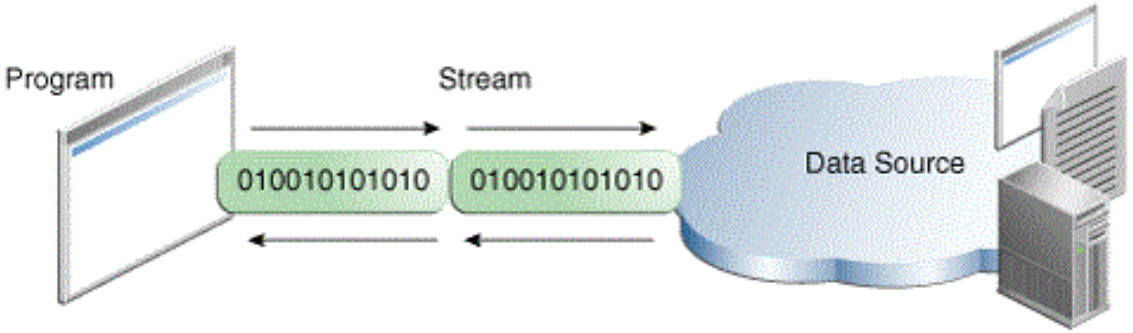
DEVMASTER.VN

Bài 13
File

Mục tiêu

- Giới thiệu luồng dữ liệu (data stream)
- Lớp **File**
- **DataInput** và **DataOutput**
- Mô tả byte stream và character stream trong java.io.package
- Lớp **ObjectInputStream** và **ObjectOutputStream**



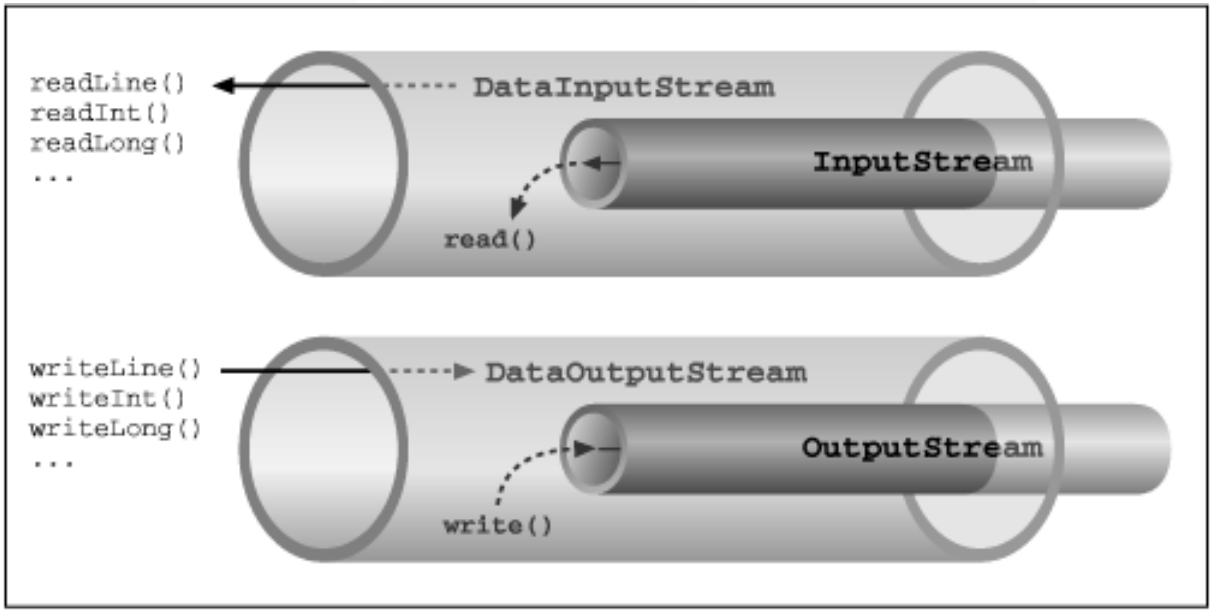
Data Stream

Luồng (**stream**) được mô tả như sau:

- Java làm việc với luồng dữ liệu gọi là **stream**.
- **Stream** là một dãy dữ liệu hoặc thực thể logic xuất hoặc nhập thông tin.
- **Stream** dữ liệu là một kênh thông tin qua đó dữ liệu được truyền từ **nguồn** tới **đích**.
- Nguồn hoặc đích có thể là: **thiết bị** đầu vào-ra, thiết bị **lưu trữ**, **mạng** máy tính…

