Ministry of SMEs and Startups × KOSME Korea SMEs and Startups Agency × LIKELION

File handling

Nguyễn Anh Tuấn

K TECH COLLEGE

Nội dung bài giảng

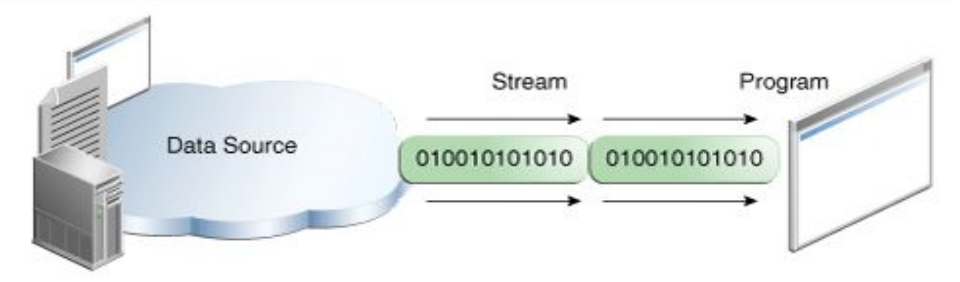
1 File handling

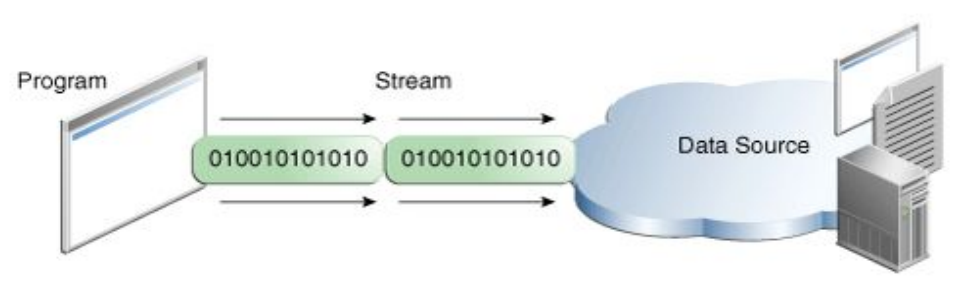
2 Byte Streams

3 Character Streams

I/O Stream

- **Mô tả**:
  - Luồng vào ra trong Java hay Input/Output (I/O) trong java được sử dụng để xử lý đầu vào và đầu ra trong java.
  - **Stream** là một dòng liên tục, có thứ tự các bytes dữ liệu chảy giữa chương trình và thiết bị ngoại vi
  - Nếu dòng dữ liệu trong Stream có hướng chảy từ thiết bị ngoại vi vào chương trình thì ta nói đây là **Stream nhập** (Input Stream), ngược lại gọi là **Stream xuất** (Output Stream)





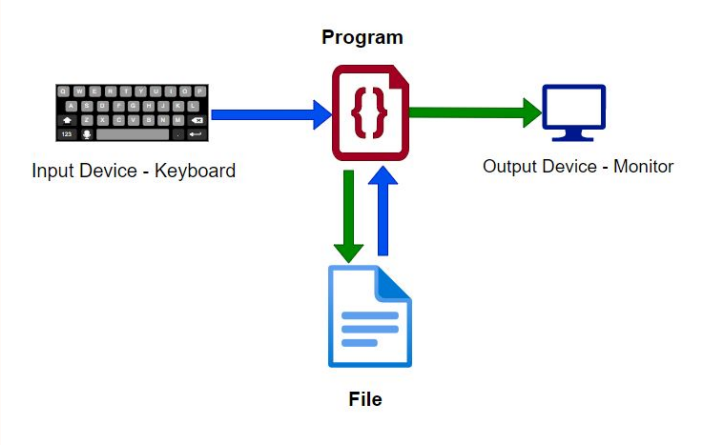
File handling

File handling

- **Mô tả**:
  - File handling trong Java là một quá trình làm việc với các file (tệp) và thư mục (directory) trên hệ thống file. Nó bao gồm các hoạt động như tạo, đọc, ghi, và xóa file hoặc thư mục.

