



Fast Track SE™

# Unit 8 : Các thư viện thường dùng trong JAVA

Giảng viên :Cao Le Thanh



FastTrack SE™

# Mục tiêu bài học

- ❖ Kết thúc bài học này bạn có khả năng
- ❖ Hiểu và sử dụng được các thư viện xử lý chuỗi , biểu thức chính quy
- ❖ Biết cách truy xuất luồng byte
- ❖ Biết cách truy xuất luồng character ( file văn bản )
- ❖ Biết cách truy xuất file nhị phân (random access)
- ❖ Xử lý ngoại lệ khi truy xuất luồng





FastTrack SE™

# Nội dung

- ❖ 1.Chuỗi và biểu thức chính quy
- ❖ 2.Luồng dữ liệu
- ❖ 3.Truy xuất luồng byte
- ❖ 4.Truy xuất file nhị phân
- ❖ 5.Truy xuất file văn bản
- ❖ 6.Quản lý ngoại lệ với luồng





# 1. Chuỗi ( string )

- ❑ String là xâu các ký tự.
  - ❖ String s = "Hello World";
- ❑ String là một class được xây dựng sẵn trong Java. String có rất nhiều phương thức giúp xử lý chuỗi một cách thuận tiện và hiệu quả.
- ❑ String là kiểu dữ liệu được sử dụng nhiều nhất trong lập trình
- ❑ Một số ký tự đặc biệt thường dùng

Ký tự	Hiển thị
\t	Ký tự tab
\r	Về đầu dòng
\n	Xuống dòng
\\	\
\"	"



## 1.2 Các thao tác chuỗi thường dùng

Phương thức	Mô tả
toLowerCase ()	Đổi in thường
toUpperCase ()	Đổi in hoa
trim()	Cắt các ký tự trắng 2 đầu chuỗi
length()	Lấy độ dài chuỗi
substring()	Lấy chuỗi con
charAt (index)	Lấy ký tự tại vị trí
replaceAll(find, replace)	Tìm kiếm và thay thế tất cả
split(separator)	Tách chuỗi thành mảng



## 1.3 Các thao tác thường dùng

Phương thức	Mô tả
equals()	So sánh bằng có phân biệt hoa/thường
equalsIgnoreCase()	So sánh bằng không phân biệt hoa/thường
contains()	Kiểm tra có chứa hay không
startsWith()	Kiểm tra có bắt đầu bởi hay không
endsWith ()	Kiểm tra có kết thúc bởi hay không
matches ()	So khớp với hay không?
indexOf()	Tìm vị trí xuất hiện đầu tiên của chuỗi con
lastIndexOf()	Tìm vị trí xuất hiện cuối cùng của chuỗi con



## 1.4 Discussion & demo

- ❖ Trao đổi và hỏi đáp về thao tác với chuỗi
- ❖ Cách ngắn nhất để xóa các khoảng trắng , ví dụ : “ Phạm Trung Hải” thành “Phạm Trung Hải” ?
- ❖ Cách nhanh nhất để in hoa tất cả các từ trong chuỗi , ví dụ : “phạm trung hải” thành “Phạm Trung Hải”
- ❖ Cách nhanh nhất để tạo định dạng chuỗi ( mở rộng sang format ) ?
- ❖ Trong nhiều trường hợp cần xử lý các bài toán phức tạp hơn thì cần biết biểu thức chính quy



## 2.1 Biểu thức chính quy

Bạn có biết các chuỗi sau đây biểu diễn những gì hay không?

❖ [teo@fpt.edu.vn](mailto:teo@fpt.edu.vn)

- ❖ 54-P6-6661
- ❖ 54-P6-666.01
- ❖ 0913745789
- ❖ 192.168.11.200

1. Bạn có biết tại sao bạn nhận ra chúng không?
2. Làm thế nào để máy tính cũng có thể nhận ra như bạn?



## 2.2 Biểu thức chính quy

- ❑ Máy tính có thể nhận dạng như chúng ta nếu chúng ta cung cấp qui luật nhận dạng cho chúng. Biểu thức chính qui cung cấp qui luật nhận dạng chuỗi cho máy tính.
- ❑ Biểu thức chính qui là một chuỗi mẫu được sử dụng để qui định dạng thức của các chuỗi. Nếu một chuỗi nào đó phù hợp với mẫu dạng thức thì chuỗi đó được gọi là so khớp (hay đối sánh).
- ❑ Ví dụ: **[0-9]{3,7}**: Biểu thức chính qui này so khớp các chuỗi từ 3 đến 7 ký tự số.
  - ❖ [0-9]: đại diện cho 1 ký tự số
  - ❖ {3,7}: đại diện cho số lần xuất hiện (ít nhất 3 nhiều nhất 7)