Data Stream

- Để đọc file vật lý có thể sử dụng **FileInputStream** hoặc **FileReader**.
- Java xây dựng **nhiều dạng stream** khác nhau để thực hiện các hoạt động **nhập/xuất** khác nhau



Data Stream

- Lớp **System** trong Java có 3 trường biểu diễn stream vào/ra tiêu chuẩn như sau:
  - **in**: dòng đầu vào tiêu chuẩn dùng để đọc các ký tự dữ liệu
  - **out**: dòng đầu ra tiêu chuẩn để hiển thị thông tin ra trên màn hình hoặc bất kỳ phương tiện khác.
  - **err**: luồng lỗi tiêu chuẩn.

Data Stream

Lớp Stream có tác dụng:

1. Đọc dữ liệu vào từ stream.
2. Xuất dữ liệu ra từ stream.
3. Quản lý file trên ổ cứng.
4. Chia sẻ dữ liệu trên mạng giữa các máy tính.

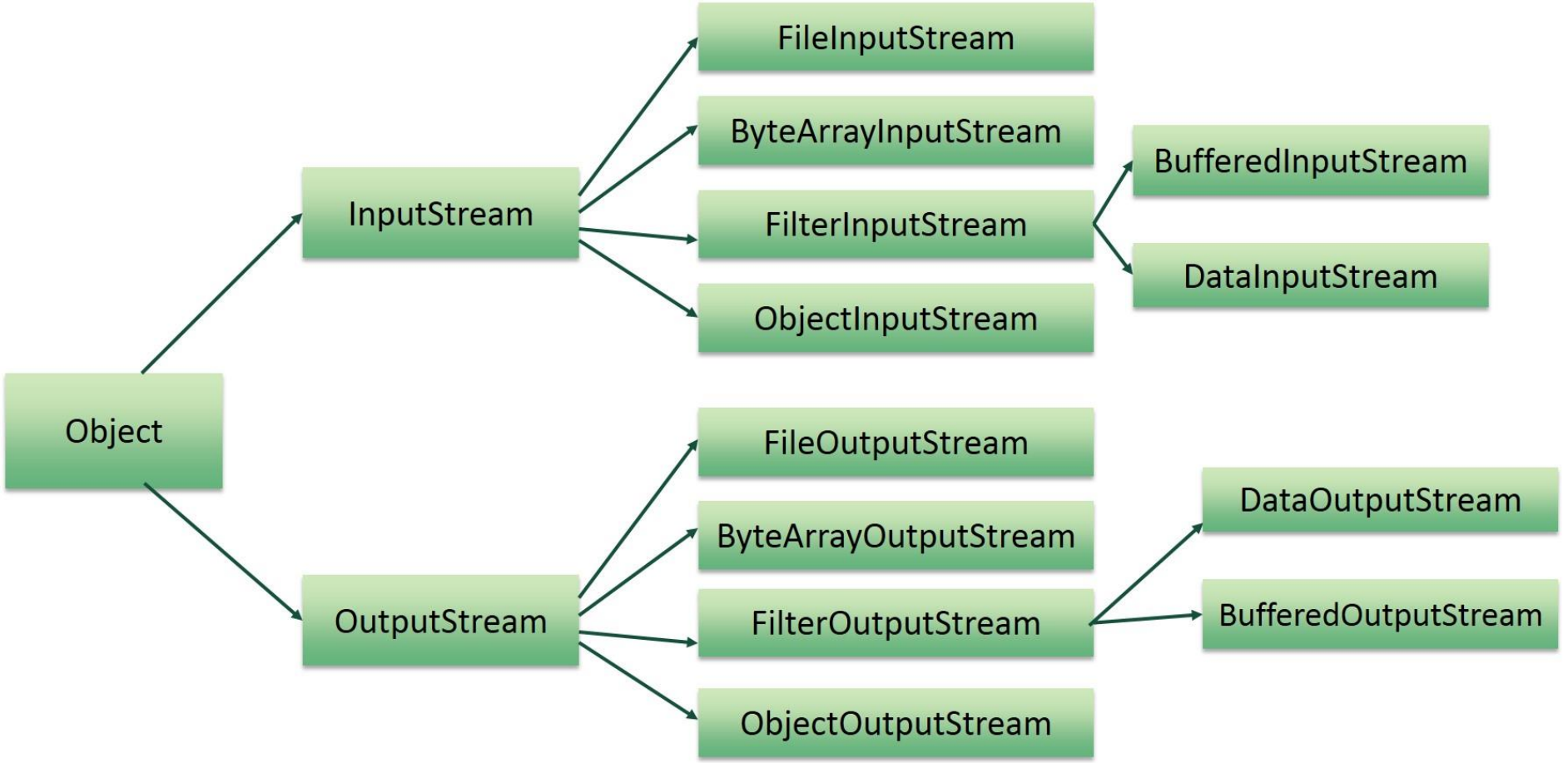
Data Stream

Các bước để bắt đầu sử dụng stream nhập/xuất dữ liệu:

1. Mở stream cùng với mô tả dữ liệu nguồn như: file, socket, URL, và ….
2. Nhập/xuất dữ liệu với stream.
3. Đóng stream.

- **Input** và **Output stream** là lớp trừu tượng được sử dụng để đọc/viết dãy không có cấu trúc của byte.
- Các **input/output stream** khác là lớp con của Input/Output stream được sử dụng để **đọc/ghi file.**

Data Stream


Object
InputStream
FileInputStream
ByteArrayInputStream
FilterInputStream
ObjectInputStream
BufferedInputStream
DataInputStream
OutputStream
FileOutputStream
ByteArrayOutputStream
FilterOutputStream
ObjectOutputStream
DataOutputStream
BufferedOutputStream

Lớp File

- Lớp File trực tiếp làm việc với các **file** và **hệ thống** tập tin.
- Các tập tin được đặt tên theo quy ước của hệ điều hành.
- Path (đường dẫn) có thể là tương đối hoặc tuyệt đối.
- Các lớp trong gói **java.io** giải quyết các đường dẫn tương đối so với thư mục người dùng hiện tại.

Lớp File