File handling

- **Các ứng dụng phổ biến sử dụng File handling**:
  - Sao lưu, khôi phục dữ liệu
  - Nhật ký hệ thống
  - Nhập, xuất báo cáo
  - Xử lý hoá đơn
  - Bảo mật, mã hoá
  - Lưu trữ, quản lý file đa phương tiện



File handling

- **Các gói thư viện mà Java cung cấp**:
  - **java.io**:
    - Cung cấp các lớp cơ bản để làm việc với file và stream.
    - Các lớp phổ biến bao gồm File, FileReader, FileWriter, BufferedReader, BufferedWriter, InputStream, và OutputStream.
  - **java.nio.file**:
    - Phần mở rộng của java.nio (Non-blocking I/O), cung cấp các lớp và giao diện nâng cao để làm việc với file và thư mục.
    - Các lớp phổ biến bao gồm Files, Paths, và Path.

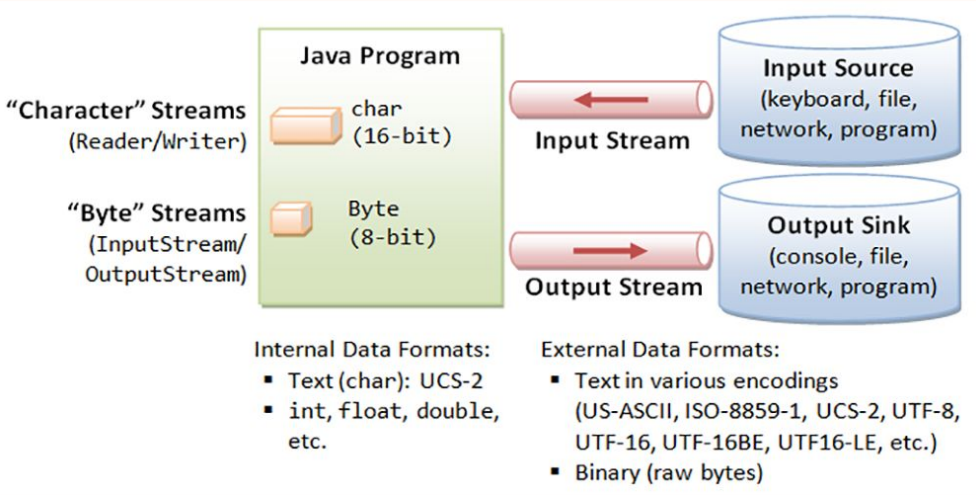
Byte Stream

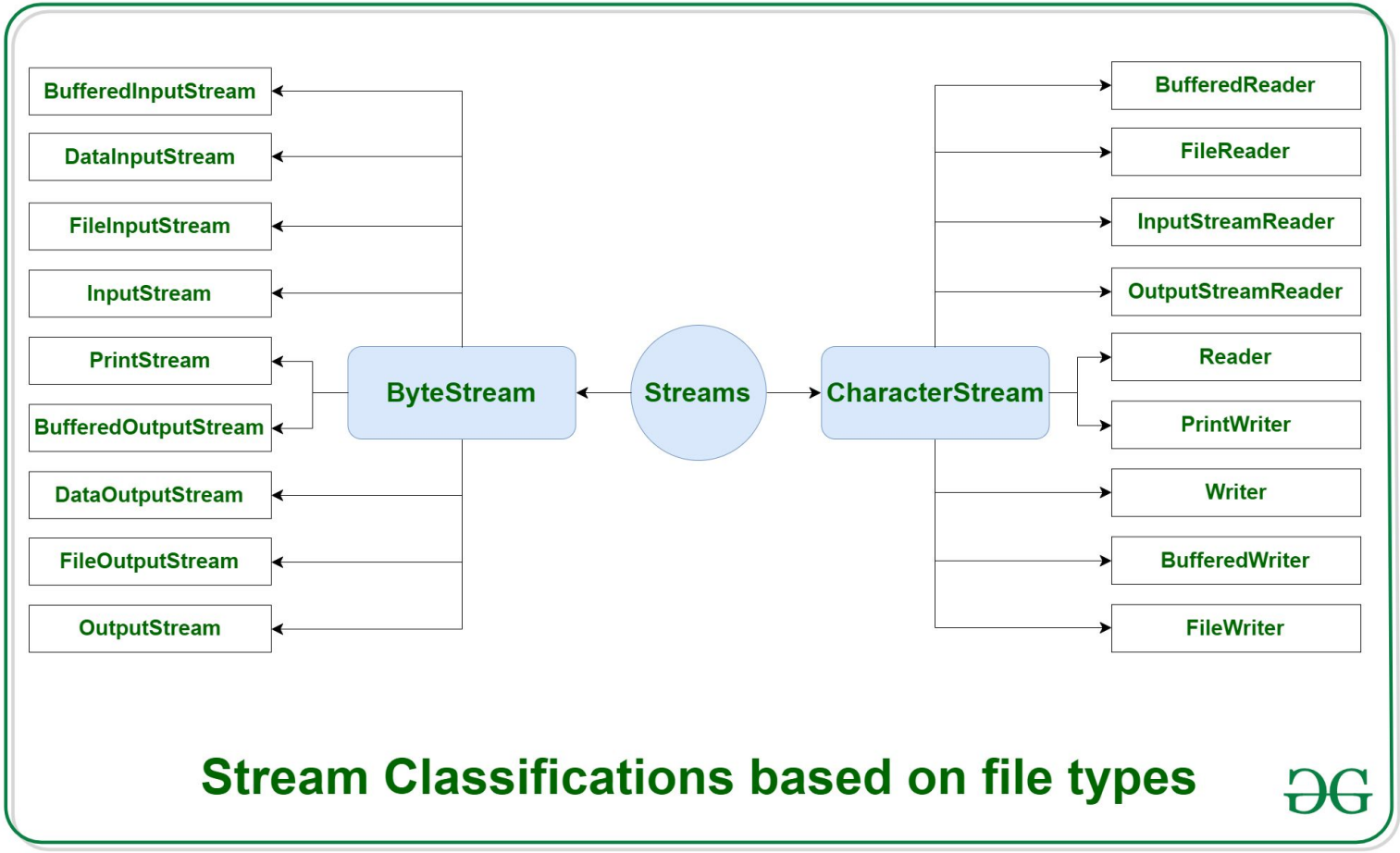
Khái niệm chung về Streams

- **Khái niệm Streams**:
  - **Streams** trong Java là một khái niệm trừu tượng để xử lý các luồng dữ liệu. Chúng cung cấp một cách chung để đọc và ghi dữ liệu từ các nguồn khác nhau, như file, mảng byte, hay socket mạng. Có hai loại streams chính:

➔ **Character Streams**: Xử lý dữ liệu theo đơn vị ký tự (16-bit). Các lớp của character stream bao gồm Reader và Writer.

➔ **Byte Streams**: Xử lý dữ liệu theo đơn vị byte (8-bit). Các lớp của byte stream bao gồm InputStream và OutputStream.

