UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

NGÔN NGỮ LẬP TRÌNH JAVA

Chương 4
NHẬP XUẤT VÀ QUẢN LÝ NGOẠI LỆ (P1)
NHẬP XUẤT TRONG JAVA

GVGD: ThS. Lê Thanh Trọng

NỘI DUNG

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

- Mục tiêu bài học
- Nhập xuất dữ liệu và luồng dữ liệu
- Một số lớp luồng hỗ trợ nhập xuất dạng nhị phân trong Java
- Một số lớp hỗ trợ việc đọc ghi file dạng text
- Sử dụng cơ chế Serialization trong Java để ghi/ đọc cái đối tượng
- Tóm tắt bài học

Mục tiêu bài học

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

- Giới thiệu cho sinh viên về cơ chế nhập xuất dữ liệu trong Java.
- Khai niệm về luồng dữ liệu và các lớp hỗ trợ việc nhập xuất như **InputStream, OutputStream, FileInputStream, FileOutputStream.**
- Các lớp luồng cung cấp cơ chế bộ đệm để việc đọc ghi dữ liệu hiệu quả hơn : **BufferedInputStream, BufferedOutputStream…**

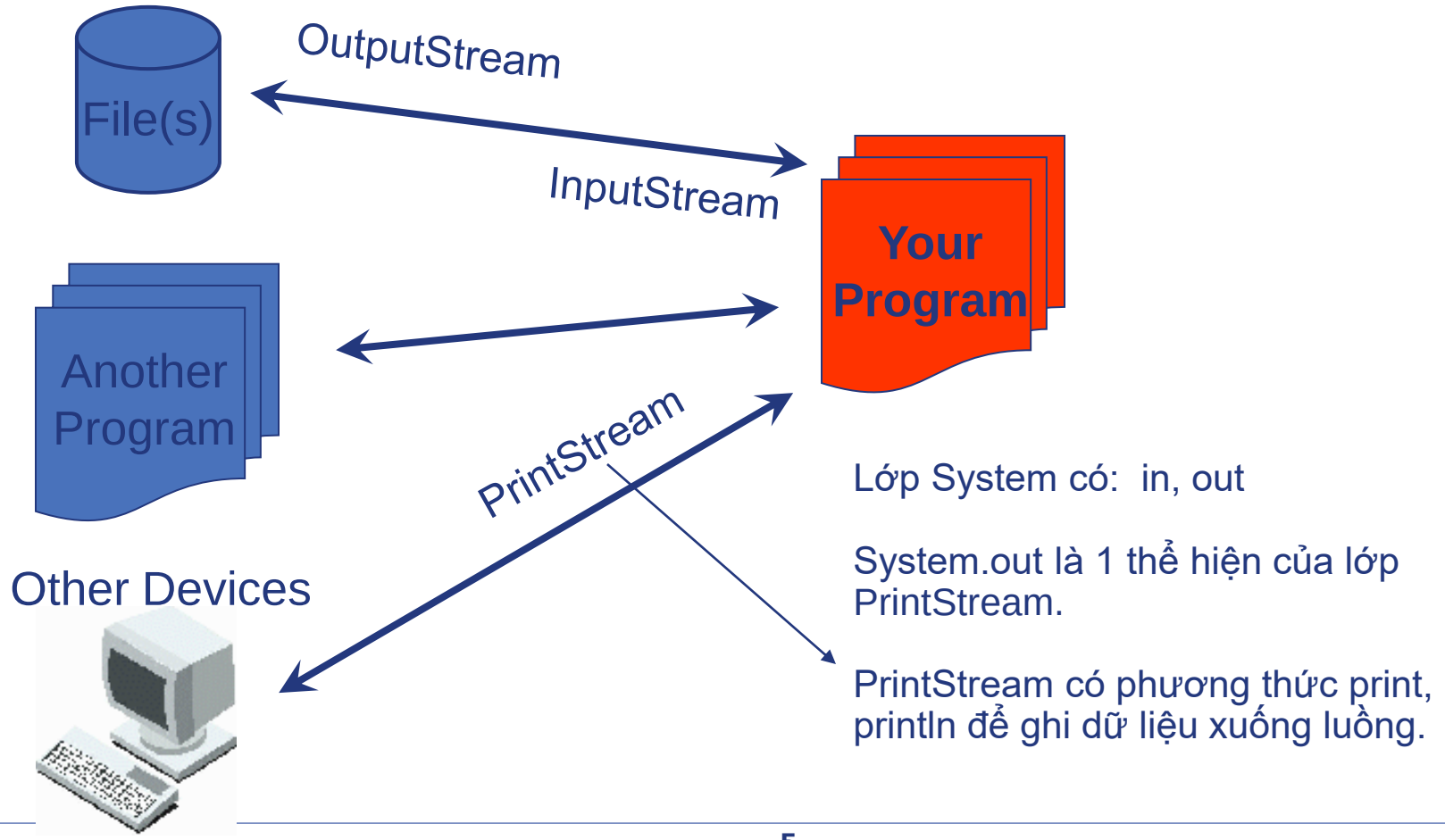
Mục tiêu bài học

- Các lớp hỗ trợ việc đọc/ ghi các tập tin text như Writer/ Reader và các lớp con của chúng.
- Sinh viên có thể nắm được định nghĩa và cách sử dụng các lớp này trong quá trình lập trình.

