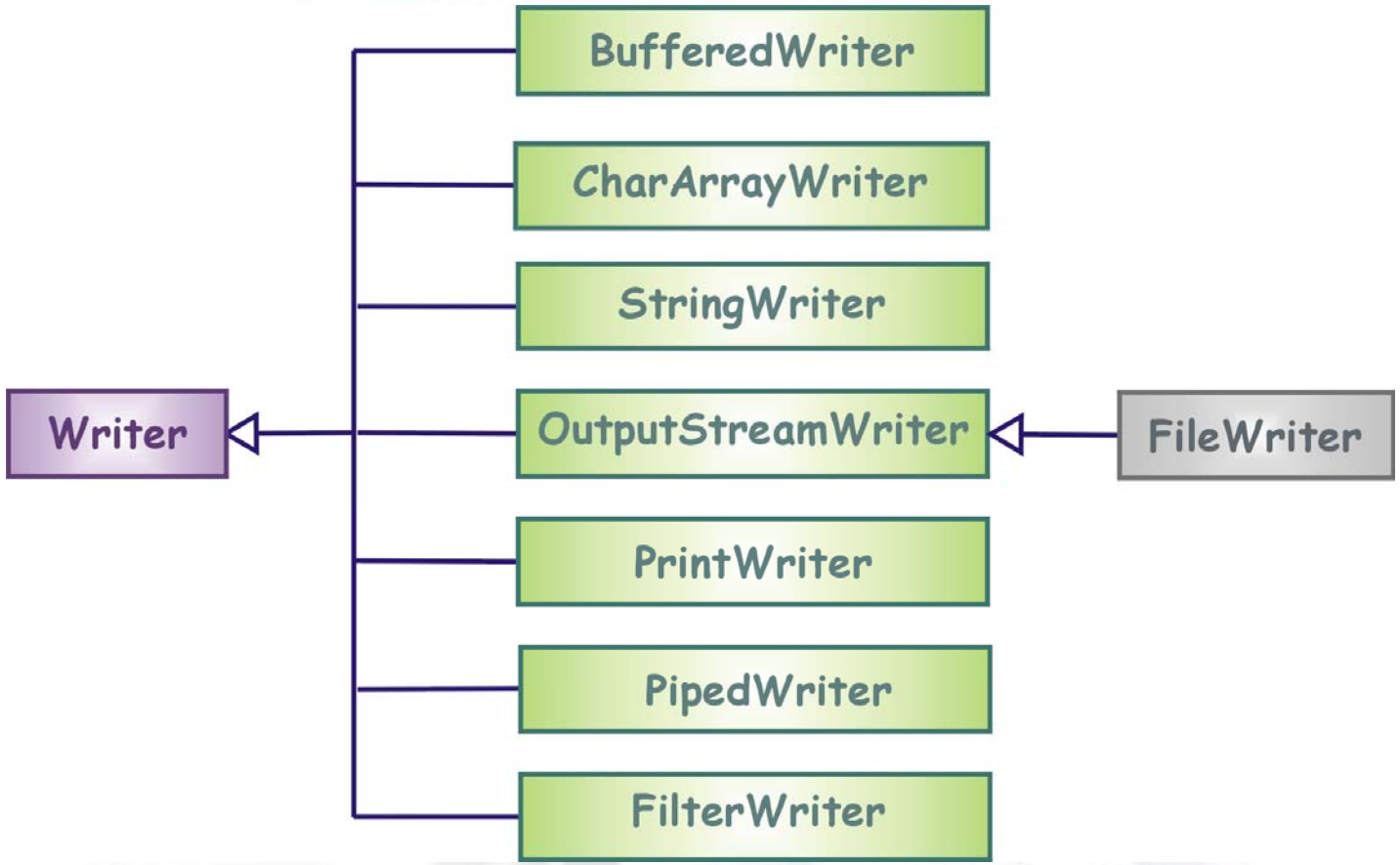
คลาสประเภท OutputStream



คลาสประเภท Reader



คลาสประเภท Writer



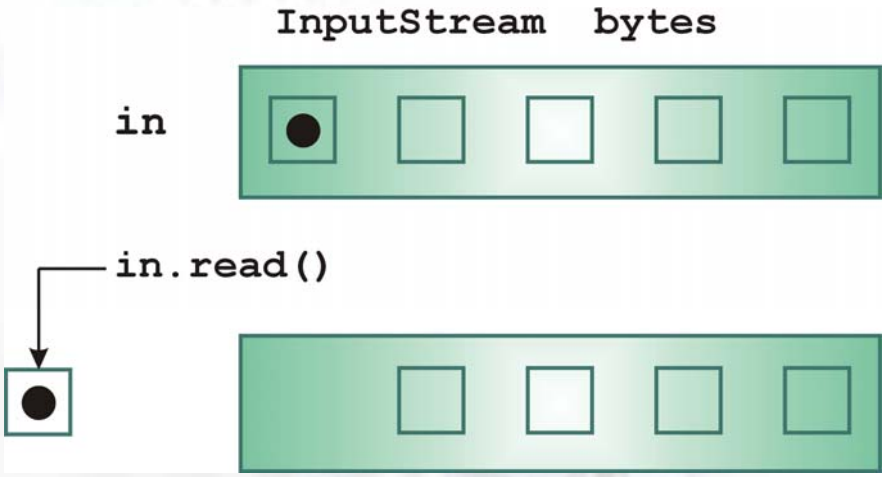
คลาสประเภท Byte Stream

- ภาษาจาวาจะมีคลาสพื้นฐานในการจัดการกับอินพุตและเอาท์พุตที่เป็นชนิดข้อมูลแบบ **byte** อยู่สองคลาสที่คู่กันคือ **InputStream** และ **OutputStream**
- คลาสทั้งสองเป็นคลาสแบบ **abstract** ซึ่งเราไม่สามารถที่จะสร้างออปเจ็คของคลาสทั้งสองได้ แต่คลาสทั้งสองจะมีคลาสที่เป็น **subclass** ซึ่งจะใช้ในการสร้างออปเจ็คสำหรับการรับและส่งข้อมูลแบบ **byte** ของโหนดที่มีต้นทางและปลายทางแบบต่าง ๆอาทิเช่น
  - **FileInputStream** และ **FileOutputStream** เป็นคลาสที่ใช้ในการสร้างออปเจ็คสำหรับต้นทางและปลายทางที่เป็นไฟล์
  - **ByteArrayInputStream** และ **ByteArrayOutputStream** เป็นคลาสที่ใช้ในการสร้างออปเจ็คสำหรับต้นทางและปลายทาง ที่เป็นอะเรย์ของชนิดข้อมูลแบบ **byte**

คลาส InputStream

- คลาส **InputStream** จะใช้ในการอ่านข้อมูลของ stream ที่เป็นชนิดข้อมูลแบบ **byte**
- คลาส **InputStream** จะนำข้อมูลจากโหนดต้นทางเข้ามาใน stream และการอ่านข้อมูลจาก stream จะเป็นการลบข้อมูลที่อ่านออกจาก stream โดยมีเมธอดที่ใช้สำหรับการอ่านข้อมูลที่เป็น **byte** หรืออะเรย์ของ **byte** เท่านั้นดังนี้
  - `int read()`
  - `int read(byte []b)`
  - `int read(byte []b,int offset,int length)`

