Môn: Lập trình Hướng đối tượng (Object Oriented Programming)

Chương 7. Nhập xuất trên Java

Nội dung

7.1. Khái niệm về các luồng (Stream) nhập xuất
7.2.Các loại luồng
7.3. Phân cấp các luồng
7.4. Thao tác với các luồng xử lý trong Java
7.5. Lớp File
7.6. Một số ví dụ

7.1. Khái niệm về các Stream nhập xuất

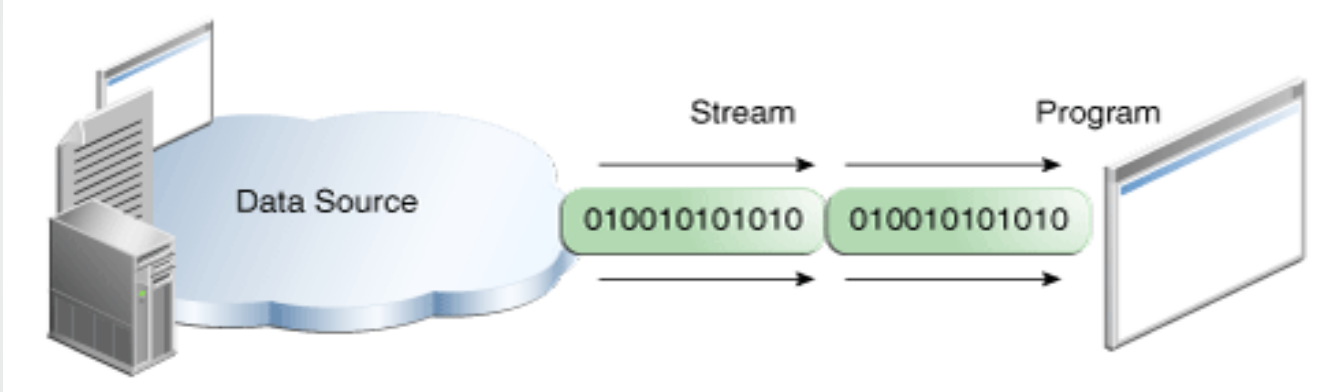
- Chương trình Java nhận và gửi dữ liệu thông qua các đối tượng là các thực thể thuộc một kiểu luồng dữ liệu nào đó.
- Luồng (stream) là một dòng dữ liệu đến từ một nguồn (source) hoặc đi đến một đích (sink)
- Nguồn và đích có thể là tập (file), bộ nhớ, một tiến trình (process), hay thiết bị (bàn phím, màn hình, …), kết nối mạng.
- ***I/O Stream*** diễn tả cho một luồng nhập hoặc luồng xuất.
  - Luồng nhập *(input stream)*: Gắn với các thiết bị nhập như bàn phím, máy scan, file…
  - Luồng xuất *(output stream):* Gắn với các thiết bị xuất như màn hình, máy in, file…

7.1. Khái niệm về các Stream nhập xuất (tt)

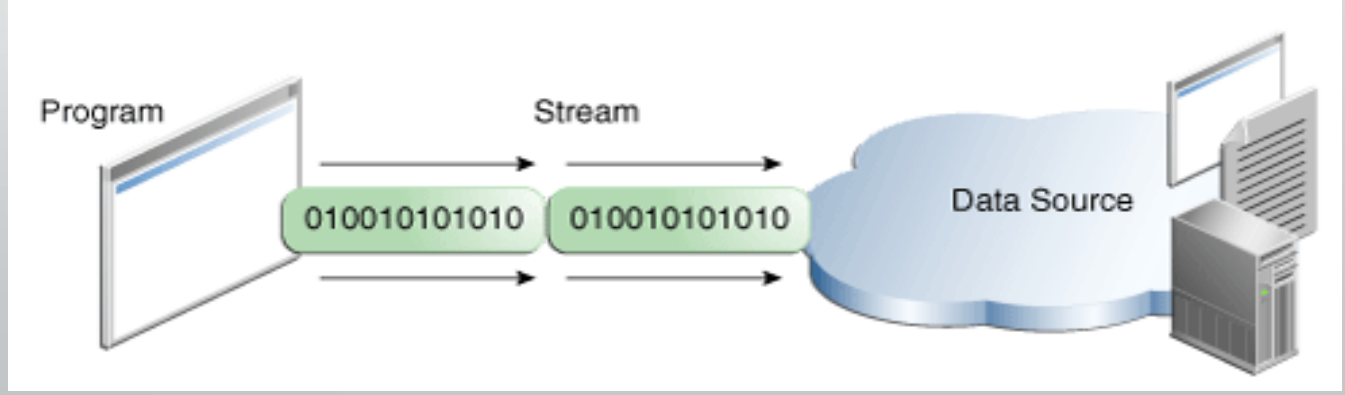
- Luồng hỗ trợ nhiều loại dữ liệu khác nhau:
  - byte,
  - các ký tự,
  - các kiểu dữ liệu cơ sở,
  - các đối tượng.
- Gói thư viện hỗ trợ nhập xuất trên Java: java.io.*
- Khi làm việc với luồng, phải bẫy lỗi tường minh lỗi IOException bằng khối try - catch.

7.1. Khái niệm về các Stream nhập xuất (tt)

- Chương trình sử dụng ***luồng nhập*** để đọc dữ liệu từ nguồn đưa vào chương trình:



- Chương trình sử dụng ***luồng xuất*** để ghi dữ liệu xuống đích.