Nhập/xuất dữ liệu

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Nhập xuất dữ liệu trong Java dựa trên mô hình luồng dữ liệu



Lớp System có: in, out

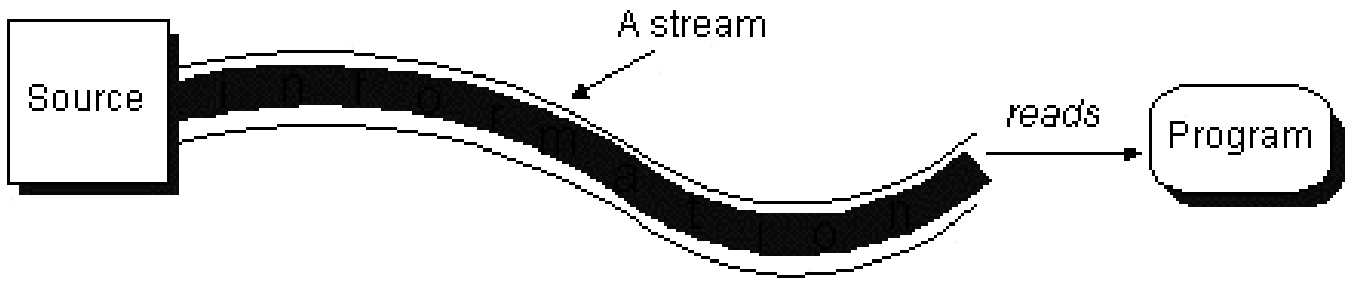
System.out là 1 thể hiện của lớp PrintStream.

PrintStream có phương thức print, println để ghi dữ liệu xuống luồng.

Nhập xuất dữ liệu

- Dữ liệu có thể đến từ bất kỳ nguồn nào và xuất ra bất kỳ nơi nào
  - Memory
  - Disk
  - Network
- Bất kể nguồn/đích loại nào đều có thể sử dụng luồng để đọc/ghi dữ liệu.

Nhập xuất dữ liệu



Đọc dữ liệu

Open a Stream

While more Information

Read

Close the Stream

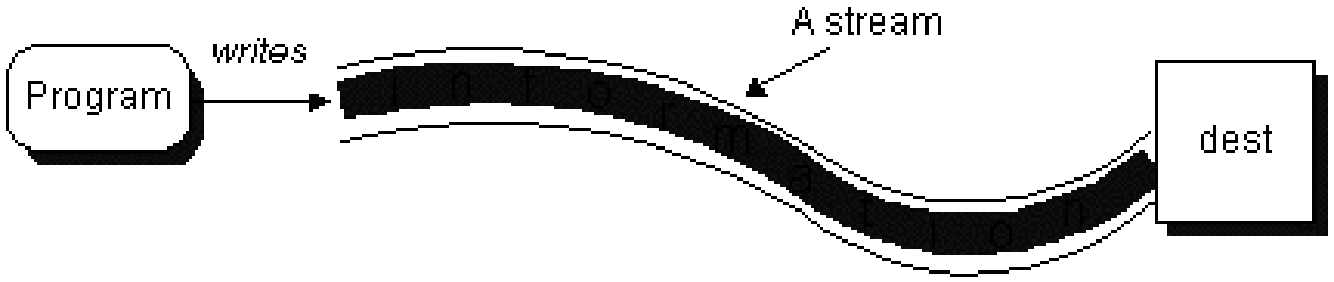
Ghi dữ liệu

Open a Stream

While more Information

Write

Close the Stream

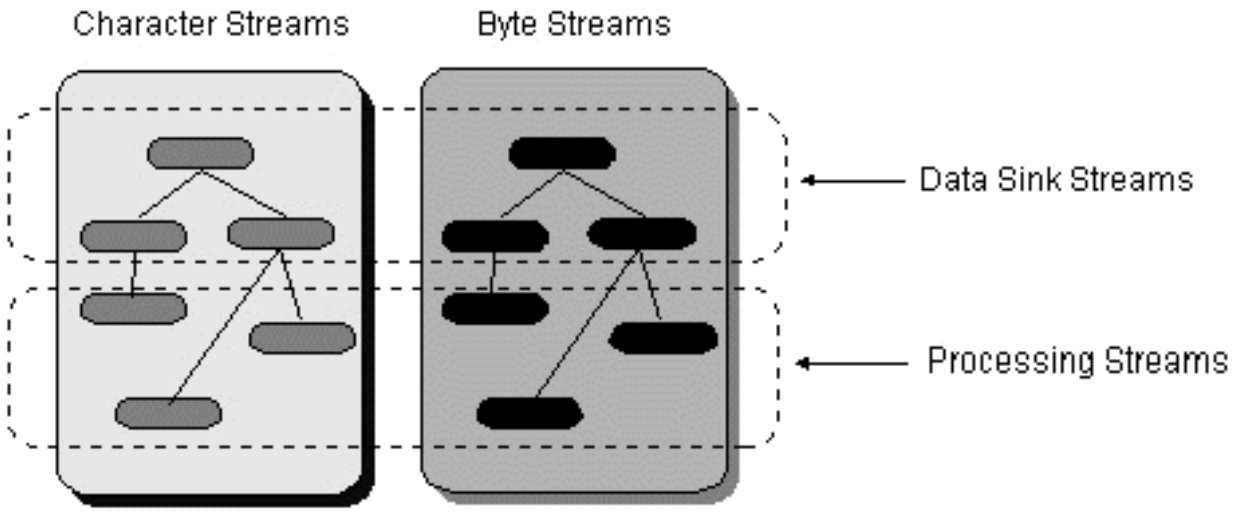


Luồng dữ liệu

- Các luồng là những đường ống dẫn để gửi và nhận thông tin trong các chương trình java.
- Khi một luồng đọc hoặc ghi, các luồng khác bị khoá.
- Nếu lỗi xảy ra trong khi đọc hoặc ghi luồng, một ngoại lệ sẽ kích hoạt.