## 2.3 Ví dụ

```
Scanner scanner = new Scanner(System.in);
System.out.print("Số mobile: ");
String mobile = scanner.nextLine();
String pattern = "0[0-9]{9,10}";
if(mobile.matches(pattern)){
    System.out.println("Bạn đã nhập đúng số mobile");
}
else{
    System.out.println("Bạn đã nhập không đúng số mobile");
}
```

Biểu thức  
chính qui

Kiểm tra mobile có so  
khớp với pattern không?

s.matches(regex)



## 2.4 Xây dựng biểu thức

### Regular Expression

Ký tự đại diện	
[xyz]	đại diện một ký tự x, y hay z
[ad-f]	đại diện một ký tự a, d, e hay f
[^xyz]	đại diện ký tự không thuộc [xyz]
\d	tương đương [0-9]
\w	tương đương [0-9a-zA-Z_]
\D	tương đương [^\d]
\W	tương đương [^\w]
\s	đại diện ký tự trắng (\r\n\t\f)
.	đại diện ký tự bất kỳ
^	chỉ ra mẫu bắt đầu
\$	chỉ ra mẫu kết thúc
\\", \., \\$, ^	đại diện '\\", '.', '\$' hay '^'

{M,N}	ít nhất M, nhiều nhất N lần
{N}	Đúng N lần
?	0-1
*	0-N
+	1-N
Không	1



[0-9]{3, 7}



## 2.5 Biểu thức thường dùng

- ❖  Số CMND
  - ❖ ❖ [0-9]{9}
- ❖  Số điện thoại di động việt nam
  - ❖ ❖ 0\d{9,10}
- ❖  Số xe máy sài gòn
  - ❖ ❖ 5\d-[A-Z]\d-((\d{4})|(\d{3}\.\d{2}))
- ❖  Địa chỉ email
  - ❖ ❖ \w+\@\w+(\.\w){1,2}



## 2.6 Demo

```
Scanner in = new Scanner(System.in);

System.out.print("Email: ");
String email = in.nextLine();

System.out.print("số điện thoại ");
String phone = in.nextLine();

String reEmail = "\\w+@\\w+\\.\\w+";
if(!email.matches(reEmail)) {
    System.out.println("Không ");
}

String rePhone = "0543\\d{6}";
if(!phone.matches(rePhone)) {
    System.out.println("Không phải số điện thoại ở Huế !");
}
```

Email đơn giản

Số điện thoại để bàn ở Huế



### 3.1 Các loại luồng dữ liệu

Các hoạt động **nhập/xuất** dữ liệu (nhập dữ liệu từ bàn phím, đọc dữ liệu từ file, ghi dữ liệu màn hình, ghi ra file, ghi ra đĩa, ghi ra máy in...) đều được gọi là **luồng** (stream).

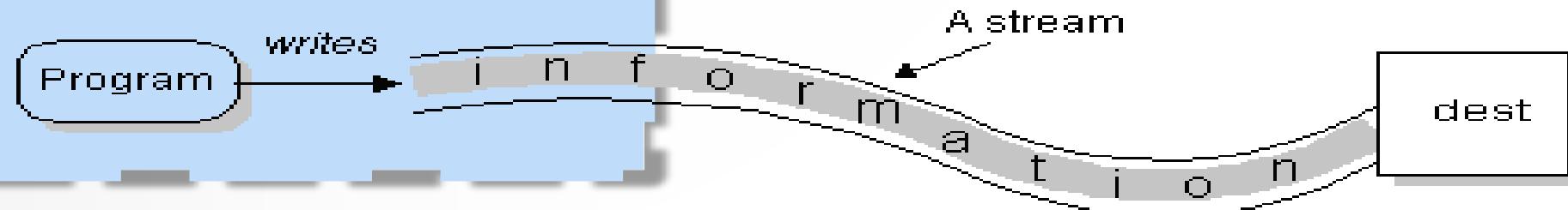
Tất cả các luồng đều có **chung một nguyên tắc** hoạt động ngay cả khi chúng được gắn kết với các thiết bị vật lý khác nhau.