คลาส InputStream

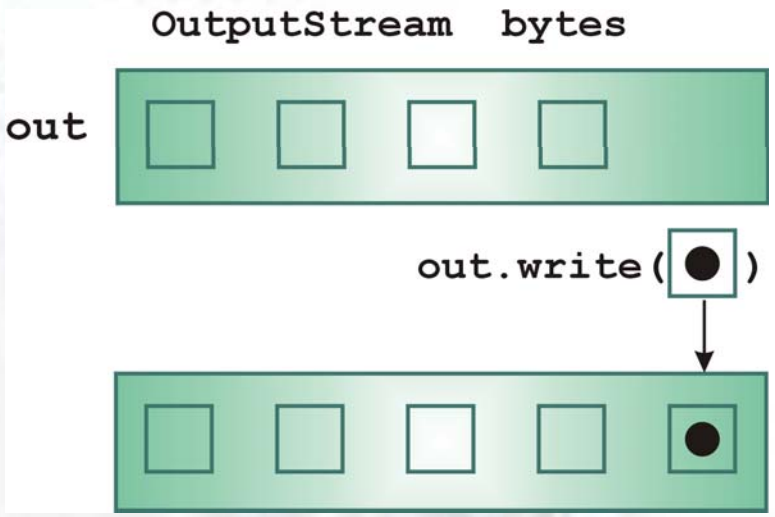


- เมธอดอื่นๆที่สำคัญ
  - `void close()`
  - `int available()`
  - `void skip(long n)`
  - `boolean markSupported()`
  - `void mark(int readlimit)`
  - `void reset()`

คลาส OutputStream

- คลาส **OutputStream** จะใช้การส่งข้อมูลของ stream ที่เป็นชนิดข้อมูลแบบ **byte** การส่งข้อมูลของออบเจ็คชนิด **Outputstream** จะเป็นการเพิ่มข้อมูลลงใน stream
- คลาส **OutputStream** จะมีเมธอดในการส่งข้อมูลชนิด **byte** ที่สอดคล้องกับเมธอด **read()** ในคลาส **InputStream** โดยคลาสนี้จะมีเมธอด **write()** ที่เป็นเมธอดแบบ **abstract** ในรูปแบบต่าง ๆดังนี้
  - `void write(int c)`
  - `void write(byte []b)`
  - `void write(byte []b,int offset,int length)`

คลาส OutputStream



- เมธอดอื่นๆที่สำคัญ
  - `void close()`
  - `void flush()`

คลาสประเภท Character Stream

- ภาษาจาวากำหนดคลาสพื้นฐานในการจัดการกับอินพุตและเอาท์พุตที่เป็นชนิดข้อมูลแบบ `char` อยู่สองคลาสคือ `Reader` และ `Writer`
- คลาสทั้งสองเป็นคลาสแบบ `abstract` โดยมี subclass ที่สืบทอดมาเพื่อใช้ในการสร้างออปเจ็คสำหรับจัดการกับโหนดต้นทางและปลายทางในรูปแบบต่าง ๆเช่น ไฟล์ หน่วยความจำ และไบท์ เป็นต้น

คลาส Reader

- **Reader** เป็นคลาสที่ใช้ในการอ่านข้อมูลของ **stream** ที่เป็นชนิดข้อมูลแบบ **char**
- **Reader** จะมีเมธอดที่เหมือนกับคลาส **InputStream** และมีหลักการทำงานที่สอดคล้องกันแต่จะรับข้อมูลหรือ **argument** ที่เป็นชนิดข้อมูล **char** โดยมีเมธอดต่างๆดังนี้
  - `int read()`
  - `int read(char []c)`
  - `int read(char []c,int offset,int length)`
  - `void close()`
  - `boolean ready()`
  - `void skip(long n)`
  - `boolean markSupported()`
  - `void mark(int readAheadlimit)`
  - `void reset()`

คลาส Writer

- Writer เป็นคลาสที่ใช้ในการเขียนข้อมูลของ stream ที่เป็นชนิดข้อมูลแบบ char
- Writer จะมีเมธอดที่เหมือนกับคลาส OutputStream และมีหลักการทีสอดคล้องกันแต่จะรับข้อมูลหรือ argument ที่เป็นชนิดข้อมูล char โดยมีเมธอดต่างๆดังนี้
  - void write(int c)
  - void write(char []c)
  - void write(char []c,int offset,int length)
  - void close()
  - void flush()
- เมธอดเพิ่มเติมเพื่อเขียนชนิดข้อมูลที่เป็น String
  - void write(String s)
  - void write(String s,int offset,int length)