7.2.Các loại Stream

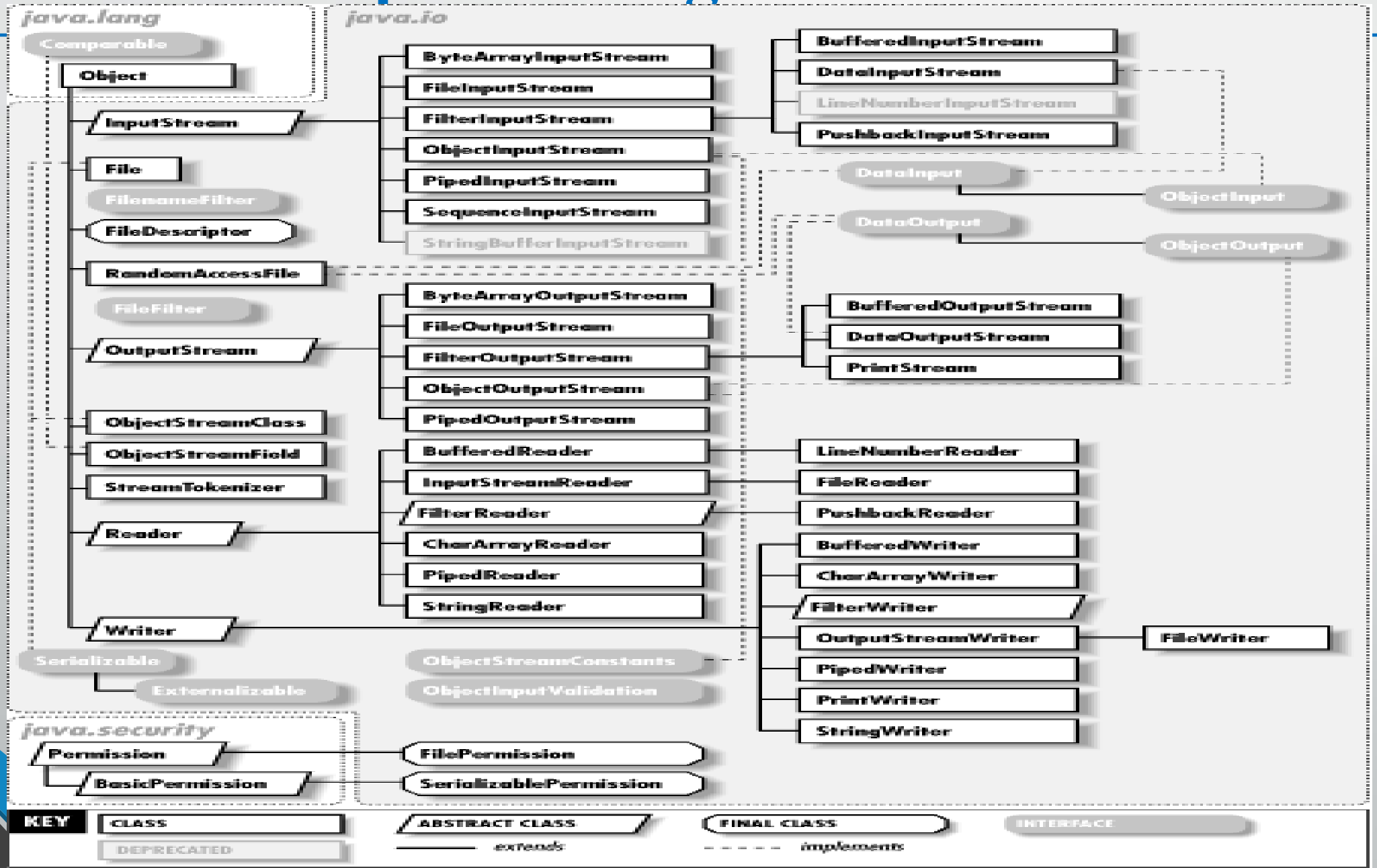
- Luồng byte: thao tác theo đơn vị byte: chức năng áp dụng cho dữ liệu dạng nhị phân
  - InputStream/OutputStream
- Luồng char: chức năng thao tác với ký tự (cả ký tự Unicode)
  - Reader/Writer
- Luồng I/O chuẩn
  - Lớp System.out: luồng xuất chuẩn, hiển thị kết quả ra màn hình
  - Lớp System.in: luồng nhập chuẩn, đọc dữ liệu từ bàn phím
  - Lớp System.err: luồng lỗi chuẩn

7.2.Các loại Stream (tt)

- Luồng dữ liệu đích *(Node streams / Data sink stream):*chức năng cơ bản cho việc đọc và ghi từ một vị trí xác định.
  - Các loại luồng node gồm: file, bộ nhớ và pipe.

- Luồng lọc *(Filter streams / Processing stream):* luồng lọc có khả năng kết nối với các luồng khác và xử lý dữ liệu “theo cách riêng”.
  - FilterInputStream/FilterOutputStream

7.3. Phân cấp các luồng



7.4. Thao tác với các luồng xử lý trong Java

- *Thao tác nhập xuất*
  - Tạo luồng, liên kết luồng với dữ liệu nguồn/đích
  - Thao tác trên luồng
  - Đóng luồng

- Abstract Classes
  - InputStream/OutputStream
  - Reader/Writer

7.4. Thao tác với các luồng xử lý trong Java (tt)

7.4.1. Byte streams
7.4.2. Character streams
7.4.3. Buffered streams
7.4.4. Standard I/O streams
7.4.5. Data streams
7.4.6. Object streams