- Các phương thức của lớp File cho phép tạo, xóa, đổi tên, liệt kê các thư mục.
- Các interface, lớp định nghĩa trong gói **java.nio** giúp máy ảo Java truy cập hệ thống tệp tin và thuộc tính.
- Các Contructor của File:
  - File(**String** dirpath)
  - File(**String** parent, **String** child)
  - File(**File** fileobj, **String** filename)
  - File(**URL** urlobj)

Lớp File

Một số phương thức của lớp File:

- **renameTo(File newname)**: đổi tên file có sẵn bằng tên mới với tham số tên.
- **delete():** xóa file.
- **exists():** kiểm tra file hoặc thư mục có tồn tai không.
- **getPath():** lấy về đường dẫn tới file/thư mục.

Lớp File

- **isFile():** kiểm tra xem có phải là file không.
- **createNewFile():** tạo file mới (với đường dẫn chỉ định) nếu không có file nào tên tương tự.
- **mkdir():** tạo thư mục mới.
- **toPath():** trả về đối tượng java.nio.file.Path.
- **toURI():** xây dựng tệp, URI. Tập tin này đại diện cho tên đường dẫn trừu tượng.

Lớp File

Mã nguồn demo phương thức trong lớp File:

Code Snippet

```
. . .
File fileObj = new File("C:/Java/Hello.txt");
System.out.println("Path is: " +fileObj.getPath());
System.out.println("Name is: " +fileObj.getName());
System.out.println("File exists is: " +fileObj.exists());
System.out.println("File is: " +fileObj.isFile());
...
```

Lớp File

DEVMASTER.VN

Lọc File theo phần mở rộng:

Code Snippet

```
import java.io.*;
class FileFilter implements FilenameFilter {
    String ext;
    public FileFilter(String ext) {
    this.ext = "." + ext;
    }
    public boolean accept (File dir, String fName) {
    return fName.endsWith(ext);
    }
}
public class DirList {
    public static void main (String [] args) {
    String dirName = "d:/resources";
```