โหนดสำหรับ Stream

- ภาษาจาวากำหนดโหนดที่เป็นต้นทางและปลายทางของ stream ไว้สามแบบ คือ ไฟล์ หน่วยความจำ และไปป์ (pipe)
- ไฟล์คือโหนดสำหรับ stream ที่เป็นไฟล์สำหรับอ่านหรือเขียนข้อมูลชนิด `byte`
- หน่วยความจำคือโหนดสำหรับ stream ที่ใช้สำหรับอ่านหรือเขียนข้อมูลที่เป็นอะเรย์หรือ `String`
- ไปป์คือโหนดสำหรับ stream ที่จะส่งหรือรับข้อมูลระหว่าง process หรือโปรแกรมเธรด

คลาสที่เป็นโหนดสำหรับ stream ต่าง ๆ

ชนิด	Byte Stream	Character Stream
File	FileInputStream FileOutputStream	FileReader FileWriter
Memory: Array	ByteArrayInputStream ByteArrayOutputStream	CharArrayReader CharArrayWriter
Memory: String	-	StringReader StringWriter
Pipe	PipedInputStream PipedOutputStream	PipedReader PipedWriter

ตัวอย่างโปรแกรม FileCopy.java

```
import java.io.*;

public class FileCopy {
   public static void main(String args[]) {
       try {
               FileInputStream  fin  = new
                                       FileInputStream(args[0]);
               FileOutputStream fout = new
                                       FileOutputStream(args[1]);
               byte b[] = new byte[100];
               int i = fin.read(b);
               while (i != -1) {
                       fout.write(b,0,i);
                       i = fin.read(b);
               }
               fin.close();
               fout.close();
        } catch (IOException e) { }
   }
}
```

ตัวอย่างโปรแกรม FileCopyReader.java

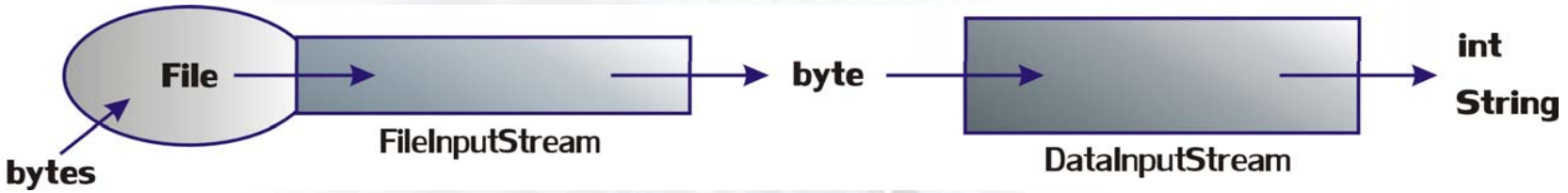
```
import java.io.*;

public class FileCopyReader {
    public static void main(String args[]) {
        try {
            FileReader     fr = new FileReader(args[0]);
            BufferedReader br = new BufferedReader(fr);
            FileWriter     fw = new FileWriter(args[1]);
            BufferedWriter bw = new BufferedWriter(fw);
            String line = br.readLine();
            while (line != null) {
                bw.write(line,0,line.length());
                bw.newLine();
                line = br.readLine();
            }
            br.close();
            bw.close();
        } catch (IOException e) { }
    }
}
```

อธิบายตัวอย่างโปรแกรม

- โปรแกรม **`FileCopy.java`** แสดงตัวอย่างของการก็อบปี้ข้อมูลของไฟล์หนึ่งไฟล์ไปยังไฟล์อื่น โดยใช้ byte stream
- โปรแกรม **`FileCopyReader.java`** แสดงตัวอย่างของก็อปปี้ข้อมูลของไฟล์เช่นเดียวกับโปรแกรม **`FileCopy.java`** แต่จะใช้ character stream

การเชื่อมต่อออปเจ็คของคลาส

- โดยทั่วไปโปรแกรมภาษาจาวาจะใช้ออปเจ็คประเภท **stream** มากกว่าหนึ่งออปเจ็คโดยจะเชื่อมออปเจ็คของ **stream** ต่าง ๆเข้าด้วยกันเพื่อใช้ในการแปลงชนิดข้อมูลประเภทต่าง ๆ