7.4. Thao tác với các luồng xử lý trong Java (tt)

- Lớp trừu tượng InputStream

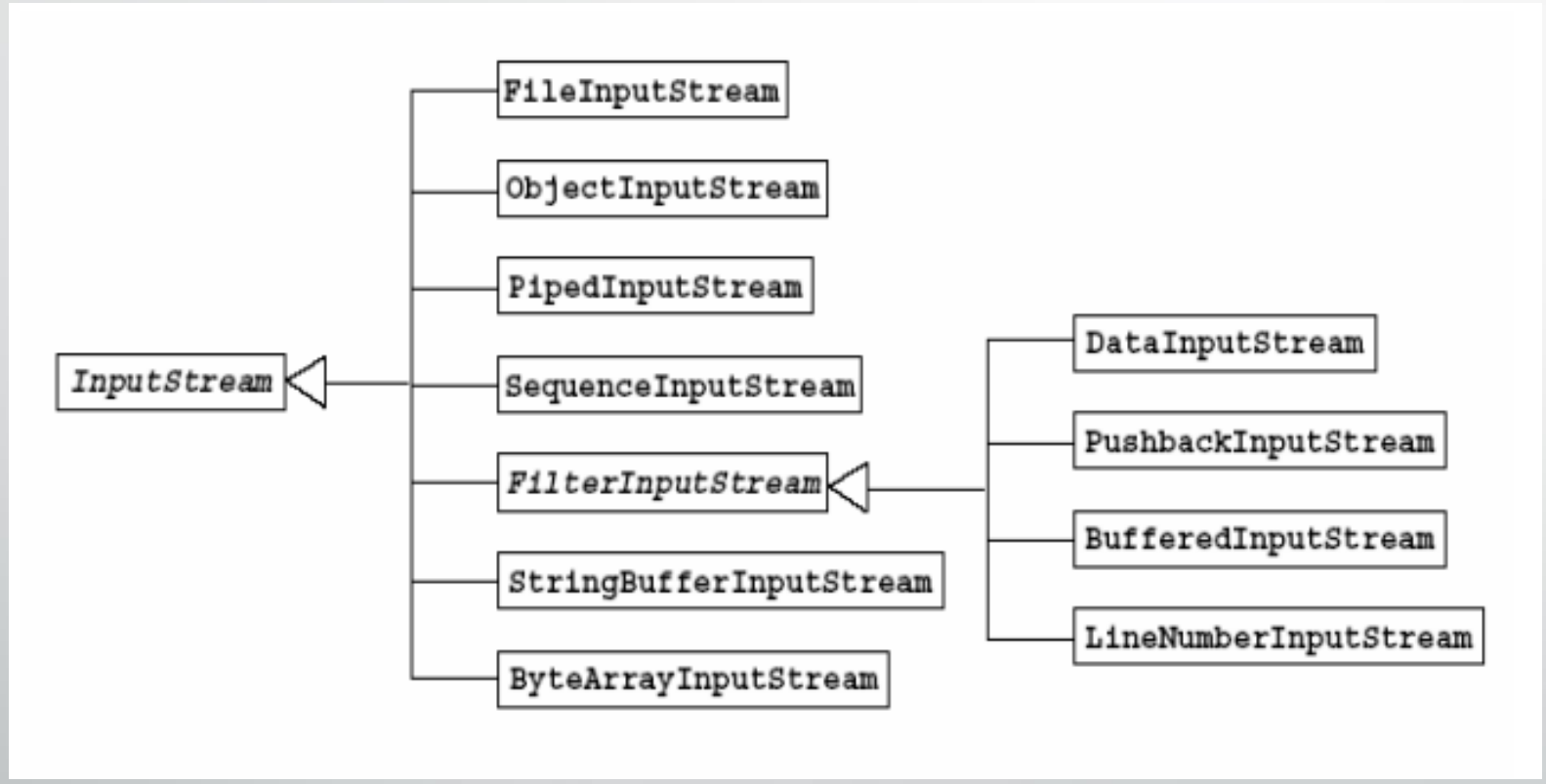
Methods

Modifier and Type	Method and Description
int	available() Returns an estimate of the number of bytes t over) from this input stream without blocking method for this input stream.
void	close() Closes this input stream and releases any sy with the stream.
void	mark(int readlimit) Marks the current position in this input strean
boolean	markSupported() Tests if this input stream supports the mark a
abstract int	read() Reads the next byte of data from the input st
int	read(byte[] b) Reads some number of bytes from the input the buffer array b.
int	read(byte[] b, int off, int len) Reads up to len bytes of data from the input bytes.
void	reset() Repositions this stream to the position at the last called on this input stream.
long	skip(long n) Skips over and discards n bytes of data from this input stream.

- `public abstract int read() throws IOException`
  Đọc một byte kế tiếp của dữ liệu từ luồng.
- `public int read(byte[] bBuf) throws IOException`
  Đọc một số byte dữ liệu từ luồng và lưu vào mảng byte bBuf.
- `public int read(byte[] cBuf, int offset, int length) throws IOException`
  Đọc `length` byte dữ liệu từ luồng và lưu vào mảng byte cBuf bắt đầu tại vị trí offset.
- `public void close() throws IOException`
  Đóng nguồn. Gọi những phương thức khác sau khi đó nguồn sẽ gây ra lỗi IOException
- `public int mark(int readAheadLimit) throws IOException`
  Đánh dấu vị trí hiện hành của stream. Sau khi đánh dấu, gọi reset() sẽ định lại vị trí của luồng đến điểm này. Không phải tất cả luồng byte –input hỗ trợ cho thao tác này.
- `public int markSupported()`
  Chỉ ra luồng có hỗ trợ thao tác mark và reset hay không

7.4. Thao tác với các luồng xử lý trong Java (tt)

- Lớp trừu tượng InputStream (tt)



7.4. Thao tác với các luồng xử lý trong Java (tt)

- Lớp trừu tượng OutputStream

Methods

Modifier and Type	Method and Description
void	close() Closes this output stream and releases any system resources associated with this stream.
void	flush() Flushes this output stream and forces any buffered output bytes to be written out.
void	write(byte[] b) Writes b.length bytes from the specified byte array to this output stream.
void	write(byte[] b, int off, int len) Writes len bytes from the specified byte array starting at offset off to this output stream.
abstract void	write(int b) Writes the specified byte to this output stream.

`public void write(int b) throws IOException`
Ghi giá trị b xác định theo dạng byte xuống output stream

`public void write(byte[] b) throws IOException`
Lưu nội dung của mảng byte b xuống luồng

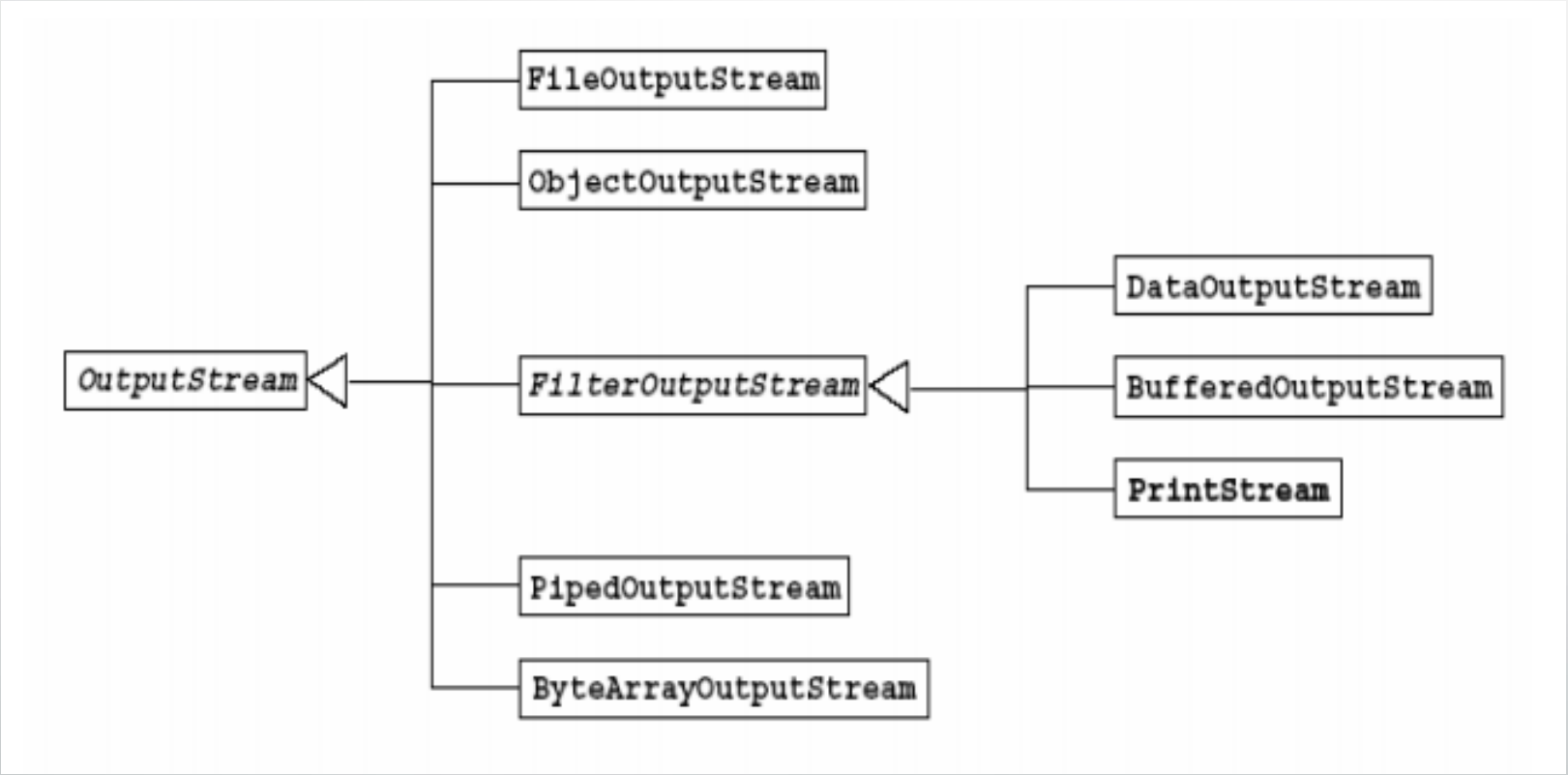
`public void write(byte[] b, int off, int len) throws IOException`
Lưu len byte của mảng byte b xuống luồng, bắt đầu từ vị trí off của mảng