## 3.2 Các loại luồng dữ liệu

Input Streams – lấy dữ liệu từ các nguồn: Files, Buffers và Sockets



Output Streams – ghi dữ liệu vào Files, Buffers in Memory, and Sockets





### 3.3 Các loại luồng dữ liệu



Java™

#### Luồng byte

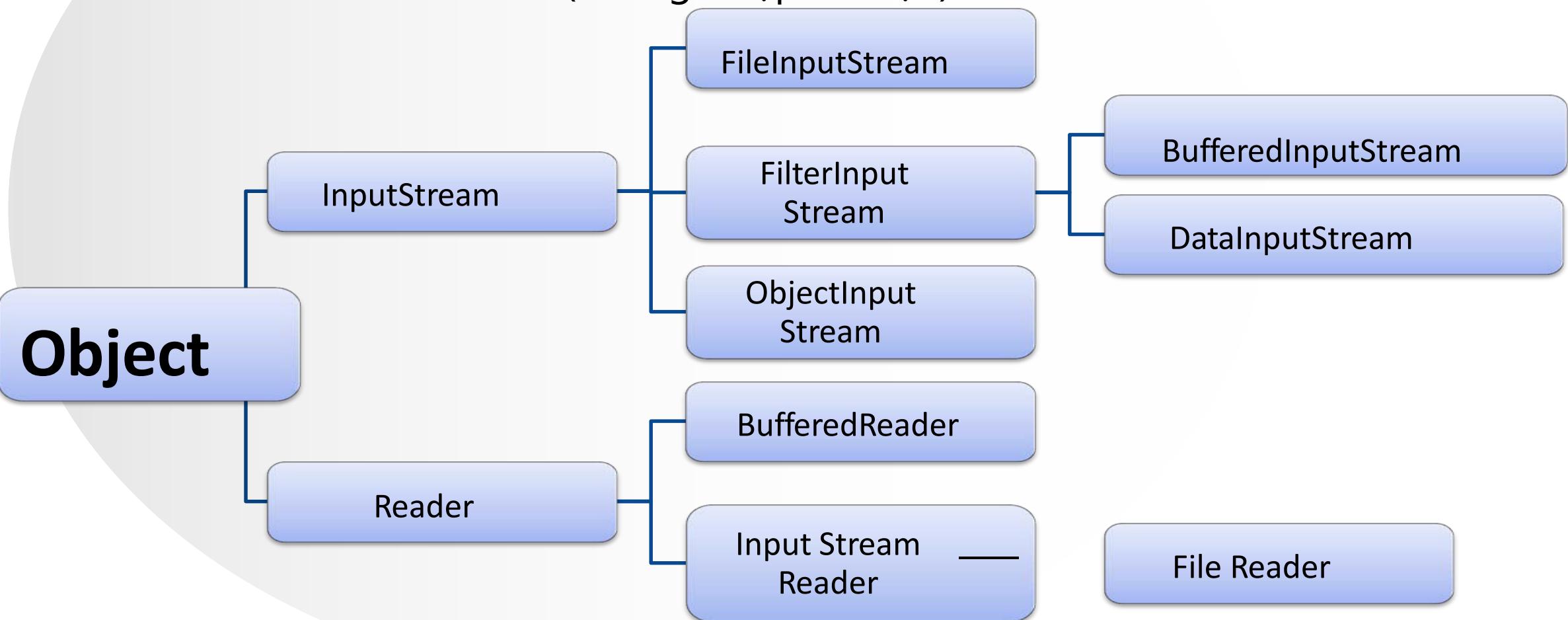
- Hỗ trợ việc xuất nhập dữ liệu trên byte,
- Thường được dùng khi đọc ghi dữ liệu nhị phân.