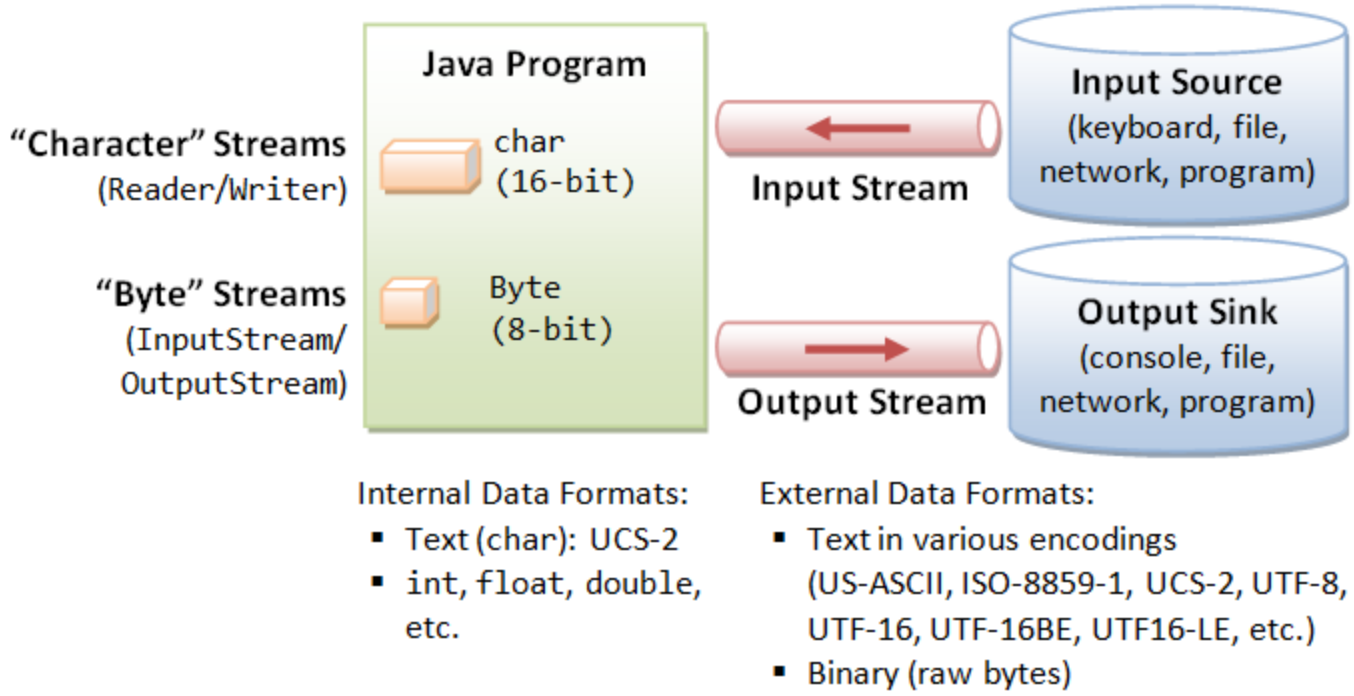
Lớp File

Lọc File theo phần mở rộng:

```
    File fileObj = new File (“d:/resources”);
    FilenameFilter filterObj = new FileFilter(“java“);
    String[] fileName = fileObj.list(filterObj);
    System.out.println(“Number of files found : “ +
fileName.length);
    System.out.println(“” );
    System.out.println(“Names of the files are : “ );
    System.out.println(“------------------------ “ );
    for(int ctr=0; ctr < fileName.length; ctr++) {
    System.out.println(fileName[ctr]);
    }
    }
}
```

DataInput và DataOutput

Stream dữ liệu hỗ trợ nhập/xuất kiểu dữ liệu cơ bản và chuỗi. Stream dữ liệu thực thi interface **DataInput** hoặc **DataOutput**.



DataInput và DataOutput

- Interface **DataInput** có các phương thức:
  - Đọc các **byte dữ liệu** từ stream nhị phân và chuyển đổi dữ liệu về kiểu **dữ liệu Java cơ bản**.
  - Chuyển đổi dữ liệu từ định dạng **Unicode** (utf-8) của Java sang dạng **chuỗi**.
- Interface DataOutput có các phương thức:
  - Chuyển đổi **dữ liệu Java cơ bản** thành dãy các **byte** và ghi chúng vào luồng nhị phân.
  - Chuyển dổi dữ liệu kiểu chuỗi trong Java sang dạng **UTF-8** và ghi chúng vào stream.