`public void close() throws IOException`
Đóng nguồn. Gọi những phương thức khác liên quan đến nguồn này sau khi gọi close se gây ra lỗi IOException.

`public void flush() throws IOException`
flushes the stream.(ví dụ: Những byte được lưu trong buffer ngay lập tức được ghi xuống đích)

7.4. Thao tác với các luồng xử lý trong Java (tt)

- Lớp trừu tượng OutputStream (tt)



7.4. Thao tác với các luồng xử lý trong Java (tt)

- Lớp trừu tượng Reader

Methods

Modifier and Type	Method and Description
abstract void	close() Closes the stream and releases any sy it.
void	mark(int readAheadLimit) Marks the present position in the strea
boolean	markSupported() Tells whether this stream supports the
int	read() Reads a single character.
int	read(char[] cbuf) Reads characters into an array.
abstract int	read(char[] cbuf, int off, int l Reads characters into a portion of an a
int	read(CharBuffer target) Attempts to read characters into the sp
boolean	ready() Tells whether this stream is ready to be
void	reset() Resets the stream.
long	skip(long n) Skips characters.

`public int` **`read`**`() throws IOException`

Đọc một ký tự

`public int` **`read`**`(char[] cbuf) throws IOException`

Đọc những ký tự và lưu chúng vào mảng cbuf

`public abstract int` **`read`**`(char[] cbuf, int off, int len) throws IOException`

Đọc len ký tự và lưu chúng vào tron mảng cbuf, bắt đầu tại vị trí off của mảng

`public abstract void` **`close`**`() throws IOException`

Đóng luồng. Gọi những phương thức Reader khác của sau khi gọi close sẽ gây ra lỗi IOException

`public void` **`mark`**`(int readAheadLimit) throws IOException`

Đánh dấu vị trí hiện hành của stream. Sau khi đánh dấu, gọi reset() để thử đặt lại vị trí luồng tới điểm này. Không phải tất cả character-input đều hỗ trợ thao tác này

`public boolean` **`markSupported`**`()`

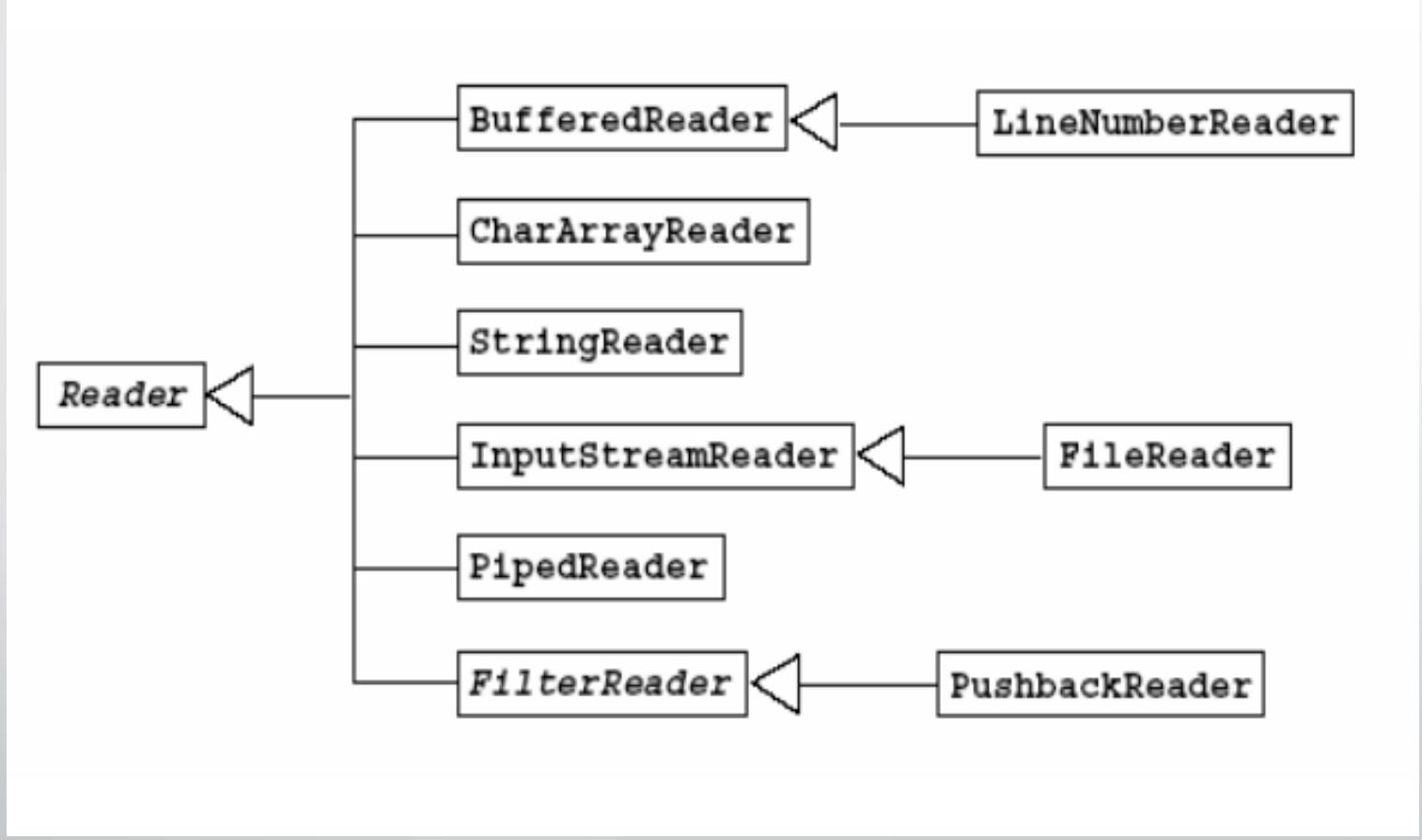
Chỉ ra luồng có hỗ trợ thao tác này hay không. Mặc định là không hỗ trợ.