#### Luồng character

- Luồng character (ký tự) được thiết kế hỗ trợ việc xuất nhập dữ liệu kiểu ký tự (Unicode)

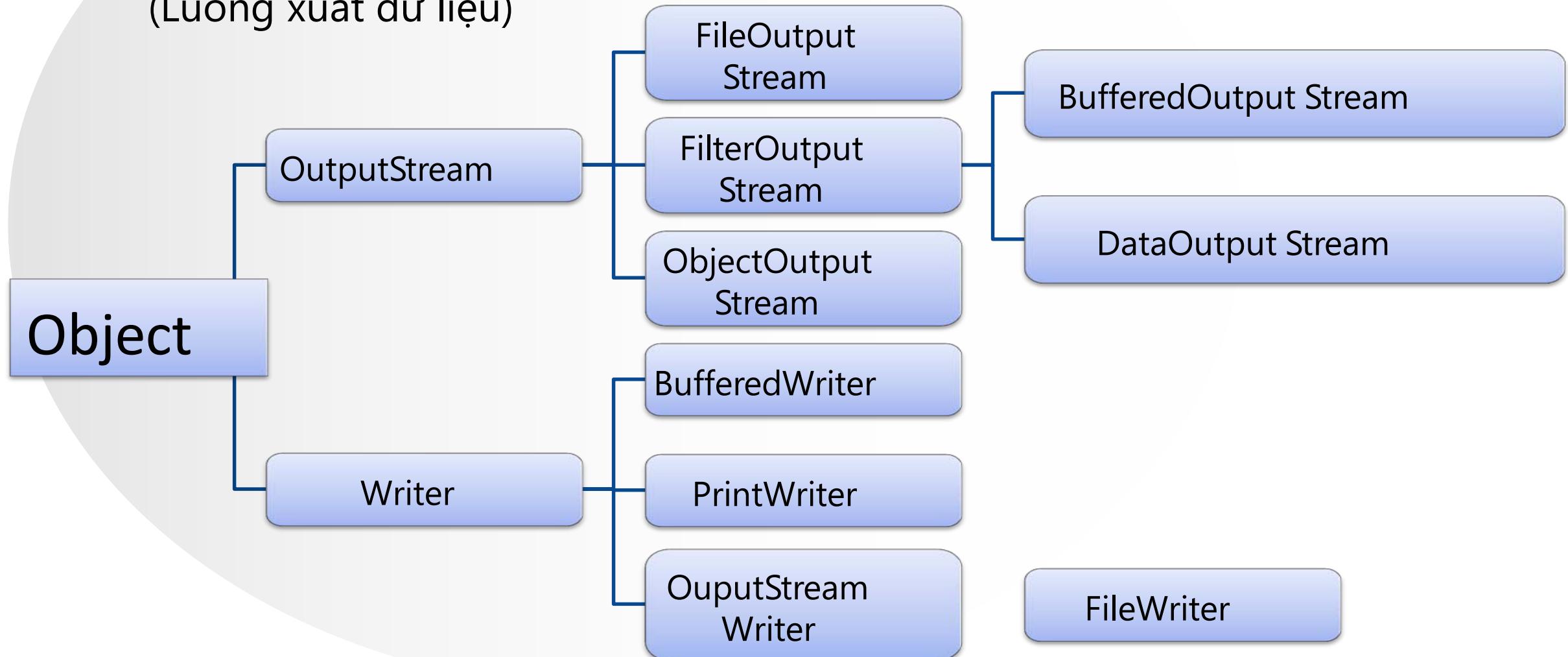
### 3.4 Các loại luồng dữ liệu

#### Kiến trúc Input Stream (Luồng nhập dữ liệu)



### 3.5 Các loại luồng dữ liệu

Kiến trúc OutputStream  
(Luồng xuất dữ liệu)





## 3.6 Các loại luồng dữ liệu

### Luồng byte

- Dữ liệu dạng nhị phân
- 2 class abstract:
  - InputStream
  - OutputStream

### Luồng character

- Dữ liệu dạng ký tự Unicode
- 2 class abstract:
  - Reader
  - Writer



## 3.7 Các loại luồng dữ liệu

### Các thao tác xử lý dữ liệu:

- import java.io.\*
- Tạo đối tượng luồng và liên kết với nguồn dữ liệu
- Thao tác dữ liệu (đọc hoặc ghi hoặc cả đọc và ghi)
- Đóng luồng.



## 3.8 Xử lý nhập xuất

Sử dụng luồng mỗi byte để nhập  
xuất dữ liệu

Tất cả các luồng byte được kế thừa  
từ 2 class:

Có nhiều class luồng byte

Chúng khác nhau về cách thức khởi tạo  
nhưng cách thức hoạt động là giống  
nhau.