DataInput và DataOutput

Các phương thức của interface DataInput:

- readBoolean()
- readByte()
- readInt()
- readDouble()
- readChar()
- readLine()
- readUTF()

DataInput và DataOutput

Code sử dụng interface DataInput:

Code Snippet

```
try
{
DataInputStream dis = new DataInputStream(System.in);
double d = dis.readDouble();
int num = dis.readInt();
}
catch(IOException e) {}
. . .
```

DataInput và DataOutput

Các phương thức của interface DataOutput:

- writeBoolean(boolean b)
- writeByte(int value)
- writeInt(int value)
- writeDouble(double value)
- writeChar(int value)
- writeChars(String value)
- writeUTF(String value)

DataInput và DataOutput

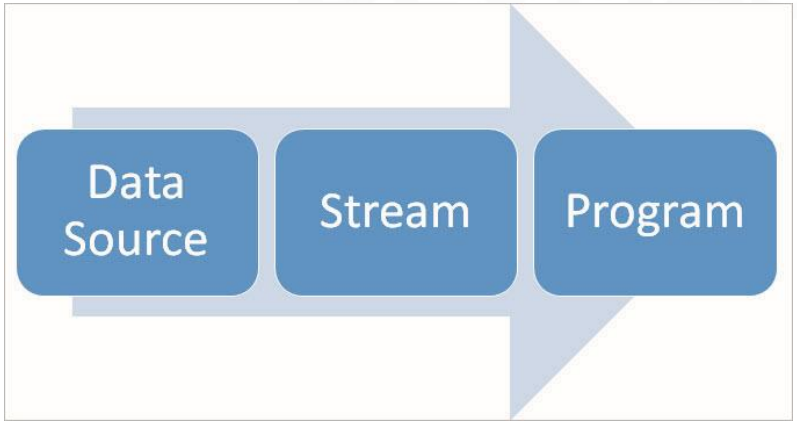
Code sử dụng interface OutInput:

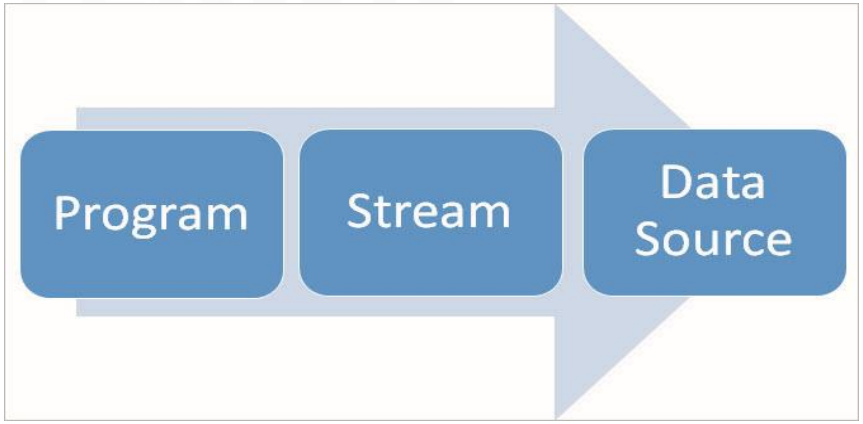
Code Snippet

```
try
{
outStream.writeBoolean(true);
outStream.writeDouble(9.95);
. . .
}
catch (IOException e) {}
. . .
```