`public void` **`reset`**`() throws IOException`

Đặt lại vị trí luồng tới vị trí đánh dấu lần cuối

7.4. Thao tác với các luồng xử lý trong Java (tt)

- Lớp trừu tượng Reader (tt)



7.4. Thao tác với các luồng xử lý trong Java (tt)

- Lớp trừu tượng Writer

Methods

Modifier and Type	Method and Description
Writer	append(char c) Appends the specified character to this writer.
Writer	append(CharSequence csq) Appends the specified character sequenc
Writer	append(CharSequence csq, int start Appends a subsequence of the specified writer.
abstract void	close() Closes the stream, flushing it first.
abstract void	flush() Flushes the stream.
void	write(char[] cbuf) Writes an array of characters.
abstract void	write(char[] cbuf, int off, int le Writes a portion of an array of characters.
void	write(int c) Writes a single character.
void	write(String str) Writes a string.
void	write(String str, int off, int len Writes a portion of a string.

`public void write(int c) throws IOException`
Ghi một ký tự đơn được thể hiện bằng số nguyên. Ví dụ: 'A' là được ghi la write(65)

`public void write(char[] cbuf) throws IOException`
Ghi nội dung của mảng ký tự cbuf xuống luồng

`public abstract void write(char[] cbuf, int off, int len) throws IOException`
Ghi một mảng ký tự cbuf với chiều dài là len, bắt đầu là vị trí off

`public void write(String str) throws IOException`
Ghi một chuỗi str

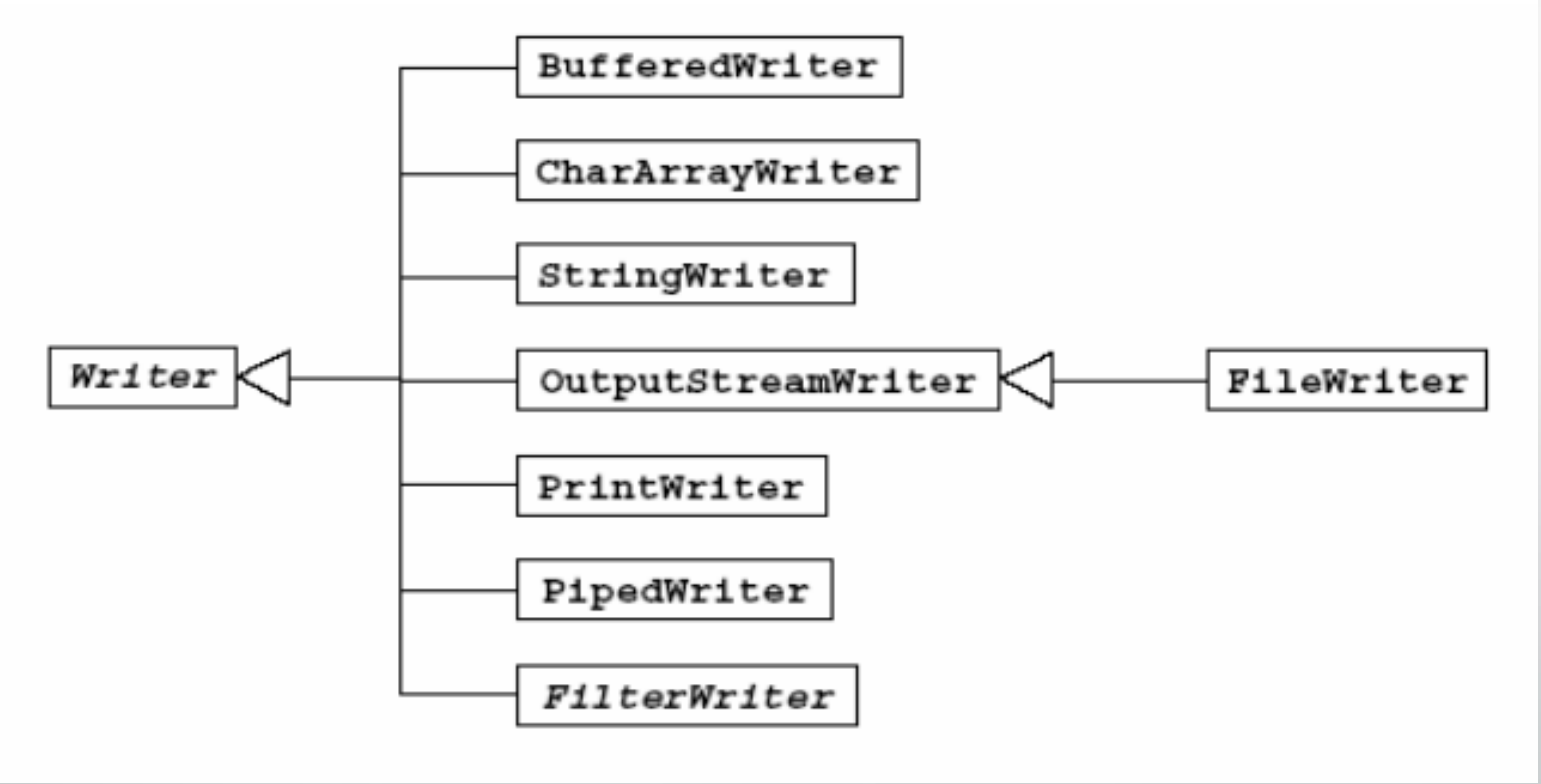
`public void write(String str, int off, int len) throws IOException`
Ghi một chuỗi str với chiều dài là len, bắt đầu từ vị trí off

`public abstract void flush() throws IOException`
Đẩy dữ liệu xuống đích đến.

`public abstract void close() throws IOException`
Đóng luồng.