- ▶ Input Stream Output Stream
- ▶ File Input Stream File Output Stream



## 4.1 Luồng byte

Sử dụng luồng byte trong các trường hợp:

**Nhập xuất kiểu dữ liệu nguyên thủy:**

Kiểu int, float, double, String, boolean...

**Nhập xuất kiểu dữ liệu khác:** Kiểu object



## 4.2 Xử lý nhập xuất dữ liệu bằng luồng byte

### Các class Byte Stream

Byte Stream Class	Meaning
BufferedInputStream	Buffered input stream
BufferedOutputStream	Buffered output stream
ByteArrayInputStream	Input stream that reads from a byte array
ByteArrayOutputStream	Output stream that writes to a byte array
DataInputStream	An input stream that contains methods for reading the Java standard data types
DataOutputStream	An output stream that contains methods for writing the Java standard data types
FileInputStream	Input stream that reads from a file
FileOutputStream	Output stream that writes to a file
FilterInputStream	Implements <b>InputStream</b>
FilterOutputStream	Implements <b>OutputStream</b>
InputStream	Abstract class that describes stream input
ObjectInputStream	Input stream for objects
ObjectOutputStream	Output stream for objects
OutputStream	Abstract class that describes stream output
PipedInputStream	Input pipe
PipedOutputStream	Output pipe
PrintStream	Output stream that contains <b>print( )</b> and <b>println( )</b>
PushbackInputStream	Input stream that allows bytes to be returned to the stream
SequenceInputStream	Input stream that is a combination of two or more input streams that will be read sequentially, one after the other



## 4.3 Xử lý nhập xuất dữ liệu bằng luồng byte

### Ví dụ 1: Tạo file 'file1.dat' và ghi dữ liệu

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

public class Example1 {

    public static void main(String[] args) throws IOException {
        FileOutputStream fos = new FileOutputStream("file1.dat");
        String text = "The quick brown fox jumped over the lazy dog";
        byte[] textAsBytes = text.getBytes();
        fos.write(textAsBytes);
    }
}
```



## 4.4 Xử lý nhập xuất dữ liệu bằng luồng byte

### Ví dụ 2: Đọc thông tin từ file 'file1.dat' và in ra màn hình

```
public class Example2 {  
  
    public static void main(String[] args) throws IOException {  
        FileInputStream fis = new FileInputStream("file1.dat");  
        int c;  
        while ((c = fis.read()) != -1) {  
            System.out.print((char) c);  
        }  
        fis.close();  
    }  
}
```



## 4.5 Xử lý nhập xuất dữ liệu bằng luồng byte

*Đọc, ghi dữ liệu nhị phân (binary data)*

Khi muốn tạo file chứa các kiểu dữ liệu như short, int, long, float, double, String, boolean... thì sử dụng 2 class:

Class **DataInputStream**  
xử lý việc nhập dữ liệu

Class **DataOutputStream**  
xử lý việc xuất dữ liệu



## 4.6 Xử lý nhập xuất dữ liệu bằng luồng byte

Một số phương thức xử lý dữ liệu nhị phân của class

### DataOutputStream

Output Method	Purpose
void writeBoolean(boolean val)	Writes the boolean specified by val.
void writeByte(int val)	Writes the low-order byte specified by val.
void writeChar(int val)	Writes the value specified by val as a char.
void writeDouble(double val)	Writes the double specified by val.
void writeFloat(float val)	Writes the float specified by val.
void writeInt(int val)	Writes the int specified by val.
void writeLong(long val)	Writes the long specified by val.
void writeShort(int val)	Writes the value specified by val as a short.



## 4.7 Xử lý nhập xuất dữ liệu bằng luồng byte

### Ví dụ 1: Ghi dữ liệu

```
public class DatastreamOutput {  
    public static void main(String[] args) throws IOException {  
        FileOutputStream fos = new FileOutputStream("filedata.dat");  
        DataOutputStream dos = new DataOutputStream(fos);  
        final int NUMBER = 5;  
        dos.writeInt(NUMBER);  
        for (int i = 0; i <= NUMBER; i++) {  
            dos.writeInt(i);  
        }  
        dos.writeUTF("Hello !");  
        dos.writeDouble(100.75);  
        dos.flush();  
        dos.close();  
    }  
}
```



## 4.8 Xử lý nhập xuất dữ liệu bằng luồng byte

### Ví dụ 2: Đọc dữ liệu