Byte Stream và Character Stream

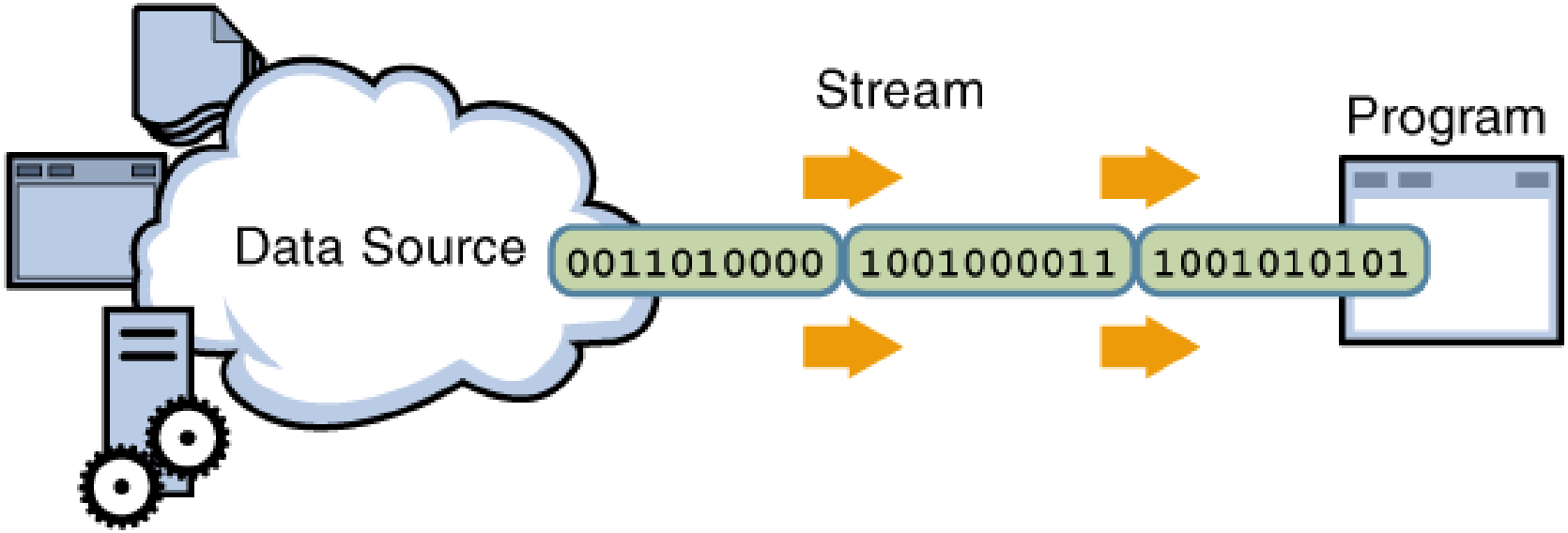
- **Stream** là đại diện cho nhiều nguồn và đích đến, chẳng hạn như file, bộ nhớ.
- Đó cũng có thể là dãy dữ liệu.
- Các stream hỗ trợ nhiều dạng dữ liệu như byte, dữ liệu cơ bản, ký tự….
- Luồng nhất định cho phép dữ liệu qua và chuyển đổi dữ liệu một cách hữu ích.