7.4. Thao tác với các luồng xử lý trong Java (tt)

- Lớp trừu tượng Writer (tt)



7.4.1. Byte streams

```
2  import java.io.FileInputStream;
3  import java.io.FileOutputStream;
4  import java.io.IOException;
5  public class CopyBytes {
6      public static void main(String[] args) throws IOException{
7          FileInputStream in=null;
8          FileOutputStream out=null;
9          try{
0              in=new FileInputStream("Data\\src.txt");
1              out=new FileOutputStream("Data\\dest.txt");
2              int c;;
3              while((c=in.read())!=-1){
4                  out.write(c);
5              }
6          }finally{
7              if(in!=null) in.close();
8              if(out!=null) out.close();
9          }
0      }
1  }
```

7.4.1. Byte streams (tt)

Lưu ý về luồng Byte

- Luồng byte biểu diễn một loại nhập xuất ở mức thấp mà ta nên tránh.
  - Nếu dữ liệu là dữ liệu ký tự, thì phương pháp tốt nhất là sử dụng luồng ký tự.
  - Ngoài ra, còn có nhiều luồng khác thích hợp cho những kiểu dữ liệu phức tạp.
- Các luồng byte chỉ nên sử dụng nhập xuất cho các kiểu nhập xuất cơ bản.
- Tất cả các luồng khác đều dựa trên luồng byte.

7.4.2. Character streams

- Java hỗ trợ đọc và thao tác trên luồng đối với các ký tự Unicode.
- Luồng ký tự *(character stream)*: Thực hiện các thao tác nhập xuất theo ký tự.
- Tất cả các lớp của luồng ký tự đều được dẫn xuất từ lớp Reader và Writer.
- Các lớp thao tác trên file của luồng ký tự:
  - FileReader
  - FileWriter.

7.4.2. Character streams (tt)

```
3  import java.io.FileReader;
4  import java.io.FileWriter;
5  import java.io.IOException;
6  public class CopyCharacters {
7      public static void main(String[] args) throws IOException{
8          FileReader in=null;
9          FileWriter out=null;
10         try{
11             in=new FileReader("Data/src.txt");
12             out=new FileWriter("Data/dest.txt");
13             int c;
14             while((c=in.read())!=-1)
15                 out.write(c);
16         }
17         finally{
18             if(in!=null) in.close();
19             if(out!=null) out.close();
20         }
21     }
22 }
```

7.4.2. Character streams (tt)

- Luồng ký tự thường là "wrappers" của luồng byte.
- Luồng ký tự sử dụng luồng byte để thực hiện nhập xuất vật lý. Trong khi đó luồng ký tự xử lý chuyển đổi giữa ký tự và byte.
  - FileReader dùng FileInputStream
  - FileWriter dùng FileOutputStream
- Dùng luồng ký tự có thể thao tác được cho luồng byte.
- Có thể chuyển từ luồng byte sang luồng ký tự nhờ: InputStreamReader và OutputStreamReader.

7.4.2. Character streams (tt)

Line-Oriented I/O

- Thông thường nhập xuất trên ký tự thường xảy ra là một chuỗi các ký tự hơn là các ký tự riêng lẻ.
  - Thông dụng nhất là một dòng: một chuỗi các ký tự với một tín hiệu kết thúc dòng.
  - Tín hiệu kết thúc dòng có thể là \r (carriage-return) hoặc \n (line-feed).

7.4.2. Character streams (tt)

Line-Oriented I/O (tt)

```
3  import java.io.FileReader;
4  import java.io.FileWriter;
5  import java.io.IOException;
6  import java.util.Scanner;
7  public class CopyLines {
8      public static void main(String[] args) throws IOException{
9          FileWriter out=null;
10         Scanner in=null;
11         try{
12             in=new Scanner(new FileReader("Data/src.txt"));
13             out=new FileWriter("Data/dest.txt");
14             String line=null;
15             while(in.hasNextLine()){
16                 line=in.nextLine();
17                 out.write(line+"\n");
18             }
19         }finally{
20             if(in!=null) in.close();
21             if(out!=null) out.close();
22         }
23     }
24 }
```

7.4.3. Buffered streams

- Nếu một I/O không có bộ đệm, nghĩa là mỗi yêu cầu đọc hoặc ghi được xử lý trực tiếp trên thiết bị.
- Để giảm các chi phí trên, nền tảng Java hỗ trợ luồng nhập xuất có bộ đệm.
  - Luồng nhập có bộ đệm *(buffered input stream)* đọc dữ liệu từ một vùng nhớ được xem như một bộ đệm; chỉ ghi vào khi nào bộ đệm rỗng.
  - Luồng xuất có bộ đệm *(buffered output stream)* ghi dữ liệu tới bộ đệm; chờ cho đến khi bộ đệm đầy mới ghi tới đích.

7.4.3. Buffered streams (tt)

Các lớp của luồng đệm

- Một chương trình có thể chuyển một luồng không bộ đệm thành luồng có bộ đệm *(buffered stream)*.
- Có 4 lớp luồng đệm dùng để "wrap" các luồng không bộ đệm:
  - BufferedInputStream và BufferedOutputStream là các luồng byte có bộ đệm.
  - BufferedReader và BufferedWriter là các luồng ký tự có bộ đệm.

7.4.3. Buffered streams (tt)

Flushing Buffered Streams

- Vài trường hợp dữ liệu không chứa đủ bộ đệm. Vì vậy, phải dùng flush để ghi hết những gì còn lại trong bộ đệm ra.
- Một vài lớp luồng xuất có bộ đệm hỗ trợ autoflush.
  - Khi chức năng autoflush được thiết lập, cần phải thiết lập sự kiện cụ thể để bộ đệm ghi ra.
  - Ví dụ, autoflush trong đối tượng PrintWriter, bộ đệm ghi ra mỗi khi có lệnh println hoặc format.
- Muốn ghi ra tại thời điểm bất kỳ, ta dùng phương thức flush().

7.4.3. Buffered streams (tt)

```
4  import java.io.BufferedOutputStream;
5  import java.io.FileInputStream;
6  import java.io.FileOutputStream;
7  import java.io.IOException;
8  import java.io.PrintWriter;
9  import java.util.Scanner;
10 public class DemoBufferedStream {
11     private final String[] dsTenSV={"An", "Lan","Hoa","Hoàng","Nga","Tâm"};
12     private final float[] DSDiemGK={7,8.5f,9,10,4.5f,6};
13     private final float[] DSDiemCK={5,5.5f,6.5f,10,6,4.5f};
14     public void writeTo (String fileName) throws IOException{
15         PrintWriter out=null;
16         try{
17             out=new PrintWriter(new BufferedOutputStream(
18                                 new FileOutputStream(fileName)), true);
19             for (int i = 0; i < dsTenSV.length; i++) {
20                 out.println(String.format("%-10s %10s %10s",
21                         dsTenSV[i]+";",DSDiemGK[i]+";", DSDiemCK[i]));
22             }
23         }finally{
24             if(out!=null) out.close();
25         }
26     }
```

7.4.3. Buffered streams (tt)

```
29 public void readFrom(String fileName) throws IOException{
30     Scanner in=null;
31     try{
32         in=new Scanner(new BufferedInputStream(
33                         new FileInputStream(fileName)));
34         String line=null;
35         while(in.hasNextLine()){
36             line=in.nextLine();
37             String[] cols=line.split(";");
38             float diemGK=Float.parseFloat(cols[1]);
39             float diemCK=Float.parseFloat(cols[2]);
40             float diemTB=(diemGK+diemCK)/2;
41             System.out.println(String.format("%-25s %10s",
42                                              line+";",diemTB));
43         }
44     }finally{
45         if(in!=null) in.close();
46     }
47 }
```

7.4.4. Standard I/O streams

- Có 3 luồng chuẩn:
  - Luồng nhập chuẩn - System.in
  - Luồng xuất chuẩn - System.out
  - Luồng xuất lỗi chuẩn - System.err
- System.out, System.err được định nghĩa như các đối tượng PrintStream.