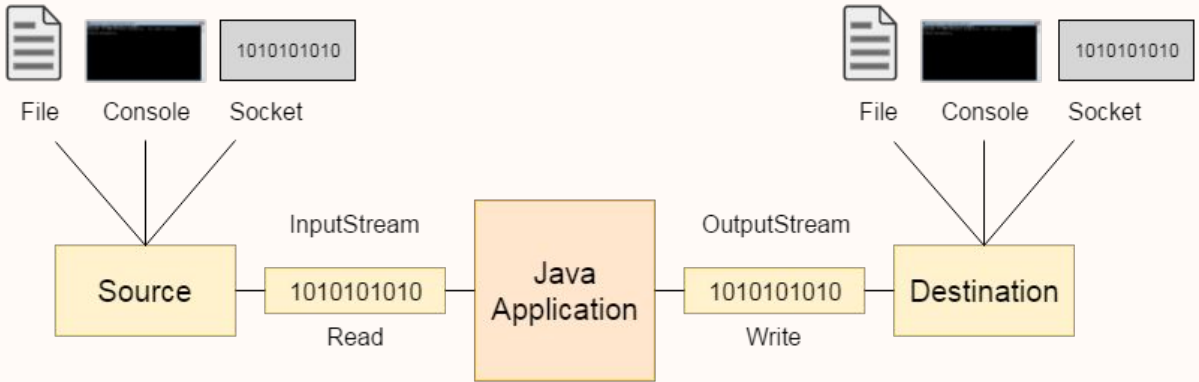



BufferedInputStream
DataInputStream
FileInputStream
InputStream
PrintStream
BufferedOutputStream
DataOutputStream
FileOutputStream
OutputStream
ByteStream
Streams
CharacterStream
BufferedReader
FileReader
InputStreamReader
OutputStreamReader
Reader
PrintWriter
Writer
BufferedWriter
FileWriter
Stream Classifications based on file types

Byte Stream

- **Định nghĩa**:
  - **Byte stream** là các lớp trong Java xử lý dữ liệu theo từng byte, được sử dụng để đọc và ghi dữ liệu nhị phân (binary data) như các file hình ảnh, file âm thanh, hoặc các file khác không phải là văn bản.

- **2 lớp super class**:
  - **Lớp trừu tượng InputStream**: được sử dụng để đọc dữ liệu từ một nguồn (source).
  - **Lớp trừu tượng OutputStream**: được sử dụng để ghi dữ liệu đến đích (destination)



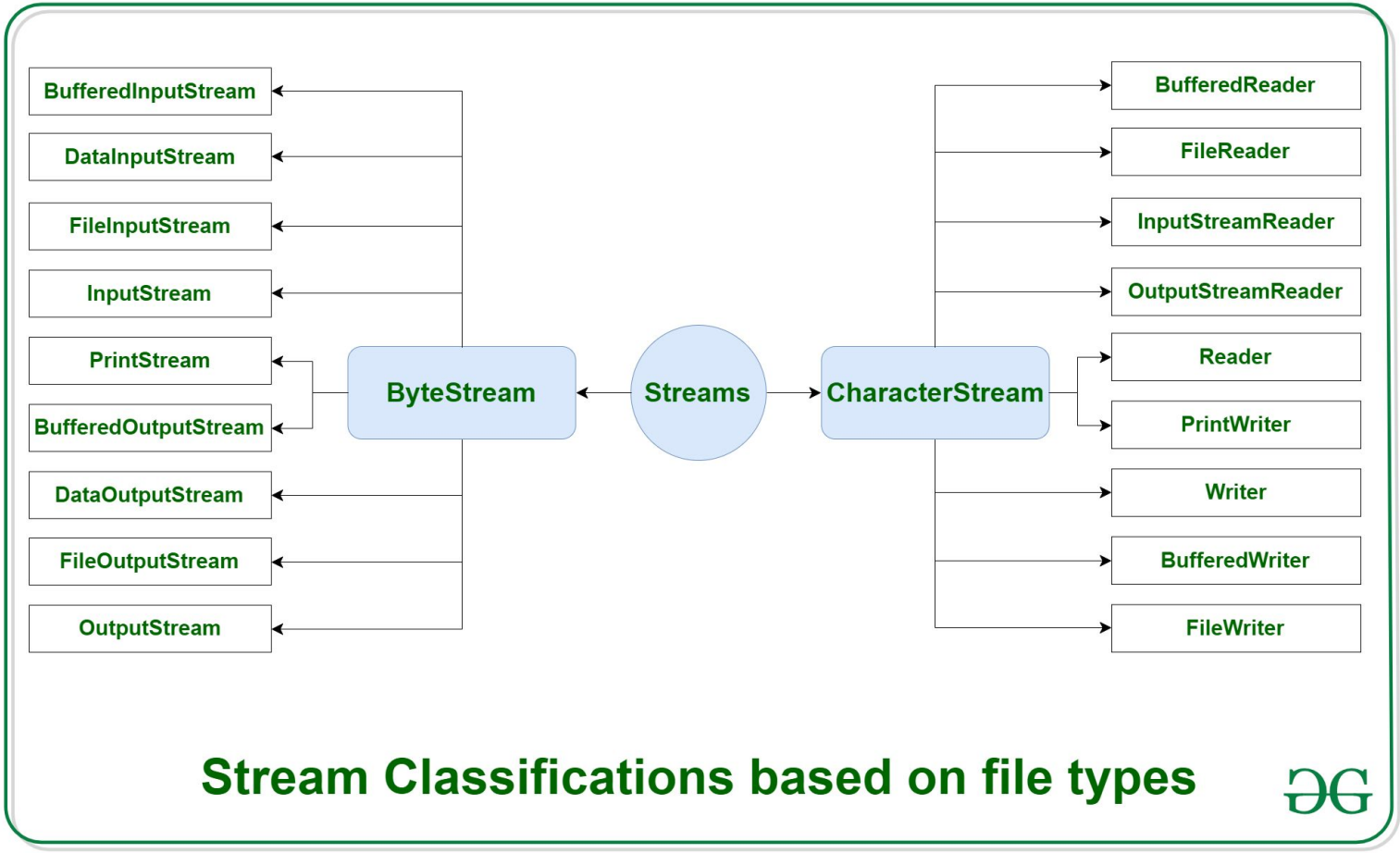
FileOutputStream

```
public static void main(String[] args) {
        String destFile = "destination.txt";      // Đường dẫn đến file đích

        FileOutputStream outputStream = null;     // Khai báo FileOutputStream

        // Mở (hoặc tạo) file đích bằng FileOutputStream
        outputStream = new FileOutputStream(destFile);

        // Chuỗi dữ liệu cần ghi
        String content = "Hello World";

        // Chuyển đổi chuỗi thành mảng byte và ghi vào file
        outputStream.write(content.getBytes());

        System.out.println("Đã ghi file thành công.");
}
```