```
public class DataInputStream {  
    public static void main(String[] args) throws IOException {  
        FileInputStream fis = new FileInputStream("filedata.dat");  
        DataInputStream dis = new DataInputStream(fis);  
        int items = dis.readInt();  
        for (int i = 0; i <= items; i++) {  
            int n = dis.readInt();  
            System.out.print(n + " ");  
        }  
        System.out.println(dis.readUTF());  
        System.out.println(dis.readDouble());  
        dis.close();  
    }  
}
```



## 4.9 Xử lý nhập xuất dữ liệu bằng luồng byte

### Đọc, ghi dữ liệu kiểu object

```
public class Stock implements Serializable {  
    private int id;  
    private String desc;  
    private double price;  
    private int quantity;  
    public Stock(int id, String desc, double price, int quantity) {  
        this.id = id;  
        this.desc = desc;  
        this.price = price;  
        this.quantity = quantity;  
    }  
    public String toString() {  
        return (id+" "+desc + " " + price + " " + quantity);  
    }  
}
```



## 4.10 Xử lý nhập xuất dữ liệu bằng luồng byte

### Ví dụ 3: Đọc, ghi dữ liệu kiểu object

```
public class ObjectExampleWrite {
    public static void main(String[] args) throws
        IOException, ClassNotFoundException {
        FileOutputStream fos = new FileOutputStream("fileobject.dat");
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        Stock[] stocks = {new Stock(1001, "CD ROM", 100.00, 20),
            new Stock(1002, "DRAM", 75.00, 30),
            new Stock(1003, "P4 Processor", 300.00, 100),
            new Stock(1004, "Canon Jet", 80.00, 10),
            new Stock(1005, "HP Scanner", 75.00, 90)};
        //Ghi mang doi tuong vao file 'fileobject.dat'
        oos.writeObject(stocks);
        oos.close();
    }
}
```



## 4.11 Xử lý nhập xuất dữ liệu bằng luồng byte

### Ví dụ 2: Đọc, ghi dữ liệu kiểu object

```
public static void main(String[] args) {  
    FileInputStream fis = null;  
    ObjectInputStream ois = null;  
    try {  
        fis = new FileInputStream("fileobject.dat");  
        ois = new ObjectInputStream(fis);  
        Stock[] stocks1 = (Stock[]) ois.readObject();  
        System.out.println("Doc tu file: ");  
        for (Stock s : stocks1) {  
            System.out.println(s);  
        }  
        ois.close(); fis.close();  
    } catch (Exception e) {  
        System.out.println("Co loi: " + e);  
    }  
}
```



## 5.1 Truy cập file ngẫu nhiên

- Sử dụng object RandomAccessFile để truy cập ngẫu nhiên nội dung một file.
- RandomAccessFile là class thực thi 2 interface là DataInput và DataOutput trong đó có định nghĩa các phương thức input/output.

Dùng phương thức :

- **seek(vị\_trí)**: để di chuyển con trỏ file tới vị\_trí mới.
- **seek(0)**: Di chuyển con trỏ tới đầu file
- **seek(file.length())**: Di chuyển con trỏ tới cuối file



## 5.2 Truy cập file ngẫu nhiên

