

LẬP TRÌNH MẠNG

VÀO RA VỚI STREAM TRONG JAVA

hungdn@ptit.edu.vn

Nội dung



- Luồng bytes
- Luồng dữ liệu
- Luồng ký tự
- Luồng đối tượng

Vào ra với Stream trong Java



- Luồng (stream)
 - Stream là chuỗi dữ liệu được truyền với chiều dài chưa xác định
 - Luồng vào đưa dữ liệu từ nguồn bên ngoài (keyboard, file, other app) vào chương trình Java
 - Luồng ra đưa dữ liệu từ chương trình Java ra đích bên ngoài (console, file, other app)
- Các luồng Java
 - Luồng byte
 - Luồng dữ liệu có định dạng
 - Luồng ký tự
 - Luồng đối tượng

Luồng I/O chuẩn trong Java



- Java cung cấp 3 luồng vào/ra chuẩn
 - Đầu vào chuẩn: truyền dữ liệu tới chương trình từ (thường) bàn phím, được biểu diễn System.in
 - Đầu ra chuẩn: truyền dữ liệu từ chương trình tới (thường) là màn hình console của người dùng, được biểu diễn System.out
 - Lỗi chuẩn: truyền dữ liệu lỗi từ chương trình tới (thường) là màn hình console của người dùng, được biểu diễn System.err



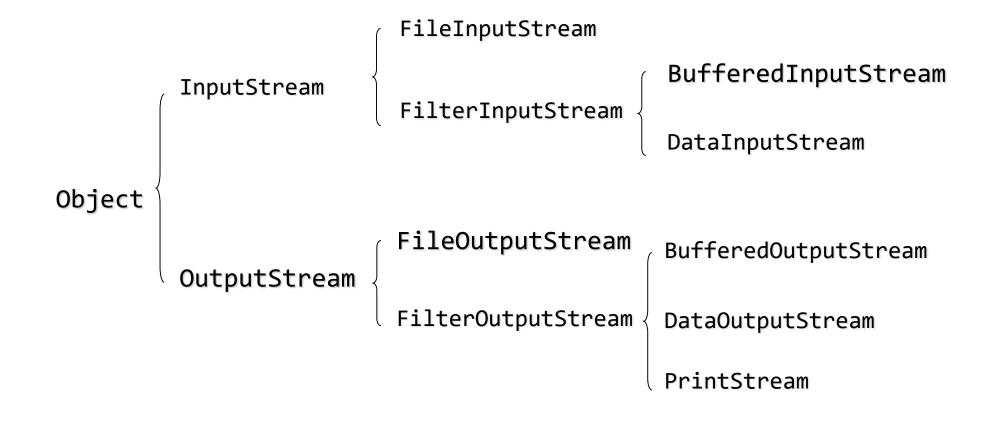
1. Luồng byte

- Output Streams
- Input Streams
- Filter Streams

Các luồng byte



Các lớp InputStream và OutputStream để đọc và ghi bytes



Output Streams



Lớp Java cơ sở làm việc với luồng ra là java.io.OutputStream

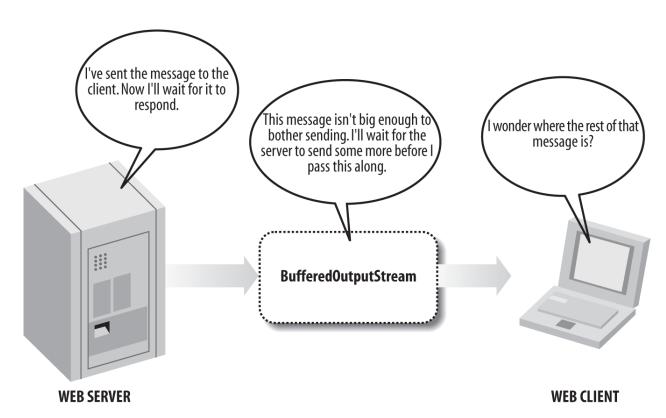
```
public abstract class OutputStream
```