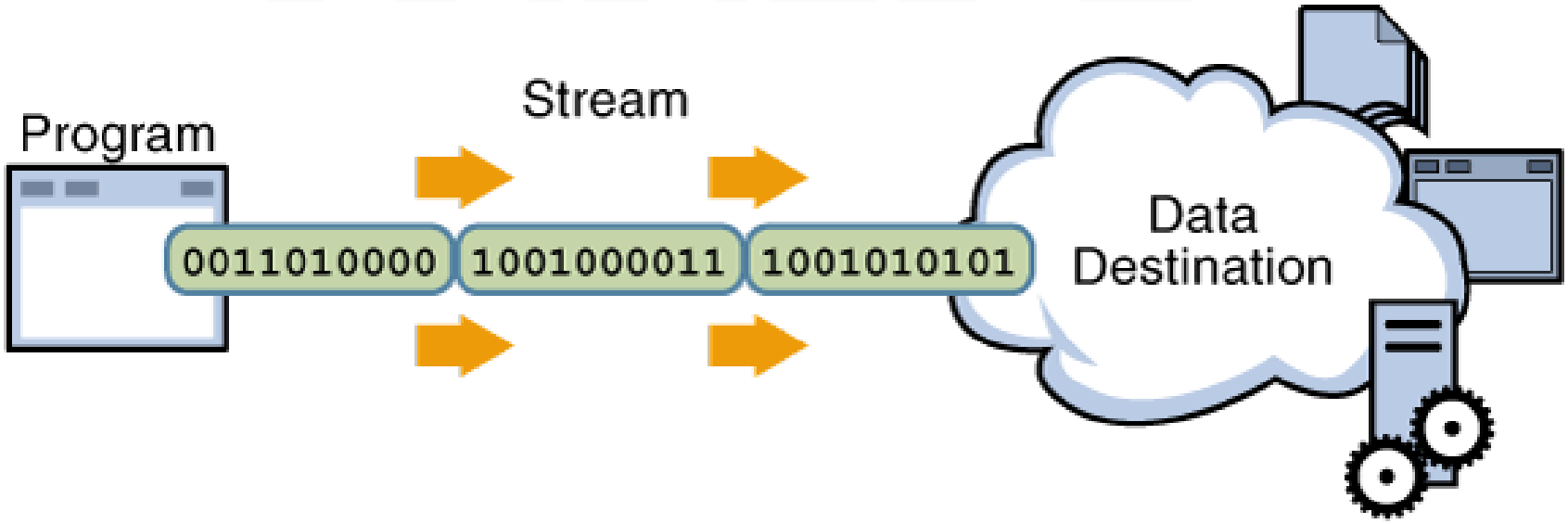
Byte Stream và Character Stream

Dữ liệu qua stream đọc vào chương trình

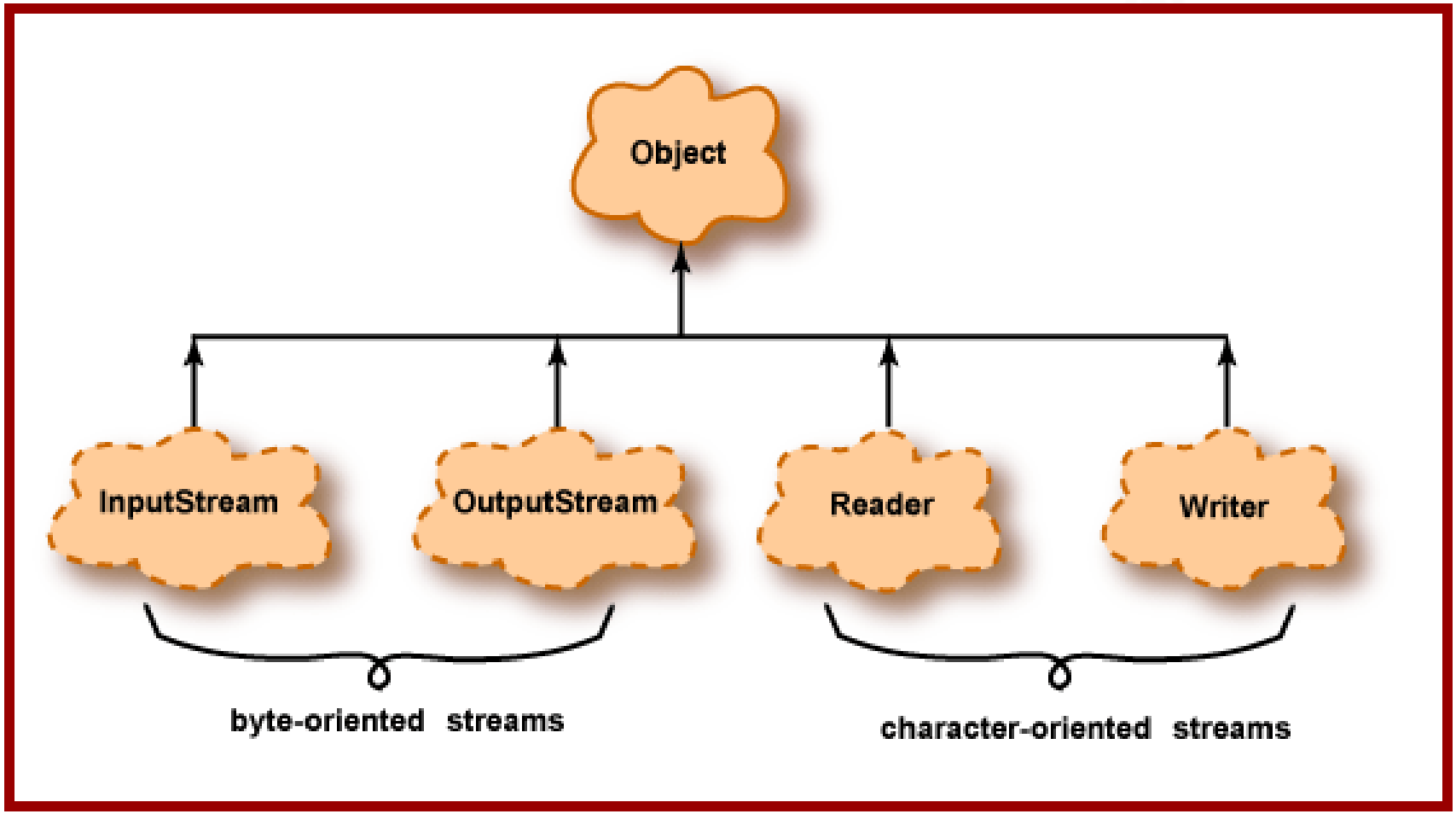


Byte Stream và Character Stream

Dữ liệu qua stream từ chương trình ghi tới đích (file, bộ nhớ…)



Byte Stream và Character Stream



Byte Stream và Character Stream

Đoạn mã sau sử dụng **FileInputStream** và **FileOutputStream** để làm việc với **byte stream**.