```
try {
    RandomAccessFile file = new RandomAccessFile("rand.txt", "rw");
    file.writeChar('a');file.writeInt(100);file.writeDouble(8.75);
    file.seek(0); //Dich chuyen con tro ve dau file va doc du lieu
    System.out.println(file.readChar());
    System.out.println(file.readInt());
    System.out.println(file.readDouble());
    file.seek(2); //Dich chuyen con tro file vao vi tri thu 2
    System.out.println("Vi tri thu 2: " + file.readInt());
    file.seek(file.length()); //Dich chuyen con tro toi cuoi file
    file.writeBoolean(true);
    file.seek(4); //Dich chuyen con tro file vao vi tri thu 4
    System.out.println("Vi tri thu 4: " + file.readBoolean());
    file.close();
} catch (Exception e) {
    System.out.println("Co loi: " + e);
}
```

## 6.1 Xử lý nhập xuất dữ liệu bằng luồng character

- ❖ Luồng byte rất mạnh mẽ và linh hoạt. Tuy nhiên nếu bạn muốn **lưu trữ file** chứa **văn bản Unicode** thì luồng **character** là lựa chọn **tốt nhất** vì ưu điểm của luồng character là nó thao tác trực tiếp trên ký tự Unicode.

Tất cả các luồng character đều  
được kế thừa từ 2 class abstract:

Reader

Writer



## Các class:

# 6.2 Xử lý nhập xuất dữ liệu bằng luồng character

Character Stream Class	Meaning
BufferedReader	Buffered input character stream
BufferedWriter	Buffered output character stream
CharArrayReader	Input stream that reads from a character array
CharArrayWriter	Output stream that writes to a character array
FileReader	Input stream that reads from a file
FileWriter	Output stream that writes to a file
FilterReader	Filtered reader
FilterWriter	Filtered writer
InputStreamReader	Input stream that translates bytes to characters
LineNumberReader	Input stream that counts lines
OutputStreamWriter	Output stream that translates characters to bytes
PipedReader	Input pipe
PipedWriter	Output pipe
PrintWriter	Output stream that contains print( ) and println( )
PushbackReader	Input stream that allows characters to be returned to the input stream
Reader	Abstract class that describes character stream input
StringReader	Input stream that reads from a string
StringWriter	Output stream that writes to a string
Writer	Abstract class that describes character stream output

## 6.3 Xử lý nhập xuất dữ liệu bằng luồng character

### Ví dụ 1

```
File filename = new File("first.txt");
try {
    FileWriter out = new FileWriter(filename);
    out.write("Doc ghi du lieu trong Java!");
    out.write("\n"); //GHI VAO FILE
    out.write("Su dung Stream Character");
    out.close();
    //DOC TU FILE TEXT
    FileReader input = new FileReader(filename);
    System.out.println("Doc tu file first.txt:");
    int ch = input.read();
    while (ch != -1) {
        System.out.print((char) ch);
        ch = input.read(); //DOC TU FILE
    }
} catch (Exception e) {
```



## 6.4 Xử lý nhập xuất dữ liệu bằng luồng character

### Ví dụ 2: Ghi vào file mảng String Student:

```
public void writeToFileText(String FileName) throws IOException {  
    FileWriter fw = new FileWriter(FileName);  
    BufferedWriter bw = new BufferedWriter(fw);  
    for (int i = 0; i < this.count; i++) {  
        String temp = Students[i].toString();  
        bw.write(temp); //GHI VAO FILE  
    }  
    bw.flush();  
    bw.close();  
}
```



## 6.5 Xử lý nhập xuất dữ liệu bằng luồng character

### Ví dụ 3: Đọc dữ liệu từ file và hiển thị ra màn hình:

```
public void readFromFileText(String FileName) throws IOException,  
    ClassNotFoundException {  
    FileReader frr = new FileReader(FileName);  
    BufferedReader br = new BufferedReader(frr);  
    String text;  
    while ((text = br.readLine()) != null) {  
        System.out.println(text);  
    }  
    br.close();  
}
```



## 7.1 Sử dụng try... catch trong nhập xuất

- ❖ Khi **input/output** dữ liệu, có những **ngoại lệ 'checked'** nên bắt buộc phải **catch** khi viết **code**, thông thường các ngoại lệ đó là:



### IOException

- ❖ FileNotFoundException
- ❖ EOFException
- ❖ NotSerializableException
- ❖ ...



## 7.2 Sử dụng try... catch trong nhập xuất

```
FileOutputStream fos = null;
try {
    fos = new FileOutputStream("file1.dat");
    String text = "The quick brown fox jumped over the lazy dog";
    byte[] textAsBytes = text.getBytes();
    fos.write(textAsBytes);
} catch (FileNotFoundException ex) {
    System.out.println("Khong tim thay file: " + ex);
} catch (IOException ex) {
    System.out.println("Co loi : " + ex);
} finally {
    try {
        fos.close();
    } catch (IOException ex) {
        System.out.println("Co loi: "+ex);
    }
}
```



## Tổng kết bài học

- Chuỗi và biểu thức chính quy
- Các loại luồng dữ liệu
- Xử lý nhập xuất bằng luồng byte
- Truy cập file ngẫu nhiên
- Xử lý nhập xuất bằng luồng character
- Sử dụng try... catch trong nhập/xuất