• Cung cấp các phương thức cần thiết để ghi dữ liệu

Bộ đệm



- Các luồng cũng có thể sử dụng bộ đệm, trực tiếp bằng mã Java hoặc trong phần cứng mạng
 - Thường được thực hiện bằng cách gắn một luồng BufferedOutput hoặc BufferedWriter vào một luồng cơ bản
 - Lưu ý: khi ghi xong dữ liệu, một việc quan trọng cần thực hiện là giải phóng (flush) luồng ra



 Phương thức flush phá vỡ bế tắc bằng cách buộc luồng đệm dữ liệu gửi dữ liệu đi ngay cả khi bộ đệm chưa đầy

Java 7



```
OutputStream out = null;
try {
  out = new
FileOutputStream("/tmp/data.txt");
  // work with the output stream...
} catch (IOException ex) {
  System.err.println(ex.getMessage());
} finally {
  if (out != null) {
    try {
      out.close();
    } catch (IOException ex) {
      // ignore
                  try (OutputStream out = new FileOutputStream("/tmp/data.txt")) {
                    // work with the output stream...
                    catch (IOException ex) {
                    System.err.println(ex.getMessage());
```

Ex: OutputStream



Luồng ra màn hình

```
try{
     OutputStream output = new BufferedOutputStream(System.out);
    output.write(1111111);
    output.close();
}catch(IOException e){
     System.out.println(e);
}
```

Luồng ra file

```
try{
     OutputStream output = new FileOutputStream("output.txt");
     output.write(1111111);
     output.close();
}catch(IOException e){
         System.out.println(e);
}
```

Ex: OutputStream



Luồng ra socket

Input Streams



• Lớp java cơ sở làm việc với luồng vào là java.io.InputStream

```
public abstract class InputStream
```

Lớp này cung cấp các phương thức cần thiết để đọc dữ liệu n

```
public abstract int read() throws IOException
public int read(byte[] input) throws IOException
public int read(byte[] input, int offset, int length) throws
IOException
public long skip(long n) throws IOException
public int available() throws IOException
public void close() throws IOException
```

Input Streams (Cont.)



 Lớp InputStream cũng cung cấp một vài phương thức ít được sử dụng hơn cho phép chương trình sao lưu và đọc lại dữ liệu đang xử lý

```
public void mark(int readAheadLimit)
public void reset() throws IOException
public boolean markSupported()
```

- Không phải tất cả các luồng vào Java đều hỗ trợ việc đánh dấu (mark)
 vị trí luồng
 - Phương thức markSupported() để kiểm tra

Ex: Luồng vào



 Luồng vào từ màn hình

```
try{
    InputStream input = new InputStream(System.in);
    while((input.read()) != -1) {
        //do something with data...
    }
    input.close();
}catch(IOException e){
        System.out.println(e);
}
```

 Luồng vào từ file

Ex:



• Luồng vào từ socket

```
try{
                                         ServerSocket(số cổng);
      ServerSocket myServer =
                                   new
               clientSocket = myServer.accept();
      Socket
                    input
       InputStream
                               new
      DataInputStream(clientSocket.getInputStream());
      while((input.read()) !=
                    something
                               with data...
             //do
       input.close();
}catch(IOException e){
      System.out.println(e);
```