FileInputStream

```
public static void main(String[] args) {
        String sourceFile = "source.txt";          // Đường dẫn đến file nguồn

        FileInputStream inputStream = null;// Khai báo FileInputStream

        // Mở file nguồn bằng FileInputStream
        inputStream = new FileInputStream(sourceFile);

        // Buffer để đọc dữ liệu
        byte[] buffer = new byte[1024];
        int bytesRead;

        // Đọc dữ liệu từ file và in ra màn hình
        while ((bytesRead = inputStream.read(buffer)) != -1) {
                System.out.print(new String(buffer, 0, bytesRead));
        }
```

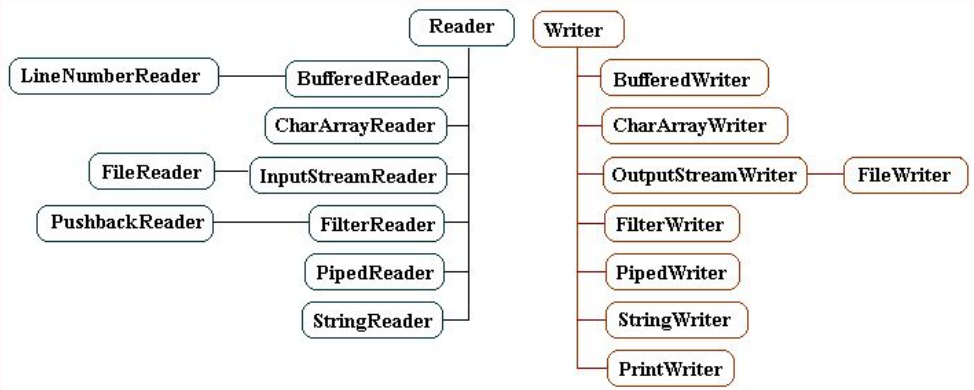
Character Stream


BufferedInputStream
DataInputStream
FileInputStream
InputStream
PrintStream
BufferedOutputStream
DataOutputStream
FileOutputStream
OutputStream
ByteStream
Streams
CharacterStream
BufferedReader
FileReader
InputStreamReader
OutputStreamReader
Reader
PrintWriter
Writer
BufferedWriter
FileWriter
Stream Classifications based on file types