Luồng dữ liệu

- Gói java.io cung cấp các lớp cài đặt luồng dữ liệu.
- Phân loại luồng



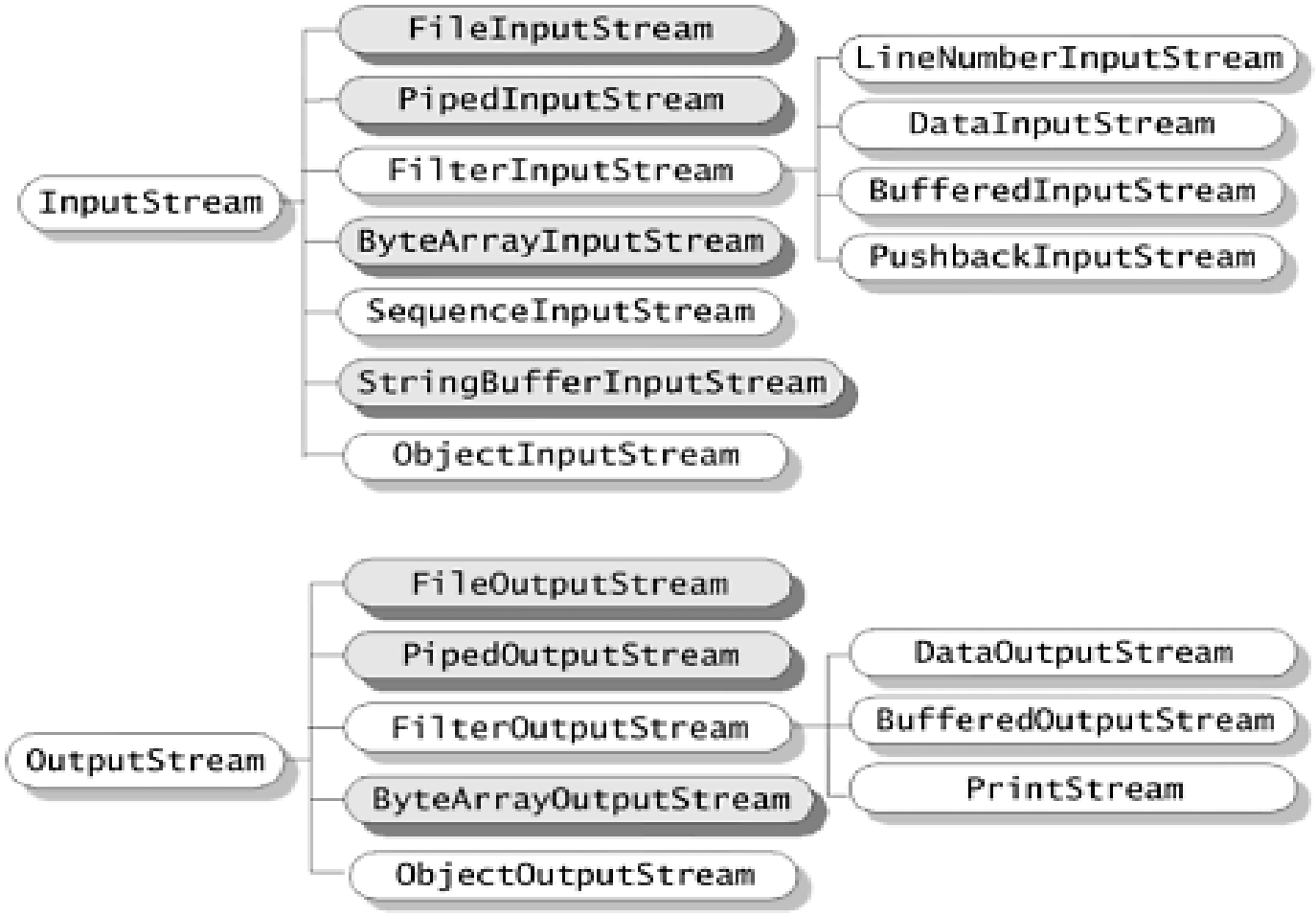
Luồng dữ liệu

- *Luồng Character* được dùng khi thao tác trên ký tự (16 bits) – Sử dụng lớp *Reader* & *Writer*
- *Byte Streams* được dùng khi thao tác dữ liệu nhị phân (8 bits) – Sử dụng lớp *InputStream* & *OutputStream*
- Data Sinks
  - Files
  - Memory
  - Pipes
- Processing
  - Buffering
  - Filtering

Các luồng nhập xuất chuẩn

- Lớp System.in
  - Cung cấp luồng cho việc đọc chuẩn – thường dùng trong việc nhập kí tự từ bàn phím
- Lớp System.out
- Lớp System.err
  - 2 lớp này cung cấp luồng xuất dữ liệu và xuất lỗi chuẩn – thường để hiển thị thông tin ra màn hình cho người dùng

Luồng Byte


InputStream
FileInputStream
PipedInputStream
FilterInputStream
ByteArrayInputStream
SequenceInputStream
StringBufferInputStream
ObjectInputStream
LineNumberInputStream
DataInputStream
BufferedInputStream
PushbackInputStream
OutputStream
FileOutputStream
PipedOutputStream
FilterOutputStream
ByteArrayOutputStream
ObjectOutputStream
DataOutputStream
BufferedOutputStream
PrintStream

Lớp InputStream

- Là lớp trừu tượng
- Định nghĩa cách nhận dữ liệu
- Cung cấp một số phương thức dùng để đọc từ các luồng dữ liệu đầu vào.
- Các phương thức:
  - **int read()**
  - **int read(byte[] buffer)**
  - **int read(byte[] buffer, int offset, int length)**
  - **int available( )**
  - **void close ( )**
  - **void reset( )**
  - **long skip( )**

Lớp InputStream

- int read()

```
byte[] b = new byte[10];
for (int i = 0; i < b.length; i++) {
   b[i] = (byte) System.in.read();
}
```

Lớp InputStream

- int read(byte[] buffer, int offset, int length)

```
try {
    byte[] b = new byte[100];
    int offset = 0;
    while (offset < b.length) {
        int bytesRead = System.in.read(b, offset, b.length - offset);
        if (bytesRead == -1) break; // end of stream
        offset += bytesRead;
    }
}
catch (IOException ex) {
    System.err.println("Couldn't read from System.in!");
}
```

Lớp InputStream

- int available()

```
try {
   byte[] b = new byte[System.in.available( )];
   System.in.read(b);
}
 catch (IOException ex) {
   System.err.println("Couldn't read from System.in!");
}
```

Lớp InputStream

- long skip()

```
try {
   long bytesSkipped = 0;
   long bytesToSkip = 80;
   while (bytesSkipped < bytesToSkip) {
        long n = in.skip(bytesToSkip - bytesSkipped);
        if (n == -1) break;
        bytesSkipped += n;
   }
 }
catch (IOException ex) { System.err.println(ex); }
```