Code Snippet

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
public class ByteStreamApp {
public static void main(String[] args) throws IOException {
   FileInputStream inObj = null;
   FileOutputStream outObj = null;
   try {
    inObj = new FileInputStream("c:/java/hello.txt");
    outObj = new FileOutputStream("outagain.txt");
    int ch;
    while ((ch = inObj.read()) != -1) {
     outObj.write(ch);
```

Byte Stream và Character Stream

DEVMASTER.VN

Code Snippet

```
    }
   } finally {
     if (inObj != null) {
      inObj.close();
      }
      if (outObj != null) {
       outObj.close();
      }
     }
    }
  }
```

Byte Stream và Character Stream

Đoạn mã sau sử dụng **FileReader** và **FileWriter** để làm việc với **character stream**.

Code Snippet

```
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
public class CharStreamApp {
 public static void main(String[] args) throws IOException {
  FileReader inObjStream = null;
  FileWriter outObjStream = null;
```

Byte Stream và Character Stream

DEVMASTER.VN

Code Snippet

```
try {
 inObjStream = new FileReader("c:/java/hello.txt");
 outObjStream = new FileWriter("charoutputagain.txt");
 int ch;
 while ((ch = inObjStream.read()) != -1) {
  outObjStream.write(ch);
 }
} finally {
  if (inObjStream != null) {
   inObjStream.close();
  }
 }
}
```

ObjectInputStream và ObjectOutputStream

ObjectInputStream là lớp mở rộng từ InputStream và thực thi interface ObjectInput. Lớp hỗ trợ đọc dữ liệu cơ bản và đối tượng vào stream trong.

Code Snippet

```
. . .
FileInputStream fObj = new FileInputStream("point");
ObjectInputStream ois = new ObjectInputStream(fObj);
Point obj = (Point) ois.readObject();
ois.close();
```

ObjectInputStream và ObjectOutputStream

ObjectOutputStream là lớp mở rộng từ OutputStream và thực thi interface ObjectOutput. Lớp hỗ trợ viết dữ liệu cơ bản và đối tượng ra stream ngoài.

Code Snippet

```
. . .
Point pointObj = new Point(50,75);
FileOutputStream fObj = new FileOutputStream("point");
ObjectOutputStream oos = new ObjectOutputStream(fObj);
oos.writeObject(pointObj);
oos.writeObject(new Date());
oos.close();
. . .
```

ObjectInputStream và ObjectOutputStream

Demo đọc/ghi dữ liệu đối tượng:

Code Snippet

```
import java.io.Serializable;
public class Employee implements Serializable{
    String lastName;
    String firstName;
    double sal;
}
public class BranchEmpProcessor {
public static void main(String[] args) {
        FileInputStream fIn = null;
        FileOutputStream fOut = null;
ObjectInputStream oIn = null;
```

ObjectInputStream và ObjectOutputStream

```
        ObjectOutputStream oOut = null;
              try {
            fOut = new FileOutputStream("E:\\NewEmplyee.Ser");
            oOut = new ObjectOutputStream(fOut);
            Employee e = new Employee();
            e.lastName = "Smith";
            e.firstName = "John";
            e.sal = 5000.00;
            oOut.writeObject(e);
            oOut.close();
            fOut.close();
            fIn = new FileInputStream("E:\\NewEmplyee.Ser");
            oIn = new ObjectInputStream(fIn);
            //de-serializing employee
            Employee emp = (Employee) oIn.readObject();
          System.out.println("Deserialized  -  "  +
emp.firstName  +  "  "  +
          emp.lastName + " from NewEmployee.ser");
```

ObjectInputStream và ObjectOutputStream

```
        } catch (IOException e) {
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } finally {
             System.out.println("finally");
        }
    }
}
```

Tóm tắt bài học

- ✓ **Stream** là thực thể logic để xuất hoặc nhập thông tin.
- ✓ **Data stream** hỗ trợ nhập/xuất dữ liệu cơ bản và chuỗi.
- ✓ **InputStream** là lớp trừu tượng định nghĩa cách dữ liệu được nhận.
- ✓ **OutputStream** xác định cách mà đầu ra được ghi vào luồng.
- ✓ Lớp **File** trực tiếp làm việc với các tệp trên hệ thống.
- ✓ **Serialization** cho phép xử lý quá trình đọc/ghi đối tượng vào luồng byte.

Thank for watching!