Filter Streams

• Vấn đề

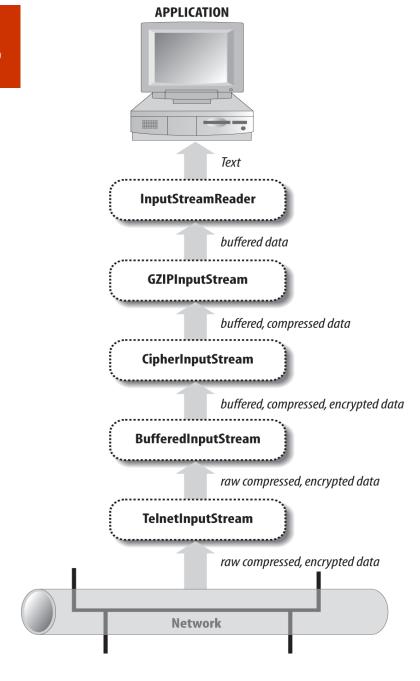
- Việc đọc và ghi file theo byte đơn lẻ hoặc nhóm
- Một số định dạng phổ biến int, float, string ...

Filter Stream

 Java cung cấp một số lớp cho phép gắn vào các luồng dữ liệu thô (byte) và chuyển đổi các dữ liệu byte thô này đến các định dạng khác và ngược lại

Filter ở 2 dạng

- Filter Streams: dữ liệu byte
- Reader và Writes: văn bản



Filter Stream



- Một filter được kết nối/gắn tới luồng I/O qua hàm khởi tạo của chúng
- Ex

```
FileInputStream fin = new FileInputStream("data.txt");
BufferedInputStream bin = new BufferedInputStream(fin);
```

```
InputStream in = new FileInputStream("data.txt");
in = new BufferedInputStream(in);
```

```
DataOutputStream dout = new DataOutputStream(new BufferedOutputStream(
    new FileOutputStream("data.txt")));
```

Buffered Streams



- Bộ nhớ đệm cho phép cải thiện đáng kể hiệu suất và tốc độ đọc/ghi
 - BufferedOutputStream lưu trữ dữ liệu trong một bộ đệm cho tới khi đầy hoặc luồng được giải phóng/đẩy (flush)
 - BufferedInputStream lưu trữ dữ liệu nhiều nhất có thể từ nguồn vào, giúp dữ liệu khả dụng gần như ngay lập tức khả dụng cho các lần yêu cầu read()
- Khởi tạo

```
public BufferedInputStream(InputStream in)
public BufferedInputStream(InputStream in, int bufferSize)
public BufferedOutputStream(OutputStream out)
public BufferedOutputStream(OutputStream out, int bufferSize)
```

 BufferedInputStream và BufferedOutputStream đều không có các phương thức của riêng mình

Ex: BufferedOutputStream (1)



Ra màn hình

Ra File

Ex: BufferedOutputStream (2)



Ra socket

Ex: BufferedInputStream (1)



Ex: BufferedInputStream (2)



Vào từ Socket

PrintStream



 Lớp PrintStream là luồng ra đầu tiên mà hầu hết các lập trình viên đều gặp vì System.out là một PrintStream

```
public PrintStream(OutputStream out)
public PrintStream(OutputStream out, boolean autoFlush)
```

- Ngoài các phương thức write(), flush() và close(), Lớp PrintStream còn có 9 phương thức nạp chồng print() và 10 phương thức nạp chồng println()
 - Boolean, char, int, long, float, double, char[]
 - String
 - Object
 - Println()

Data Stream



- Lớp DataInputStream và DataOutputStream cung cấp các phương thức để đọc và ghi các dữ liệu nguyên thủy và String dưới dạng nhị phân.
- Luồng đầu ra dữ liệu ghi là gì, luồng đầu vào dữ liệu có thể đọc được

```
public final void writeBoolean(boolean b) throws IOException public final void writeShort(int s) throws IOException public final void writeChar(int c) throws IOException public final void writeInt(int i) throws IOException public final void writeLong(long l) throws IOException public final void writeFloat(float f) throws IOException public final void writeFloat(float f) throws IOException public final void writeDouble(double d) throws IOException public final void writeChars(String s) throws IOException public final void writeBytes(String s) throws IOException public final void writeUTF(String s) throws IOException
```

Ex: DataOutputStream (1)