Lớp OutputStream

- Là lớp trừu tượng.
- Định nghĩa cách ghi dữ liệu vào luồng.
- Cung cấp tập các phương thức trợ giúp. trong việc tạo, ghi và xử lý các luồng xuất.
- Các phương thức:
  - **void write(int)**
  - **void write(byte[ ])**
  - **write(byte[ ], int, int)**
  - **void flush( )**
  - **void close( )**

Lớp OutputStream

- void write(int i)

```
public class AsciiChart {
   public static void main(String[] args) {
        for (int i = 32; i < 127; i++) {
            System.out.write(i);
            // break line after every eight characters.

            if (i % 8 == 7) System.out.write('\n');
            else System.out.write('\t');
        }
        System.out.write('\n');
   }
}
```

Lớp OutputStream

- void write(byte[] buff)
- void write(byte[] buff, int offset, int length)

```
public class WriteBytes {
    public static void main(String[] args){
        try{
            String message = "Hello World";
            byte[] data = message.getBytes();
            System.out.write(data);
        }
        catch(IOException e)
        {
            System.out.println("IO errors");
        }
    }
}
```

Lớp OutputStream

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

```
public class StreamCopier {
    public static void main(String[] args) {
            try {
                    copy(System.in, System.out); }
            catch (IOException ex) {
                    System.err.println(ex);
            }
     }
    public static void copy(InputStream in, OutputStream out) throws
                                            IOException {
            byte[] buffer = new byte[1024];
            while (true) {
                    int bytesRead = in.read(buffer);
                    if (bytesRead == -1) break;
                    out.write(buffer, 0, bytesRead);
            }
    }
}
```

Lớp FileOutputStream

- Cho phép kết xuất để ghi ra một luồng tập tin
- Để tạo đối tượng FileOutputStream ta cần thông số một chuỗi tên tập tin, tập tin, hay đối tượng FileDescriptor như một tham số.
- Lớp này nạp chồng các phương thức của lớp OutputStream và cung cấp phương thức 'getFD( )'

Sử Dụng FileOutputStream

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

```
byte[] originalData = new byte[10];
for (int i=0; i<originalData.length; i++)
  {
  originalData[i]=(byte)(Math.random()*128.0);
  }
FileOutputStream fw=null;
try { fw = new FileOutputStream("io1.dat"); }
catch (IOException fe )
  { System.out.println(fe); }
for (int i=0; i<originalData.length; i++)
  {
  try { fw.write(originalData[i]); }
  catch (IOException ioe)
  {System.out.println(ioe); }
  }
try { fw.close(); }
catch (IOException ioe)
  {System.out.println(ioe); }
```

Lớp FileInputStream

- Cho phép đầu vào đọc từ một tập tin
- Chuỗi tên tập tin, tập tin, đối tượng FileDescriptor như một tham số được dùng để khởi tạo ra đối tượng FileInputStream
- Các phương thức nạp chồng của lớp InputStream. nó cung cấp phương thức ‘getFD( )’

Sử Dụng FileInputStream

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

```
byte data=0;
int bytesInFile=0;
FileInputStream fr = null;
try
  {
  fr = new FileInputStream("io1.dat");
  bytesInFile = fr.available();
  for (int i=0; i<bytesInFile; i++)
      {
      data=(byte)fr.read();
      System.out.println("Read "+data);
      }
  fr.close();
  }
catch (IOException ioe)
  {
  System.out.println("Problem with file");
  }
```

ByteArrayInputStream

- ByteArrayInputStream có thể được sử dụng để đọc mảng byte như là input stream.
- Lớp ByteArrayInputStream trong java chứa một bộ đệm bên trong được sử dụng để đọc mảng byte dưới dạng luồng.
- Bộ đệm của ByteArrayInputStream tự động tăng theo kích thước dữ liệu.

ByteArrayInput

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

```
public class ByteArrayInputStreamExample {
    public static void main(String[] args) throws IOException {
        ByteArrayInputStream bis = null;

        try {
            byte[] buf = { 35, 36, 37, 38 };
            // Create the new byte array input stream
            bis = new ByteArrayInputStream(buf);
            int k = 0;
            while ((k = bis.read()) != -1) {
                // Conversion of a byte into character
                char ch = (char) k;
                System.out.println("ASCII value of Character is:" + k + "; "
                        + "Special character is: " + ch);
            }
        } finally {
            bis.close();
        }
    }
}
```

ByteArrayOutput

- **Lớp ByteArrayOutputStream** tạo một buffer trong bộ nhớ và tất cả các data được gửi đi sẽ lưu trữ ở đây.
- ByteArrayOutputStream giữ một bản sao của dữ liệu và chuyển tiếp nó đến nhiều stream.
- Bộ đệm của ByteArrayOutputStream tự động tăng theo kích thước dữ liệu.

ByteArrayOutputStream

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

```
public class ByteArrayOutputStreamExample {
    public static void main(String args[]) throws Exception {
        FileOutputStream fout1 = null;
        FileOutputStream fout2 = null;
        ByteArrayOutputStream bout = null;

        try {
            fout1 = new FileOutputStream("D:\\f1.txt");
            fout2 = new FileOutputStream("D:\\f2.txt");
            bout = new ByteArrayOutputStream();
            String data = "Hello";
            bout.write(data.getBytes());
            bout.writeTo(fout1);
            bout.writeTo(fout2);
            bout.flush();
            System.out.println("Success...");
        } catch (IOException ex) {
           ……..
           }
```

Bộ lọc

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

- Lọc:
  - Là kiểu luồng sửa đổi cách điều quản một luồng hiện có.
  - Về cơ bản được sử dụng để thích ứng các luồng theo các nhu cầu của chương trình cụ thể.
  - Bộ lọc nằm giữa luồng cơ sở và chương trình.
  - Thực hiện một số tiến trình đặt biệt trên các byte được chuyển giao từ đầu vào đến kết xuất.
  - Có thể phối hợp để thực hiện một dãy các tuỳ chọn lọc.