Character Stream

- **Định nghĩa**:
  - **Character stream** là các lớp được thiết kế để đọc và ghi dữ liệu ký tự (chữ).
  - **Character streams** xử lý dữ liệu theo từng ký tự Unicode 16-bit, giúp xử lý văn bản hiệu quả hơn, đặc biệt với các ngôn ngữ khác nhau và các ký tự không phải ASCII.

- **2 lớp super class**:
  - **Lớp trừu tượng Reader**: Được sử dụng để đọc dữ liệu từ một nguồn (source).
  - **Lớp trừu tượng Writer**: Được sử dụng để ghi dữ liệu đến đích (destination)



FileWriter

```
public static void main(String[] args) {
        String destFile = "destination.txt";      // Đường dẫn đến file đích

        FileWriter writer = null;  // Khai báo FileWriter

        // Mở (hoặc tạo) file đích bằng FileWriter
        writer = new FileWriter(destFile);

        // Chuỗi dữ liệu cần ghi
        String content = "Hello World";

        // Ghi chuỗi vào file
        writer.write(content);

        System.out.println("Đã ghi file thành công.");
}
```

FileReader

```
public static void main(String[] args) {
        String sourceFile = "source.txt";          // Đường dẫn đến file nguồn

        FileReader reader = null;                  // Khai báo FileReader

        // Mở file nguồn bằng FileReader
        reader = new FileReader(sourceFile);

        // Biến để lưu trữ ký tự đọc được
        int character;

        // Đọc từng ký tự từ file và in ra màn hình
        while ((character = reader.read()) != -1) {
                System.out.print((char) character);
        }
}
```

BufferedWriter

```
public static void main(String[] args) {
        String destFile = "destination.txt";      // Đường dẫn đến file đích

        BufferedWriter writer = null;    // Khai báo BufferedWriter

        // Mở (hoặc tạo) file đích bằng FileWriter và BufferedWriter
        writer = new BufferedWriter(new FileWriter(destFile));

        // Chuỗi dữ liệu cần ghi
        String content = "Hello, World!\nWelcome to Java File Handling.";

        // Ghi chuỗi vào file
        writer.write(content);

        System.out.println("Đã ghi file thành công.");
}
```

BufferedReader

```
public static void main(String[] args) {
        String sourceFile = "source.txt";          // Đường dẫn đến file nguồn

        BufferedReader reader = null;              // Khai báo BufferedReader

        // Mở file nguồn bằng FileReader và BufferedReader
        reader = new BufferedReader(new FileReader(sourceFile));

        // Biến để lưu trữ dòng đọc được
        String line;

        // Đọc từng dòng từ file và in ra màn hình
        while ((line = reader.readLine()) != null) {
                System.out.println(line);
        }
}
```

Thực hành 60’

Tạo 1 chương trình trong đó:

- Cho phép tạo 1 file B, tên do người dùng nhập
- Lấy danh sách học viên từ file A có sẵn để ghi vào file B
- Trong quá trình ghi file B thì file A cũng bị xoá

Lưu ý:

- *Cần thực hiện việc xoá file A trong khi ghi dữ liệu vào file B (**KHÔNG** thực hiện xoá sau khi đã hoàn tất quá trình ghi file B)*
- *Giữa các thuộc tính trong file A là 1 tab*

Nâng cao (30’):

- Giả sử ghi được 1 phần file B thì bị ngắt quãng (dừng ghi file mà ko phải do tắt chương trình), hãy tiếp tục ghi file B mà không phải là ghi lại từ đầu

ROAD TO KOREA

Nếu có bất kỳ thắc mắc nào, hãy đặt câu hỏi qua

mail@mail.com hoặc Zalo 0xxx xxx xxx