- ตัวอย่างแสดงการเชื่อมต่อออปเจ็คของคลาส **FileInputStream** ที่อ่านข้อมูลเข้ามาโดยมีชนิดข้อมูลเป็น **byte** เข้ากับออปเจ็คของคลาส **DataInputStream** เพื่อใช้อ่านข้อมูลชนิดอื่นๆได้มากขึ้นเช่น

```
FileInputStream fin = new FileInputStream(“test.dat”);
DataInputStream din = new DataInputStream(fin);
```

คลาสประเภท stream ระดับสูง

- ออปเจ็คที่ใช้ในการเชื่อมต่อ stream จะเป็นออปเจ็คของคลาสประเภท steam ระดับสูง (high-level stream) ซึ่งสามารถที่จะอ่านหรือเขียนข้อมูลที่เป็นชนิดข้อมูลอื่น ๆแล้วแปลงข้อมูลให้เป็นชนิดข้อมูลแบบ byte หรือมีบัฟเฟอร์ในการเพิ่มประสิทธิภาพในการอ่านหรือเขียนข้อมูล
- คลาสเหล่านี้จะไม่สามารถอ่านหรือเขียนข้อมูลไปยังโหนดต้นทางหรือปลายทางที่เป็นไฟล์ หน่วยความจำ หรือ Pipe ได้โดยตรง แต่จะรับข้อมูลมาจาก stream อื่น ๆที่เป็นคลาสพื้นฐานในการอ่านหรือเขียนข้อมูล

คลาสประเภท stream ระดับสูงที่สำคัญ

- **DataInputStream และ DataOutputStream**
  - เป็นคลาสที่ใช้ในการแปลงชนิดข้อมูลระหว่างชนิดข้อมูลแบบ byte กับชนิดข้อมูลแบบอื่น ๆ
- **BufferedInputStream และ BufferedOutputStream**
  - เป็นคลาสที่มีบัฟเฟอร์สำหรับชนิดข้อมูล byte อยู่ภายในเพื่อให้สามารถอ่านหรือเขียนข้อมูลขนาดใหญ่ ซึ่งจะช่วยเพิ่มประสิทธิภาพในการอ่านหรือเขียนข้อมูล
- **PrintStream**
  - เป็นคลาสที่ใช้ในการเขียนข้อความที่เป็น String ที่แปลงมาจากชนิดข้อมูลแบบ byte ออปเจ็คที่ชื่อ out และ err ที่อยู่ในคลาสที่ชื่อ System เป็นตัวอย่างของออปเจ็คที่ใช้คลาสนี้
- **PushbackInputStream**
  - คลาสนี้อนุญาตให้ส่งข้อมูลชนิด byte ที่เพิ่งอ่านมากลับไปยัง stream ได้

มีต่อ

คลาสประเภท stream ระดับสูงที่สำคัญ

- **ButteredReader และ BufferedWriter**
  - เป็นคลาสที่มีบัฟเฟอร์สำหรับชนิดข้อมูลแบบ char เพื่อให้สามารถอ่านหรือเขียนข้อมูลข้อใหญ่ได้
- **InputStreamReader และ OutputStreamWriter**
  - เป็นคลาสที่ใช้ในการแปลงชนิดข้อมูลระหว่างชนิดข้อมูลแบบ char กับชนิดข้อมูลแบบอื่น ๆ
- **PrintWriter**
  - เป็นคลาสที่ใช้ในการเขียนข้อความที่เป็น String ที่แปลงมาจากชนิดข้อมูลแบบ char
- **PushbackReader**
  - เป็นคลาสที่อนุญาตให้ส่งข้อมูลชนิด char ที่เพิ่งอ่านมากลับไปยัง stream ได้

คลาสประเภท stream ระดับสูง

ชนิด	Byte Stream	Character Stream
Buffering	BufferedInputStream BufferedOutputStream	BufferedReader BufferedWriter
Filtering	FilterInputStream FilterOutputStream	FilterReader FilterWriter
Data conversion	DataInputStream DataOutputStream	-
Printing	PrintStream	PrintWriter
Peeking ahead	PushbackInputStream	PushbackReader