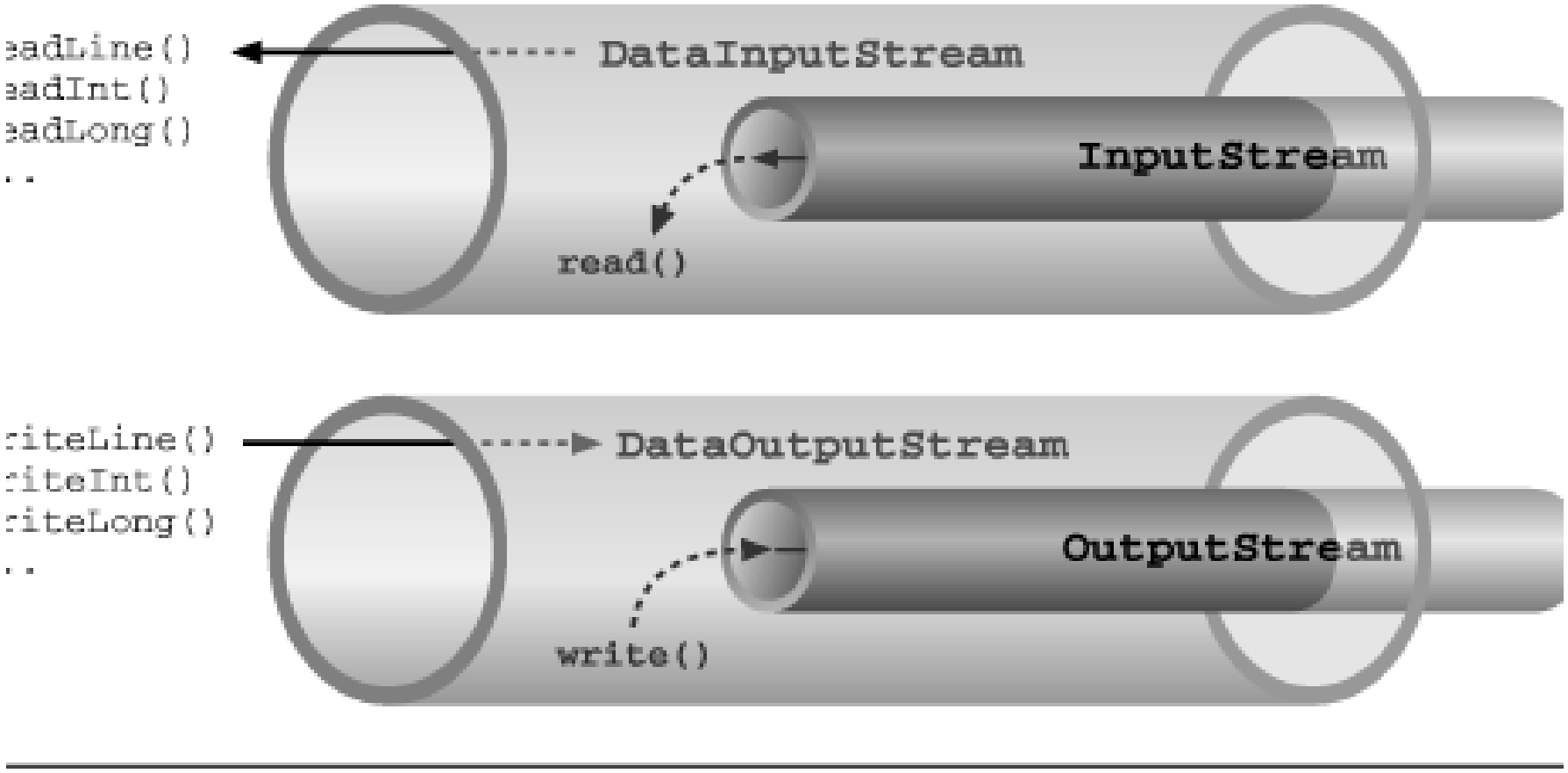
Lớp FilterInputStream

- Là lớp trừu tượng.
- Là cha của tất cả các lớp luồng nhập đã lọc.
- Cung cấp khả năng tạo ra một luồng từ luồng khác.
- Một luồng có thể đọc và cung cấp cung cấp dưới dạng kết xuất cho luồng khác.
- Duy trì một dãy các đối tượng của lớp 'InputStream'
- Cho phép tạo ra nhiều bộ lọc kết xích (chained filters).

Lớp FilterOutputStream

- Là dạng bổ trợ cho lớp 'FilterInputStream'.
- Là cha của tất cả các lớp luồng kết xuất.
- Duy trì đối tượng của lớp 'OutputStream' như là một biến 'out'.
- Dữ liệu ghi ra lớp này có thể sửa đổi để thực hiện các thao tác lọc, và sau đó phản hồi đến đối tượng 'OutputStream'.

Luồng Lọc


eadLine()
eadInt()
eadLong()
..
DataInputStream
InputStream
read()
riteLine()
riteInt()
riteLong()
..
DataOutputStream
OutputStream
write()

Vùng đệm nhập/xuất

- Vùng đệm:
  - Là kho lưu trữ dữ liệu.
  - Có thể cung cấp dữ liệu thay vì quay trợ lại nguồn dữ liệu gốc ban đầu.
  - Java sử dụng vùng đệm nhập và kết xuất để tạm thời lập cache dữ liệu được đọc hoặc ghi vào một luồng.
- Trong khi thực hiện vùng đệm nhập:
  - Số lượng byte lớn được đọc cùng thời điểm, và lưu trữ trong một vùng đệm nhập.
  - Khi chương trình đọc luồng nhập, các byte nhập được đọc vào vùng đệm nhập.

Vùng đệm nhập/xuất (tt...)

- Trong trường hợp vùng đệm kết xuất, một chương trình ghi ra một luồng.
- Dữ liệu kết xuất được lưu trữ trong một vùng đệm kết xuất.
- Dữ liệu được lưu trữ cho đến khi vùng đệm trở nên đầy, hay luồng kết xuất được xả trống.
- Kết thúc, vùng đệm kết xuất được chuyển gửi đến đích của luồng xuất.