Ra màn hình

```
try{
          DataOutputStream output = new DataOutputStream(System.out);
          output.writeUTF("some thing output.close();
}catch(IOException e){
          System.out.println(e);
}
```

Ra File

```
try{
    DataOutputStream output = new DataOutputStream(new FileOutputStream("output.txt"));
    output.writeUTF("some thing to write");
    output.close();
}catch(IOException e){
    System.out.println(e);
}
```

Ex: DataOutputStream (2)



Ra socket

Ex: DataInputStream (1)



Vào từ bàn phím

• Vào từ File

Ex: DataInputStream (2)



Vào từ Socket

```
try{
                               new ServerSocket(số
                                                    cổng);
   ServerSocket
                 myServer =
   Socket clientSocket = myServer.accept();
                              new DataInputStream(
   DataInputStream input =
                                  clientSocket.getInputStream());
  String
                   input.readUTF();
           in =
   //do something
                    with
                          data...
   input.close();
}catch(IOException
   System.out.println(e);
```



2. Luồng ký tự

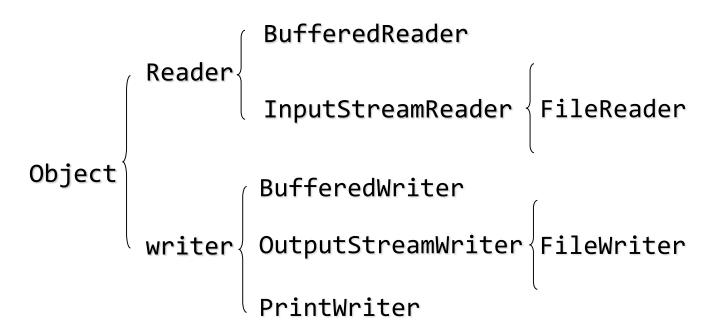
- Writers
- OutputStreamWriter

- Readers
- Filter Readers and Writer

Luồng ký tự



- Trên một số giao thức hiện đại (http),
 tồn tại nhiều định dạng mã hóa ký tự
 - Java UTF-16 Unicode
 - Big-5 Chinese
 - ISO 8859-9
- Do vậy, Java cung cấp một bản sao gần như hoàn chỉnh của các luồng I/O, được thiết kế để làm việc riêng với ký tự thay vì bytes
 - Reader vs InputStream
 - Writer vs OutputStream



Writer



• Lớp Writer là bản sao của lớp java.io.OutputStream

```
protected Writer()
protected Writer(Object lock)
public abstract void write(char[] text, int offset, int length)
    throws IOException
public void write(int c) throws IOException
public void write(char[] text) throws IOException
public void write(String s) throws IOException
public void write(String s, int offset, int length) throws
IOException
public abstract void flush() throws IOException
public abstract void close() throws IOException
```

OutputStreamWriter



- OutputStreamWriter là lớp con quan trọng nhất của của lớp Writer
- Hàm khởi tạo

```
public OutputStreamWriter(OutputStream out, String encoding)
    throws UnsupportedEncodingException
```

- Nếu không chỉ định định dạng mã hóa -> sử dụng định dạng mặc định của Java, UTF-8 trên Mac và Linux
- Phương thức trả về định dạng mã hóa

```
public String getEncoding()
```

Reader



Lóp Reader là bản sao của lớp java.io.InputStream

```
protected Reader()
protected Reader(Object lock)
public abstract int read(char[] text, int offset, int length)
    throws IOException
public int read() throws IOException
public int read(char[] text) throws IOException
public long skip(long n) throws IOException
public boolean ready()
public boolean markSupported()
public void mark(int readAheadLimit) throws IOException
public void reset() throws IOException
public abstract void close() throws IOException
```

InputStreamReader



- InputStreamReader là lớp con quan trọng nhất của lớp Reader
- Hàm khởi tạo

```
public InputStreamReader(InputStream in)
public InputStreamReader(InputStream in, String encoding)
    throws UnsupportedEncodingException
```