การเชื่อมต่อ Stream หลายชั้น

- เราสามารถเชื่อมต่อ Stream ได้หลายชั้น เช่น

```
FileInputStream f = new FileInputStream("text.dat");
BufferedInputStream b = new BufferedInputStream(f);
DataInputStream d = new DataInputStream(b);
```

คลาส DataInputStream

- คลาส **DataInputStream** มีเมธอดในการอ่านข้อมูลชนิดต่างๆดังนี้
  - `boolean readBoolean()`
  - `byte readByte()`
  - `char readChar()`
  - `double readDouble()`
  - `float readFloat()`
  - `int readInt()`
  - `long readLong()`
  - `short readShort()`
  - `String readUTF()`

คลาส DataOutputStream

- DateOutputStream เป็น high-level FilterStream ซึ่งมีเมธอดในการเขียนข้อมูลชนิดต่างๆดังนี้
  - void writeBoolean(boolean b)
  - void writeByte(int b)
  - void writeByte(String s)
  - void writeChar(int c)
  - void writeDouble(double d)
  - void writeFloat(float f)
  - void writeInt(int i)
  - void writeLong(long l)
  - void writeShort(int s)
  - void writeUTF(String s)

ตัวอย่างของการใช้คลาส DataOutputStream

```
import java.io.*;

public class DataWriterDemo {
        public static void main (String args[]) {
                try {
                        FileOutputStream fout;
                        DataOutputStream dout;
                        fout = new FileOutputStream("data.dat");
                        dout = new DataOutputStream(fout);
                        dout.writeInt(124);
                        dout.writeDouble(2.45);
                        dout.writeChar('a');
                        dout.writeUTF("test");
                        dout.close();
                        fout.close();
                } catch (IOException ex) {
                        System.out.println(ex.toString());
                }
        }
}
```

ตัวอย่างโปรแกรม DataInputStream

```
import java.io.*;

public class DataReaderDemo {
        public static void main (String args[]) {
                try {
                        FileInputStream fin;
                        DataInputStream din;
                        fin = new FileInputStream("data.dat");
                        din = new DataInputStream(fin);
                        int x = din.readInt();
                        double d = din.readDouble();
                        char ch = din.readChar();
                        String line = din.readUTF();

                        System.out.println("x= " + x + " d= "
                                + d +  " ch= " + ch);
                        System.out.println(line);
```

มีต่อ

ตัวอย่างโปรแกรม DataInputStream

```
                    din.close();
                    fin.close();
                } catch (IOException ex) {
                    System.out.println(ex.toString());
                }
        }
}
```

ผลลัพธ์ที่ได้จากการรันโปรแกรม

```
x= 124 d= 2.45 ch= a
test
```

การแปลงข้อมูลระหว่าง byte และ char

- เราสามารถที่จะแปลง stream ระหว่างชนิดข้อมูลแบบ byte และชนิดข้อมูลแบบ char ได้โดยใช้
  - คลาส **InputStreamReader** ซึ่งจะอ่านชนิดข้อมูลแบบ byte แล้วแปลงเป็นชนิดข้อมูลแบบ char
  - คลาส **OutputStreamWriter** ซึ่งจะอ่านชนิดข้อมูลแบบ char แล้วแปลงเป็นชนิดข้อมูลแบบ byte

ตัวอย่างโปรแกรม

```
import java.io.*;

public class FileWriter {
        public static void main(String args[]) {
                FileOutputStream fout;
                OutputStreamWriter oout;
                PrintWriter p;
                String line1 = "This is a test message";
                String line2 = "This is another line";
                try {
                        fout = new
                                FileOutputStream("data.dat");
                        oout = new OutputStreamWriter(fout);
                        p = new PrintWriter(oout);
                        p.println(line1);
                        p.println(line2);
```

ตัวอย่างโปรแกรม

```
                        p.close();
                        oout.close();
                        fout.close();
                } catch (IOException ex) {
                        System.out.println(ex.toString());
                }
        }
}
```