Lớp BufferedInputStream

- Tự động tạo ra và duy trì vùng đệm để hổ trợ vùng đệm nhập.
- bởi lớp 'BufferedInputStream' là một bộ đệm, nó có thể áp đụng cho một số các đối tượng nhất định của lớp 'InputStream'.
- Cũng có thể phối hợp các tập tin đầu vào khác.
- Sử dụng vài biến để triển khai vùng đệm nhập.

Lớp BufferedInputStream

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

- Định nghĩa hai phương thức thiết lập:
  - Một cho phép chỉ định kích thước của vùng đệm nhập.
  - phương thức kia thì không.
- Cả hai phương thức thiết lập đều tiếp nhận một đối tượng của lớp 'InputStream' như một tham số.
- Nạp chồng các phương thức truy cập mà InputStream cung cấp, và không đưa vào bất kỳ phương thức mới nào.

Lớp BufferedOutputStream

- Thực hiện vùng đệm kết xuất theo cách tương ứng với lớp 'BufferedInputStream'.
- Định nghĩa hai phương thức thiết lập. Nó cho phép chúng ta ấn định kích thước của vùng đệm xuất trong một phương thức thiết lập, cũng giống như cung cấp kích thước vùng đệm mặc định.
- Nạp chồng tất cả phương thức của lớp 'OutputStream' và không đưa vào bất kỳ phương thức nào.

Lớp DataInputStream

- Được sử dụng để đọc các byte từ luồng nhị phân, và xây dựng lại dữ liệu trong một số kiểu dữ liệu nguyên thuỷ (int, float,…).
- Hỗ trợ một số phương thức để đọc được các kiểu dữ liệu nguyên thuỷ.
- Ngoài ra, còn có thể đọc được các chuỗi Java mã hóa dạng UTF-8.

Những phương thức của DataInputStream

- boolean readBoolean( )
- byte readByte( )
- char readChar( )
- short readShort( )
- long readLong( )
- float readFloat( )
- int readInt( )
- double readDouble( )
- String readUTF( )
- String readLine( )

Lớp DataOutputStream

- Được sử dụng để ghi dữ liệu có kiểu dữ liệu nguyên thuỷ (int,float) vào luồng nhị phân (OutputStream). Và có thể đọc ra bằng DataInputStream.

- Định nghĩa một số phương thức để ghi dữ liệu và tất cả phương thức kích hoạt IOException trong trường hợp lỗi.

Các phương thức của DataOutputStream

- void writeBoolean(boolean b)
- void writeByte(int value)
- void writeChar(int value)
- void writeShort(int value)
- void writeLong(long value)
- void writeFloat(float value)
- void writeInt(int value)
- void writeDouble(double value)
- void writeUTF(String value)

Sử Dụng DataOutputStream

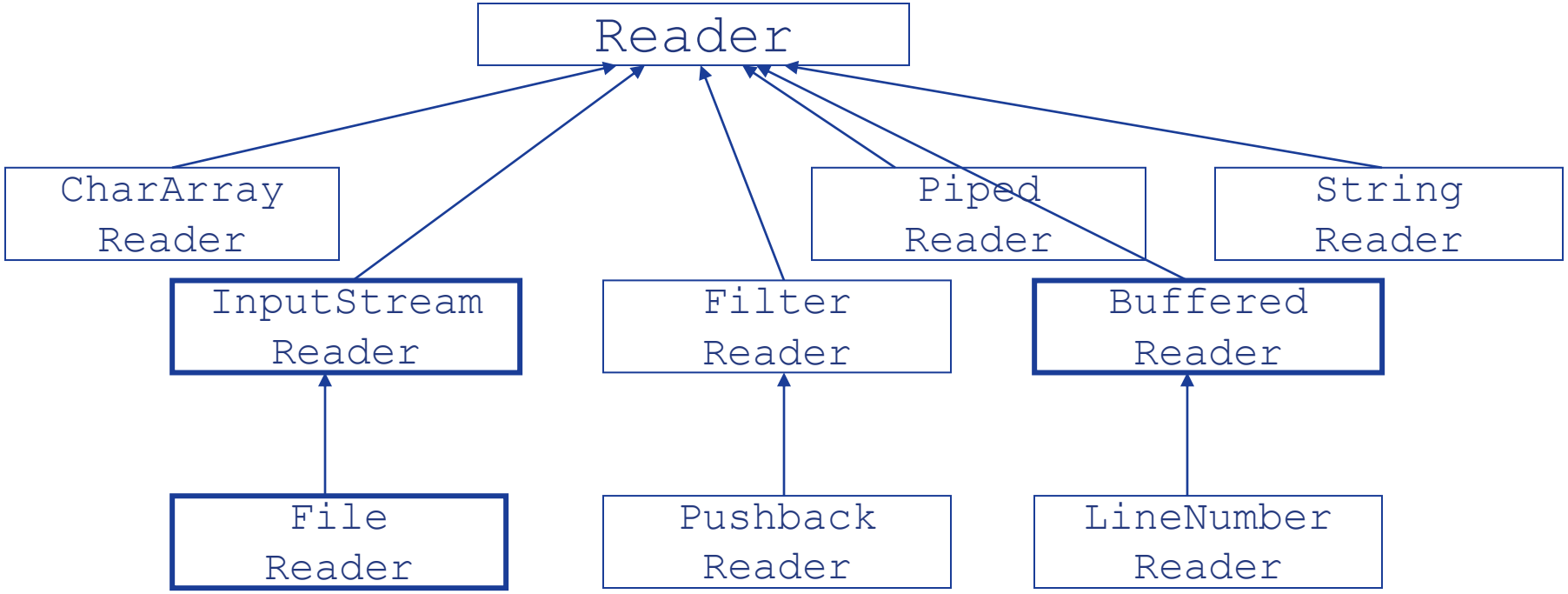
UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

```
int[] originalData = new int[10];
for (int i=0; i<originalData.length; i++)
  originalData[i]=(int)(Math.random()*1000.0);
FileOutputStream fos=null;
try { fos = new FileOutputStream("io2.dat"); }
catch (IOException fe )
  { System.out.println("Cant make new file"); }
DataOutputStream dos = new DataOutputStream(fos);
for (int i=0; i<originalData.length; i++)
  {
  try { dos.writeInt(originalData[i]); }
  catch (IOException ioe)
   {System.out.println("Cant write to file"); }
  }
```

Lớp Reader và Writer

- Là các lớp trừu tượng.
- Là các lớp dạng charater stream
- Chúng nằm tại đỉnh của hệ phân cấp lớp, hỗ trợ việc đọc và ghi các luồng ký tự Unicode.