- Nếu không chỉ định định dạng mã hóa -> sử dụng định dạng mặc định của Java
- Nếu không xác định được định dạng mã hóa -> UnsupportedEncodingException

Filter Reader and Writer



- Một bộ lọc ký tự có thể được thêm, thành một lớp nằm trên luồng đọc/ghi ký tự
 - BufferedReader
 - BufferedWriter
 - LineNumberReader
 - PushbackReader
 - PrintWriter

Ex: BufferedWriter



• Ra màn hình

Ra File

Ra Socket

Ex:BufferedReader



Vào từ bàn phím

• Vào từ File

Vào từ Socket



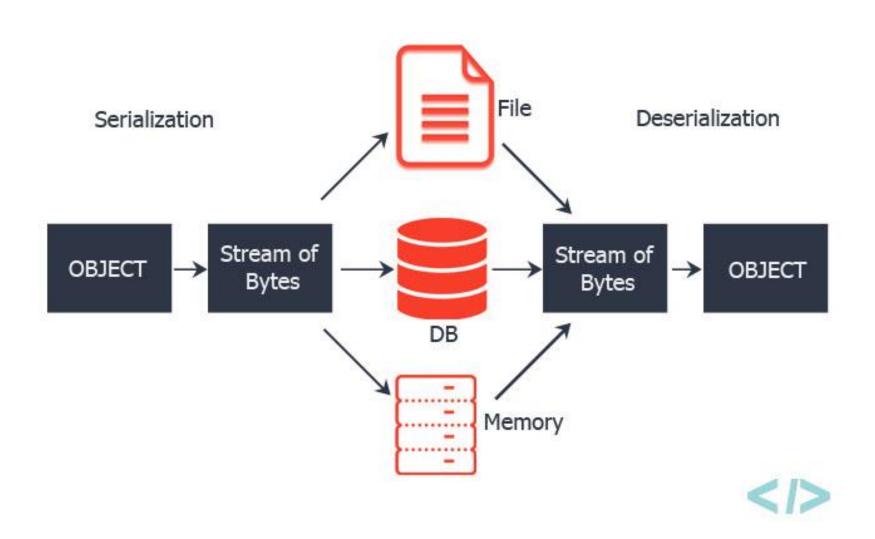
3. Luồng đối tượng

•

•

Serialization and Deserialization





Object Streams



ObjectInputStream

- Chịu trách nhiệm cho quá trình deserialization các đối tượng đã được tuần tự hóa hoặc dữ liệu nguyên thủy
- Giúp đọc đối tượng từ luồng vào
- Phương thức chính readObject() được sử dụng để deserialize đối tượng

ObjectOutputStream

- Chịu trách nhiệm cho quá trình serialization một đối tượng
- Giúp chuyển đổi trạng thái đối tượng vào luồng ra
- Phương thức chính writeObject() được sử dụng để serialize đối tượng

• Lưu ý

- Một đối tượng có thể ghi tới luồng ra (serializable) nếu cài đặt interface java.io.Serializable
- Interface <u>Serializable</u> là một interface đánh dấu (marker interface)
- Từ khóa Transient được sử dụng trong trường hợp muốn bỏ qua một số

DEMO



- Bytes Stream
- Data Stream
- Object Stream

- Console
- File

Tổng kết



• Luồng

- Luồng bytes
- Luồng ký tự
- Luồng dữ liệu nguyên thủy và String
- Luồng đối tượng

I/O

- Console / App
- File
- Socket

Exercise



- 1. Byte Stream
- 2. Character Stream
- 3. Data Stream
- 4. Object Stream (v)
- 5. ...

- 1. Tìm kiếm trong file, trong folder (tương tự notepad++)
- 2. Chương trình copy file sử dụng dòng lệnh theo định dạng [program name] [source file] [des file]
- 3. Đọc và ghi đối tượng
- 4. ..
- 5. .



Q & A