7.4.4. Standard I/O streams (tt)

```
 3 import java.util.Scanner;
 4 public class DemoStandardStream {
 5     public static void main(String[] args) {
 6         int[] a = { 10, 4, 9 ,15 };
 7         Scanner sc=null;
 8         try {
 9             sc = new Scanner(System.in);
10             System.out.println("Nhập vị trí: ");
11             int i = sc.nextInt();
12             System.out.println(
13                     String.format("Phần tử tại vi tri %d là: %s", i, a[i]));
14         }catch (ArrayIndexOutOfBoundsException ex) {
15             System.err.println("Lỗi:"+ex.toString());
16         }
17         catch (Exception ex) {
18             System.err.println("Lỗi:"+ex.getStackTrace()[2]);
19         }
20         finally{
21             if(sc!=null) sc.close();
22         }
23     }
24 }
```

7.4.5. Data streams

- Luồng dữ liệu *(data streams)* hỗ trợ nhập xuất nhị phân trên các kiểu dữ liệu cơ sở *(boolean, char, byte, short, int, long, float, và double)* và String.
- Tất cả các luồng dữ liệu hiện thực từ giao diện DataInput hoặc từ DataOutput.
- Hầu hết việc nhập xuất trên luồng dữ liệu thì dùng lớp DataInputStream và DataOutputStream.
- Những dữ liệu được ghi bởi DataOutputStream sẽ đọc được bởi DataInputStream

7.4.5. Data streams (tt)

- Một số phương thức của DataInputStream

Methods

Modifier and Type	Method and Description
int	read(byte[] b) Reads some number of bytes from the contained input stream and stores them into the buffer array b.
int	read(byte[] b, int off, int len) Reads up to len bytes of data from the contained input stream into an array of bytes.
boolean	readBoolean() See the general contract of the readBoolean method of DataInput.
byte	readByte() See the general contract of tl
char	readChar() See the general contract of tl
double	readDouble() See the general contract of tl
float	readFloat() See the general contract of tl
void	readFully(byte[] b) See the general contract of tl
void	readFully(byte[] b, int See the general contract of tl
int	readInt() See the general contract of tl
String	readLine()
long	readLong() See the general contract of the readLong method of DataInput.
short	readShort() See the general contract of the readShort method of DataInput.
int	readUnsignedByte() See the general contract of the readUnsignedByte method of DataInput.
int	readUnsignedShort() See the general contract of the readUnsignedShort method of DataInput.
String	readUTF() See the general contract of the readUTF method of DataInput.
static String	readUTF(DataInput in) Reads from the stream in a representation of a Unicode character string encoded in modified UTF-8 format; this string of characters is then returned as a String.
int	skipBytes(int n) See the general contract of the skipBytes method of DataInput.

7.4.5. Data streams (tt)

- Một số phương thức của DataOutputStream

Methods

Modifier and Type	Method and Description
void	`flush()` Flushes this data output stream.
int	`size()` Returns the current value of the counter `written`, the number of bytes written to this data output stream so far.
void	`write(byte[] b, int off, int len)` Writes `len` bytes from the specified byte array starting at offset `off` to the underlying output stream.
void	`write(int b)` Writes the specified byte (the low eight bits of the argument `b`) to the underlying output stream.
void	`writeBoolean(boolean v)` Writes a `boolean` to the underlying output stream as a 1-byte value.
void	`writeByte(int v)` Writes out a `byte` to the underlying
void	`writeBytes(String s)` Writes out the string to the underlyi
void	`writeChar(int v)` Writes a `char` to the underlying out
void	`writeChars(String s)` Writes a string to the underlying ou
void	`writeFloat(float v)` Converts the float argument to an `int` using the `floatToIntBits` method in class `Float`, and then writes that `int` value to the underlying output stream as a 4-byte quantity, high byte first.
void	`writeInt(int v)` Writes an `int` to the underlying output stream as four bytes, high byte first.
void	`writeLong(long v)` Writes a `long` to the underlying output stream as eight bytes, high byte first.
void	`writeShort(int v)` Writes a `short` to the underlying output stream as two bytes, high byte first.
void	`writeUTF(String str)` Writes a string to the underlying output stream using modified UTF-8 encoding in a machine-independent manner.

7.4.5. Data streams (tt)

```
3  import java.io.BufferedInputStream;
4  import java.io.BufferedOutputStream;
5  import java.io.DataInputStream;
6  import java.io.DataOutputStream;
7  import java.io.FileInputStream;
8  import java.io.FileOutputStream;
9  import java.io.IOException;
10 public class DemoDataStream {
11     private final String[] DSTenSV={"An", "Lan","Hoa","Hoàng","Nga","Tâm"};
12     private final float[] DSDiemGK={7,8.5f,9,10,4.5f,6};
13     private final float[] DSDiemCK={5,5.5f,6.5f,10,6,4.5f};
14     public void writeTo
15         DataOutputStrea
16         try{
17             out=new Dat
18
19             for (int i
20                 out.wri
21                 out.wri
22                 out.wri
23             }
24         }finally{ if(ou
25     }
```

```
28 public void readFrom(String fileName) throws IOException{
29     DataInputStream in=null;
30     try{
31         in=new DataInputStream(new BufferedInputStream(
32                                new FileInputStream(fileName)));
33         while(in.available()!=0){
34             String tenSV=in.readUTF();
35             float diemGK=in.readFloat();
36             float diemCK=in.readFloat();
37             float diemTB=(diemGK+diemCK)/2;
38             System.out.println(String.format("%-10s %7s %7s %7s",
39                                              tenSV,diemGK,diemCK,diemTB));
40         }
41     }finally{
42         if(in!=null) in.close();
43     }
44 }
```

7.4.6. Object streams

Tuần tự hóa dữ liệu

- Tính bền vững *(persistence)* là khả năng một đối tượng duy trì sự tồn tại độc lập sau thời gian sống của chương trình tạo ra nó.
- Java cung cấp cơ chế được gọi là tuần tự hóa đối tượng *(Object Serialization)* để tạo đối tượng bền vững.
- Khi một đối tượng được tuần tự hóa, nó sẽ được chuyển thành tuần tự các byte dạng thô, biểu diễn đối tượng.