Lớp Reader

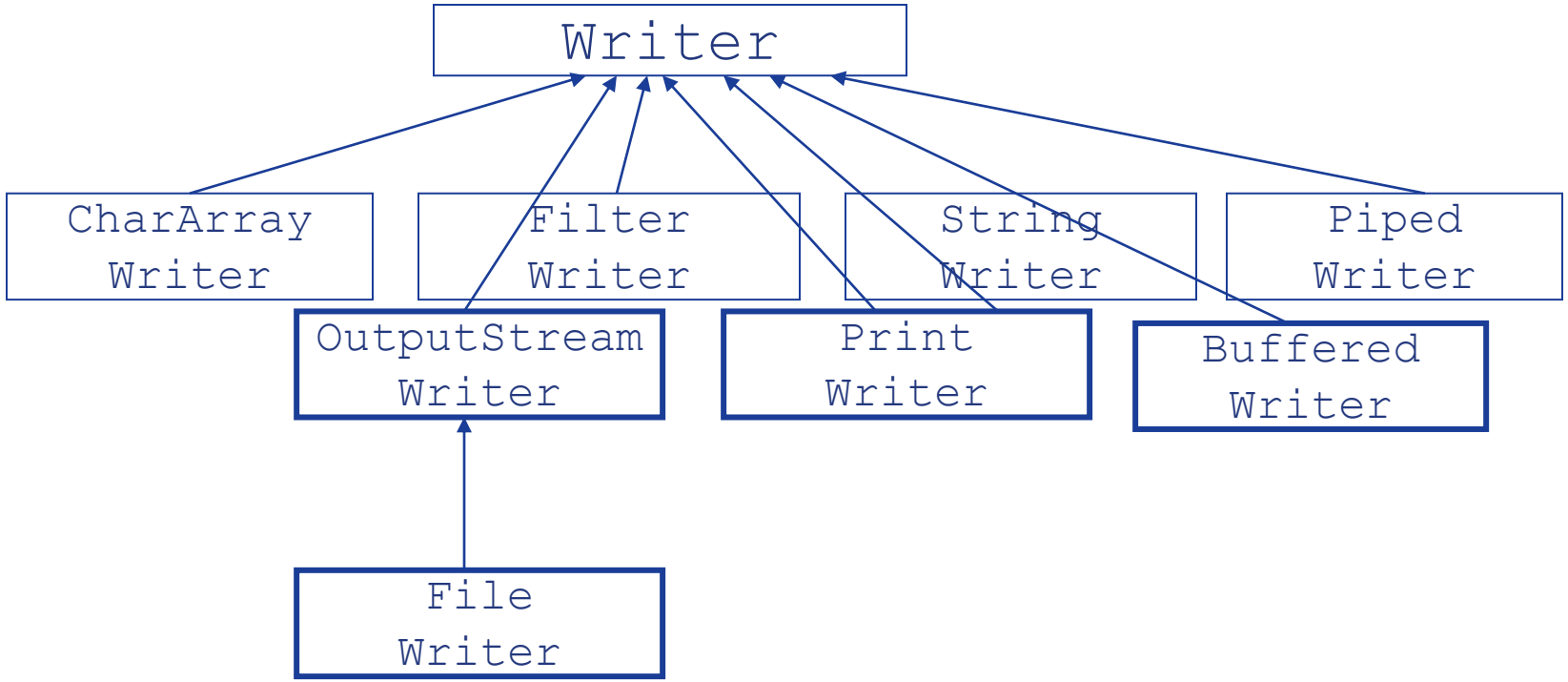

Reader
CharArray Reader
InputStream Reader
Filter Reader
Piped Reader
Buffered Reader
String Reader
File Reader
Pushback Reader
LineNumber Reader

Lớp Reader

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

- Hỗ trợ các phương thức sau:
  - **int read( )**
  - **int read(char[] data)**
  - **int read(char[] data, int offset, int len)**
  - **void reset( )**
  - **long skip( )**
  - **void close( )**
  - **boolean ready( )**

Lớp Writer


Writer
CharArray Writer
Filter Writer
String Writer
Piped Writer
OutputStream Writer
Print Writer
Buffered Writer
File Writer

Lớp Writer

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

- Hỗ trợ các phương thức sau :
  - **void write(int ch)**
  - **void write(char[] text)**
  - **void write(String str)**
  - **void write(String str, int offset, int len)**
  - **void flush( )**
  - **void close( )**

Đọc Tập Tin

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

```
import java.io.*;
public class IntFileRead
{
  public static void main(String[] args)  throws IOException
  {
    File dataf = new File (”vd.txt");
    int number;

    //Create a reader for the file
    FileReader fdat = new FileReader(dataf);
    BufferedReader dat = new BufferedReader(fdat);
    System.out.println("DATA FROM THE FILE:");

    String line = dat.readLine();
    while (line != null)
    {
       number=Integer.parseInt(line);
       System.out.println(number);
       line = dat.readLine();
    }

    System.out.println("END OF FILE REACHED");
    dat.close();
  }//end main
}//end IntFileRead
```

Lớp RandomAccessFile

- Cung cấp khả năng thực hiện I/O theo các vị trí cụ thể bên trong một tập tin.
- Dữ liệu có thể đọc hoặc ghi ngẫu nhiên ở những vị trí bên trong tập tin thay vì một kho lưu trữ thông tin liên tục.
- Phương thức 'seek( )' hỗ trợ truy cập ngẫu nhiên.
- Thực hiện cả đầu vào và đầu ra dữ liệu.
- Hỗ trợ các cấp phép đọc và ghi tập tin cơ bản.
- Kế thừa các phương thức từ các lớp 'DataInput' và 'DataOutput'

Các phương thức của lớp RandomAccessFile

- RandomAccessFile(String fn,String mode)
  "r", "rw", ...
- seek( )
- getFilePointer( )
- length( )
- readBoolean()
- ....
- writeBoolean()
- .....