คลาส File

- คลาสที่ชื่อ **File** เป็นคลาสที่อยู่ในแพคเก็จ **java.io** เป็นคลาสที่ใช้ในการสร้างออปเจ็คที่เป็นไฟล์หรือไดเร็กทอรี่
- คลาส **File** จะมีเมธอดในการจัดการกับไฟล์หรือไดเร็กทอรี่ และเมธอดในการสืบค้นข้อมูลต่าง ๆอยู่หลายเมธอด
- ออปเจ็คของคลาส **File** จะสร้างมาจาก constructor ที่มีรูปแบบดังนี้
  - `public File(String name)`
  - `public File(String dir, String name)`
  - `public File(File dir, String name)`

เมธอดของคลาส File

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

การสร้างออปเจ็คของคลาส File

- เราสามารถสร้างออปเจ็คชนิดคลาส **File** เพื่อตรวจสอบว่ามีไฟล์ดังกล่าวหรือไม่
- Constructor ของ **FileInputStream/FileOutputStream** หรือ **FileReader/FileWriter** มี argument ที่เป็นออปเจ็คชนิดคลาส **File**
- ตัวอย่าง

```
File f = new File("test.dat");
if (f.exists()) {
      FileInputStream fir = new FileInputStream(f);
}
```

คลาส RandomAccessFile

- ภาษาจาวามีคลาสที่สนับสนุนการอ่านหรือเขียนไฟล์เป็นแบบ random access กล่าวคือไม่จำเป็นจะต้องอ่านหรือเขียนข้อมูลของไฟล์เรียงตามลำดับจากตำแหน่งแรกสุดไปยังตำแหน่งสุดท้ายของไฟล์ คลาสประเภทนี้คือคลาส **RandomAccessFile** ซึ่งจะใช้เวลาในการอ่านหรือเขียนข้อมูลในตำแหน่งต่าง ๆเท่ากัน

- คลาส **RandomAccessFile** จะมี constructor อยู่สองรูปแบบดังนี้
  - `public RandomAccessFile(String name,String mode)`
  - `public RandomAccessFile(File file,String mode)`

  mode อาจกำหนดเป็น
    - **"r"** เป็นการเปิดไฟล์เพื่ออ่านข้อมูล
    - **"rw"** เป็นการเปิดไฟล์เพื่ออ่านและเขียนข้อมูล

เมธอดของคลาส RandomAccessFile

- คลาส **RandomAccessFile** จะมีเมธอดสำหรับการอ่านและเขียนข้อมูลที่เป็นชนิดข้อมูลแบบพื้นฐานชนิดต่าง ๆคล้ายกับคลาส **DataInputStream** และ **DataOutputStream**
- คลาส **RandomAccessFile** มีเมธอดอื่น ๆที่สำคัญเพิ่มเติมมาดังนี้
  - `long getFilePointer()`
  - `long length()`
  - `void seek(long pos)`

ObjectStream

- ภาษาจาวามีคลาสที่ใช้ในการรับและส่งข้อมูลของออปเจ็คที่ชื่อ
  - `ObjectInputStream`
  - `ObjectOutputStream`
- ทั้งนี้ออปเจ็คที่จะสามารถส่งผ่าน stream เหล่านี้ได้จะต้องเป็นออปเจ็คของคลาสที่ `implements` อินเตอร์เฟส `Serializable` ซึ่งเป็นอินเตอร์เฟสในแพคเก็จ `java.io`
- ออปเจ็คบางประเภทจะไม่สามารถที่จะ `implements` อินเตอร์เฟส `Serializable` ได้ เนื่องจากข้อมูลของออปเจ็คอาจเปลี่ยนแปลงได้ตลอดเวลา อาทิเช่น ออปเจ็คของคลาส `Thread` หรือ `FileInputStream`

ObjectStream

- เราสามารถจะระบุคุณลักษณะของออปเจ็คหรือคุณลักษณะของคลาสที่ไม่ต้องการจะเก็บหรือเรียกดูข้อมูลผ่าน stream ได้โดยการระบุให้คุณลักษณะดังกล่าวมี modifier เป็นคีย์เวิร์ดที่ชื่อ `transient` ตัวอย่างเช่น

```
public class Student implements Serializable {
    private String id,name;
    private double gpa;
    private transient String parent;
}
```