7.4.6. Object streams (tt)

Luồng đối tượng

- **Luồng đối tượng** *(Object Streams)* hỗ trợ việc đọc, ghi các đối tượng.
- Nếu đối tượng hiện thực giao diện Serializable thì ta có thể sử dụng luồng đối tượng để đọc, ghi đối tượng đó.
- Hai lớp hỗ trợ luồng đối tượng:
  - ObjectInputStream
  - ObjectOutputStream
- Hai lớp này tương ứng hiện thực các giao diện:
  - ObjectInput
  - ObjectOutput

7.4.6. Object streams (tt)

Luồng đối tượng (tt)

- Bất kỳ đối tượng nào mà ta muốn tuần tự hóa *(serialize)* thì bắt buộc phải hiện thực giao diện Serializable.
- Để tuần tự hóa một đối tượng, gọi phương thức writeObject của lớp ObjectOutputStream.
- Để khôi phục lại đối tượng đã được tuần tự hóa trước đó (*deserialize)*, gọi phương thức readObject của lớp ObjectInputStream.
- Các đối tượng được tuần tự hóa có thể được ghi vào file, truyền qua mạng hoặc có thể chuyển sang các luồng khác.

7.4.6. Object streams (tt)

ObjectSerialization.java

```
1  package myio;
2
3  import java.io.FileInputStream;
4  import java.io.FileOutputStream;
5  import java.io.ObjectInputStream;
6  import java.io.ObjectOutputStream;
7
8  public class ObjectSerialization {
9      public void serialize2file(Object obj, String dataFile)throws Exception{
10         FileOutputStream fos=new FileOutputStream(dataFile);
11         ObjectOutputStream oos=new ObjectOutputStream(fos);
12         oos.writeObject(obj);
13         oos.close();
14     }
15     public Object deserialize(String datafile)throws Exception{
16         Object obj=null;
17         FileInputStream fis=new FileInputStream(datafile);
18         ObjectInputStream ois=new ObjectInputStream(fis);
19         obj=ois.readObject();
20         ois.close();
21         return obj;
22     }
23 }
```

7.5. Lớp File

- Lớp File dùng cho việc thao tác trên file và thư mục.
- Tạo đối tượng File

```
File myFile;
myFile = new File("data.txt");
myFile = new File("myDocs", "data.txt");
```

- Thư mục cũng được coi như là một tập tin
  - File myDir = new File("myDocs");
  - File myFile = new File(myDir, "data.txt");
  - có phương thức riêng để thao tác với thưmục

7.5. Lớp File (tt)

- Một số phương thức của lớp File:
  - Tên tập tin
    - String getName()
    - String getPath()
    - String getAbsolutePath()
    - String getParent()
    - boolean renameTo(File newName)
  - Kiểm tra tập tin
    - boolean exists()
    - boolean canWrite(), boolean canRead()
    - boolean isFile()
    - boolean isDirectory()
    - boolean isAbsolute()

7.5. Lớp File (tt)

- Một số phương thức của lớp File (tt):
  - Nhận thông tin
    - long lastModified()
    - long length()
    - boolean delete()
  - Thư mục
    - boolean mkdir()
    - String[] list()

7.6. Một số ví dụ

- Copy file

```
import java.io.*;

public class CopyFile {
    public static void main(String args[]) {
        try {
            FileReader src = new FileReader(args[0]);
            BufferedReader in = new BufferedReader(src);
            FileWriter des = new FileWriter(args[1]);
            PrintWriter out = new PrintWriter(des);
            String s;

            s = in.readLine();
            while (s != null) {
                out.println(s);
                s = in.readLine();
            }

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

7.6. Một số ví dụ (tt)

- Copy file

```
import java.io.*;

public class CopyFile2 {
    public static void main(String args[]) {
        try {
            FileReader src = new FileReader(args[0]);
            FileWriter des = new FileWriter(args[1]);
            char buf[] = new char[128];
            int charsRead;
            charsRead = src.read(buf);
            while (charsRead != -1) {
                des.write(buf, 0, charsRead);
                charsRead = src.read(buf);
            }
            src.close();
            des.close();
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

7.6. Một số ví dụ (tt)

- Ghi đối tượng

```
import java.io.*;

public class TestObjectOutputStream {
   public static void main(String args[]) {
        Record r[] = { new Record("john", 5.0F),
                       new Record("mary", 5.5F),
                       new Record("bob", 4.5F) };

        try {
            FileOutputStream fout = new FileOutputStream(args[0]);
            ObjectOutputStream out = new ObjectOutputStream(fout);

            for (int i=0; i<r.length; i++)
                out.writeObject(r[i]);

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

```
import java.io.Serializable;

class Record implements Serializable {
  private String name;
  private float score;

  public Record(String s, float sc) {
      name = s;
      score = sc;
  }

  public String toString() {
      return "Name: " + name + ", score: " + score;
  }
}
```

7.6. Một số ví dụ (tt)

- Đọc đối tượng

```
import java.io.*;

public class TestObjectInputStream {
    public static void main(String args[]) {
        Record r;

        try {
            FileInputStream fin = new FileInputStream(args[0]);
            ObjectInputStream in = new ObjectInputStream(fin);

            while (true) {
                r = (Record) in.readObject();
                System.out.println(r);
            }
        }
        catch (EOFException e) {
            System.out.println("No more records");
        }
        catch (ClassNotFoundException e) {
            System.out.println("Unable to create object");
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```
import java.io.Serializable;

class Record implements Serializable {
   private String name;
   private float score;

   public Record(String s, float sc) {
       name = s;
       score = sc;
   }

   public String toString() {
       return "Name: " + name + ", score: " + score;
   }
}
```

7.6. Một số ví dụ (tt)

- Đọc/ghi ngẫu nhiên

```
import java.io.*;

public class WriteRandomFile {
   public static void main(String args[]) {
        int a[] = { 2, 3, 5, 7, 11, 13 };

        try {
            File fout = new File(args[0]);
            RandomAccessFile out;
            out = new RandomAccessFile(fout, "rw");

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

```
import java.io.*;

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

        try {
            File fin = new File(args[0]);
            RandomAccessFile in = new RandomAccessFile(fin, "r");

            int recordNum = (int) (in.length() / 4);
            for (int i=recordNum-1; i>=0; i--) {
                in.seek(i*4);
                System.out.println(in.readInt());
            }
        }
        catch (IOException e) {
            e.printStackTrace();
        }
   }
}
```