Constructor lớp RandomAccessFile

```
1 RandomAccessFile(File file, String mode)
2 RandomAccessFile(String file, String mode)
```

File file: là đối tượng file trỏ đến file hiện tại đang muốn đọc, viết
String file: đường dẫn file
mode: là quyền truy cập hay nói cách khác là chế độ.

Mode r: Chỉ đọc
Mode rw: đọc và ghi

Phương thức seek(long pos)

- Phương thức seek(long pos) đưa con trỏ đến vị trí trong file để tiến hành đọc và ghi.
- Giá trị pos được tính từ đầu file.
- Nếu pos lớn hơn chiều dài file, sẽ không làm thay đổi chiều dài file.

RandomAccessFile example

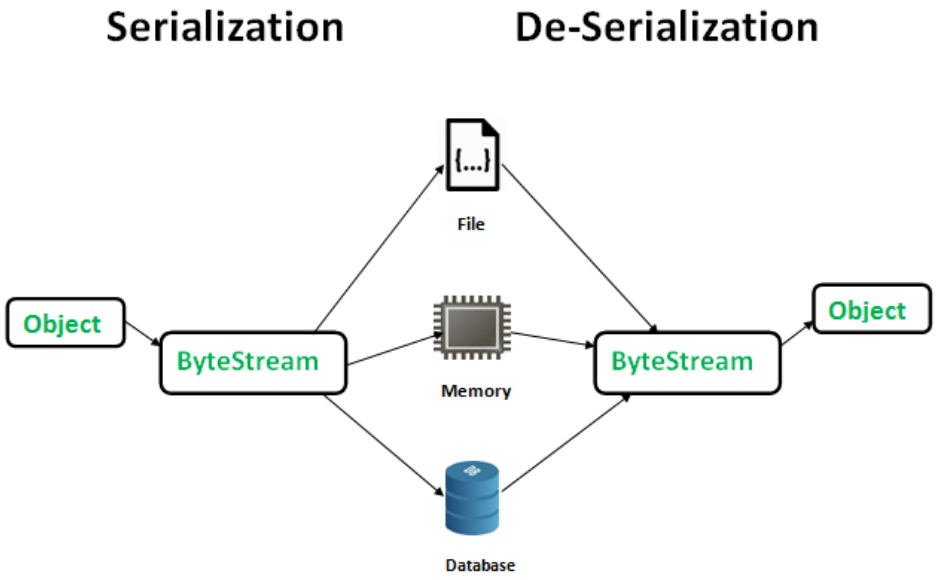
```
RandomAccessFile rf = new
   RandomAccessFile("doubles.dat","rw");

// read third double: go to 2 * 8 (size of one)
rf.seek(8*2);
double x = rf.readDouble();

// overwrite first double value
rf.seek(0);
rf.writeDouble(x);
rf.close();
```

Serialization

- Serialization là cơ chế để chuyển đổi trạng thái của một đối tượng sang 1 byte stream.
- Deserialization là quá trình ngược lại khi byte stream được dùng để tạo ra một đối tượng Java thực tế trong bộ nhớ.



ĐK để có thể serializability

- Đối tượng được serialized nếu:
  - Lớp là public
  - Lớp phải implement Serializable
  - Lớp phải có no-argument constructor

Serialization

- Đề tạo ra một đối tượng có thể có thuộc tính serializable, thì ta cần xây dựng lớp hiện thực hóa giao diện java.io.Serializable

- class A implements Serializable{}

- Sau đó, ta có thể lưu đối tượng được tạo ra này vào file bằng cách gọi hàm writeObject() của ObjectOutputStream hoặc dùng readObject() để đọc ra đối tượng từ file.

ObjectOutputStream and ObjectInputStream

- Lớp ObjectInputStream và **ObjectOutputStream** của gói java.io có thể được sử dụng để đọc/ ghi đối tượng
- Để tạo một **ObjectOutputStream (hay** ObjectInputStream) ta cần có 1 đối tượng FileOutputStream (or FileInputStream) để làm đối số cho PT khởi tạo

- FileOutputStream fileStream = new FileOutputStream(String file);

```
ObjectOutputStream objStream = new
ObjectOutputStream(fileStream);
```

Ví dụ: Tạo một lớp Demo

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

```
class Demo implements java.io.Serializable
{
    public int a;
    public String b;

    // Default constructor
    public Demo(int a, String b)
    {
        this.a = a;
        this.b = b;
    }

}
```

Lưu đối tượng vào file

```
Demo object = new Demo(1, "UIT");
String filename = "file.ser";
try{
        FileOutputStream file = new
FileOutputStream(filename);
        ObjectOutputStream out = new
ObjectOutputStream(file);
        out.writeObject(object);
        out.close();
        file.close();
        } catch(IOException ex) {
        System.out.println("IOException is caught");
       }
```

Đọc đối tượng từ file

```
Demo object1 = null;

    // Deserialization
    try{
        FileInputStream file = new FileInputStream(filename);
        ObjectInputStream in = new ObjectInputStream(file);
        object1 = (Demo)in.readObject();
        in.close();
        file.close();
        } catch(IOException ex) {
        System.out.println("IOException is caught");
    }
```

Tóm tắt bài học

- Nhập xuất dữ liệu trong Java dựa trên mô hình luồng dữ liệu
- Ta có thể sử dụng các lớp nhập xuất dạng byte stream (OutputStream, InputStream) hoặc character stream (Reader, Writer) để đọc ghi dữ liệu phù hợp
- Ngoài ra, ta có thể sử dụng các lớp đệm (Buffered) cho phép quá trình đọc/ ghi được hiệu quả hơn
- Lớp RandomAccessFile hỗ trợ việc đọc ghi file từ một vị trí bất kì trong file nhờ phương thức seek
- Có thể ghi xuống hoặc đọc một đối tượng từ file nhờ cơ chế serialization