- คลาส `ObjectOutputStream` เป็นคลาสที่ใช้ในการเขียนออปเจ็คใด ๆลงใน stream โดยมีเมธอด `writeObject()`ที่ใช้ในการเขียนข้อมูล
- ส่วนคลาส `ObjectInputStream` เป็นคลาสที่ใช้ในการรับออปเจ็คใด ๆมาจาก stream โดยมีเมธอด `readObject()` ที่ใช้ในการอ่านข้อมูล

ตัวอย่างโปรแกรม

```
import java.io.*;

public class ObjectPersistenceDemo {
        public static void main(String args[]) {
                Student s = new Student("Somchai", "4514452-2",
                                                        3.21);
                FileOutputStream fout;
                ObjectOutputStream oout;
                FileInputStream fin;
                ObjectInputStream oin;
                try {
                        fout = new FileOutputStream("data.dat");
                        oout = new ObjectOutputStream(fout);
                        oout.writeObject(s);
                        System.out.println("Write to file
                                                successfully");
                        oout.close();
                        fout.close();
```

มีต่อ

ตัวอย่างโปรแกรม

```
                    fin = new FileInputStream("data.dat");
                    oin = new ObjectInputStream(fin);
                    Student s2 = (Student) oin.readObject();
                    oin.close();
                    fin.close();
                    s2.showAllDetails();
                } catch (ClassNotFoundException ex) {
                    System.out.println(ex.toString());
                } catch (IOException ex) {
                    System.out.println(ex.toString());
                }
        }
}
```

ผลลัพธ์ที่ได้จากการรันโปรแกรม

```
Write to file successfully
ID: 4514452-2
Name: Somchai
GPA: 3.21
```

สรุปเนื้อหาของบท

- stream เปรียบเสมือนท่อส่งข้อมูล สามารถแบ่งออกได้เป็น 2 ประเภทคือ byte stream และ character stream
- byte stream มีคลาสพื้นฐานคือ คลาส `InputStream` และ `OutputStream`
- character stream มีคลาสพื้นฐานคือ คลาส `Reader` และ `Writer`
- คลาส `InputStream` จะใช้ในการอ่านข้อมูลของ stream และ `OutputStream` จะนำข้อมูลจากโหนดต้นทางเข้ามาใน stream สำหรับข้อมูลที่มีชนิดข้อมูลเป็นแบบ `byte`

มีต่อ

สรุปเนื้อหาของบท

- คลาส **Reader** เป็นคลาสที่ใช้ในการอ่านข้อมูลของ **stream** และคลาส **Writer** เป็นคลาสที่ใช้ในการเขียนข้อมูลของ **stream** สำหรับข้อมูลที่มีชนิดข้อมูลเป็นแบบ **char**
- คลาสประเภท **stream** ระดับสูงที่สำคัญคือ คลาส **DataInputStream, DataOutputStream, BufferedInputStream, BufferedOutputStream, PrintStream, PushbackInputStream, BufferedReader, BufferedWriter, InputStreamReader, OutputStreamWriter, PrintWriter** และ **PushbackReader**
- การเชื่อมต่อ **stream** หลายชั้น จะทำให้สามารถเรียกใช้เมธอดได้หลายรูปแบบตามความต้องการในการใช้งาน

มีต่อ

สรุปเนื้อหาของบท

- คลาส `File` ใช้ในการสร้างออปเจ็คของไฟล์หรือไดเร็กทอรี่ โดยจะมีเมธอดที่ใช้ในการจัดการกับไฟล์
- คลาส `RandomAccessFile` จะมีรูปแบบในการอ่านและเขียนข้อมูลเป็นแบบ random access
- ออปเจ็คที่จะสามารถรับหรือส่งผ่าน stream ได้จะต้องเป็นออปเจ็คของคลาสที่ `implements` อินเตอร์เฟส `Serializable`

แบบฝึกหัด

- ข้อที่ 1
  - จากโปรแกรมจำลองระบบธนาคาร ให้ทดลองเขียนโปรแกรมเพื่อเก็บข้อมูลต่าง ๆลงไฟล์ และสามารถเรียกกลับมาใช้งานได้อย่างถูกต้อง