LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Bài 6: Vào ra trong Java

Trinh Thi Van Anh – PTIT

Nội dung

- Tổng quan
- Luồng nhập xuất
- Các loại luồng.
- Luồng byte.
- Luồng ký tự.
- Luồng đệm.
- Các luồng nhập/xuất chuẩn.
- Luồng dữ liệu.
- Luồng đối tượng.
- Lớp File.
- Một số ví dụ

Tổng quan

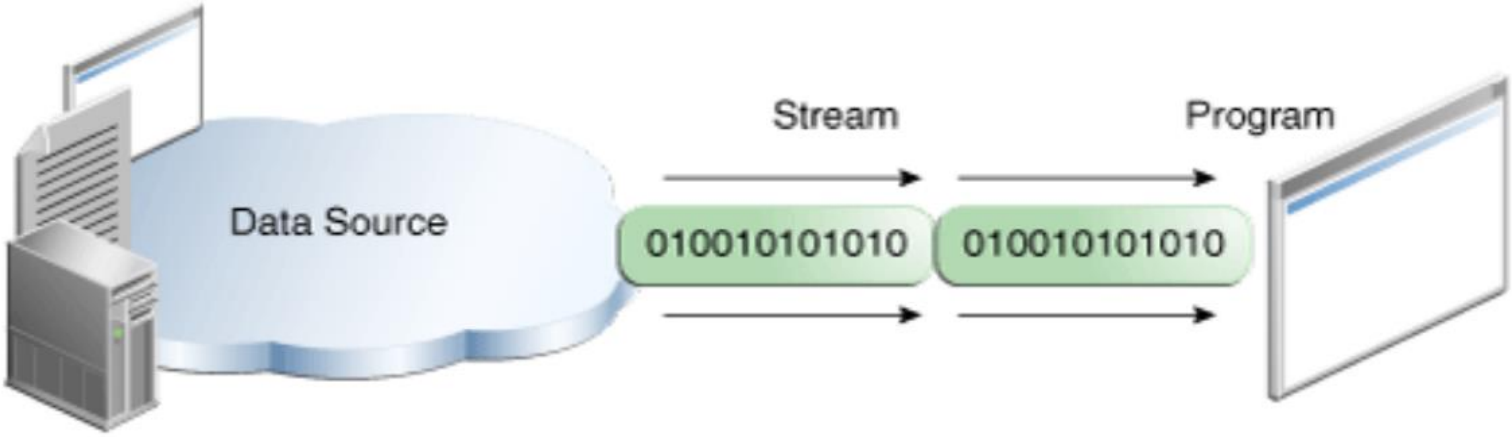
- I/O = Input/Output
- Ở đây là đưa dữ liệu vào (input) và lấy dữ liệu ra (output) từ chương trình
- Input có thể là từ bàn phím hoặc từ file
- Output có thể là ra thiết bị hiển thị (màn hình) hoặc ra file
- Ưu điểm của file I/O
  - Sao lưu trên máy
  - Output từ một chương trình có thể trở thành input cho một chương trình khác
  - Các giá trị input có thể được tự động nhập (thay vì phải gõ từng giá trị)

IO Stream (luồng)

- Luồng là một dòng lưu chuyển của dữ liệu, nhận thông tin từ nguồn và gửi thông tin tới đích.
- I/O Stream diễn tả cho một luồng nhập hoặc luồng xuất.
  - Luồng nhập (input stream): Gắn với các thiết bị nhập như bàn phím, máy scan, file…
  - Luồng xuất (output stream): Gắn với các thiết bị xuất như màn hình, máy in, file…
- Nguồn và đích có thể là: tập tin, các thiết bị, các chương trình khác, kết nối mạng, và bộ nhớ.
- Luồng hỗ trợ nhiều loại dữ liệu khác nhau: byte, các ký tự, các kiểu dữ liệu cơ sở, các đối tượng.
- Package java.io
- Khi làm việc với luồng, chúng ta cần phải bẫy lỗi tường minh lỗi IOException bằng khối try- catch.

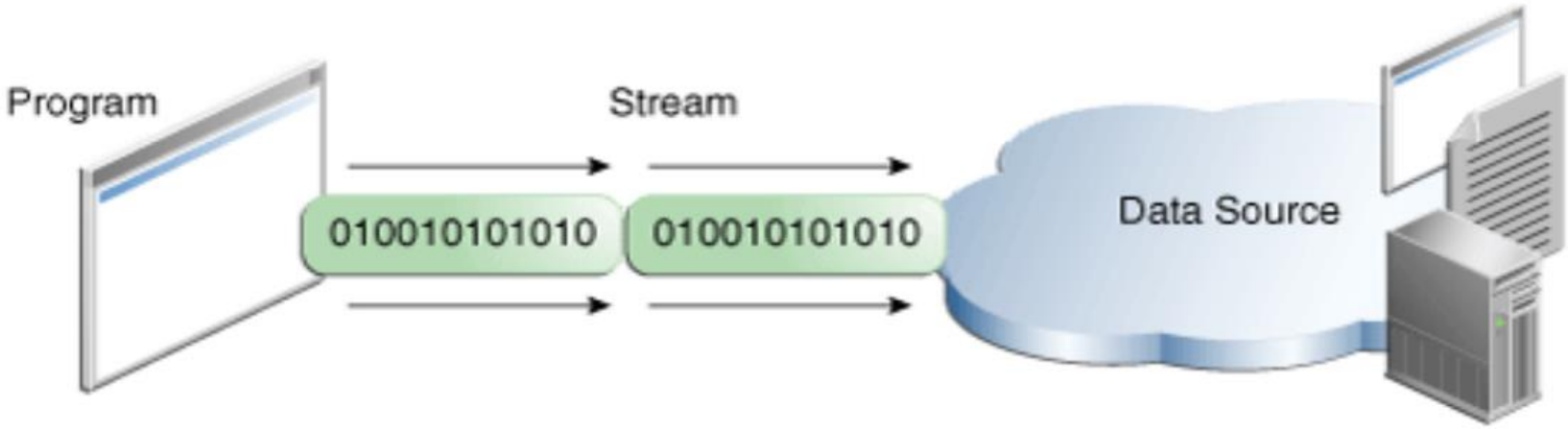
Luồng nhập

- Chương trình sử dụng **luồng nhập** để đọc dữ liệu từ nguồn (mỗi unit tại một thời điểm) đưa vào chương trình:



Luồng xuất

- Chương trình sử dụng luồng xuất để ghi dữ liệu xuống đích (mỗi unit tại một thời điểm).



Các loại luồng

- Luồng ký tự và luồng byte (Character và Byte Streams)
  - Character vs. Byte
- Luồng nhập và luồng xuất (Input và Output Streams).
  - Dựa trên nguồn hoặc đích.
- Luồng dữ liệu đích và luồng lọc (Node và Filter Streams)
  - Dữ liệu trên một luồng có thể được thao tác hoặc chuyển đổi hoặc không.

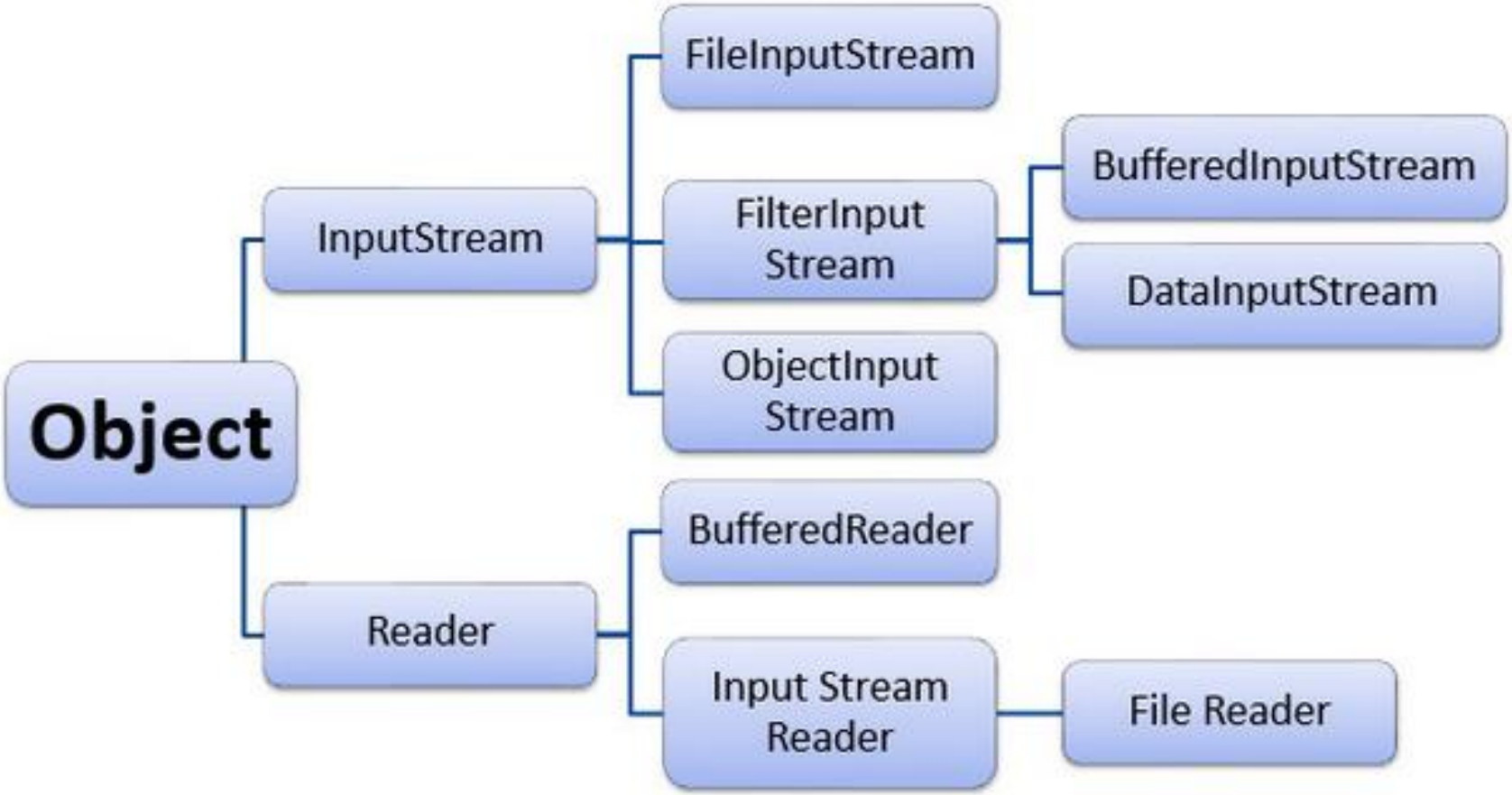
Luồng ký tự và luồng byte

- Luồng byte (byte streams)
  - Cho dữ liệu dạng nhị phân.
  - Các lớp gốc cho luồng byte:
    - InputStream
    - OutputStream
    - Cả 2 lớp là trừu tượng (abstract)
- Luồng ký tự (character streams)
  - Cho các ký tự Unicode.
  - Các lớp gốc cho luồng ký tự:
    - Reader
    - Writer
    - Cả 2 lớp là trừu tượng (abstract)

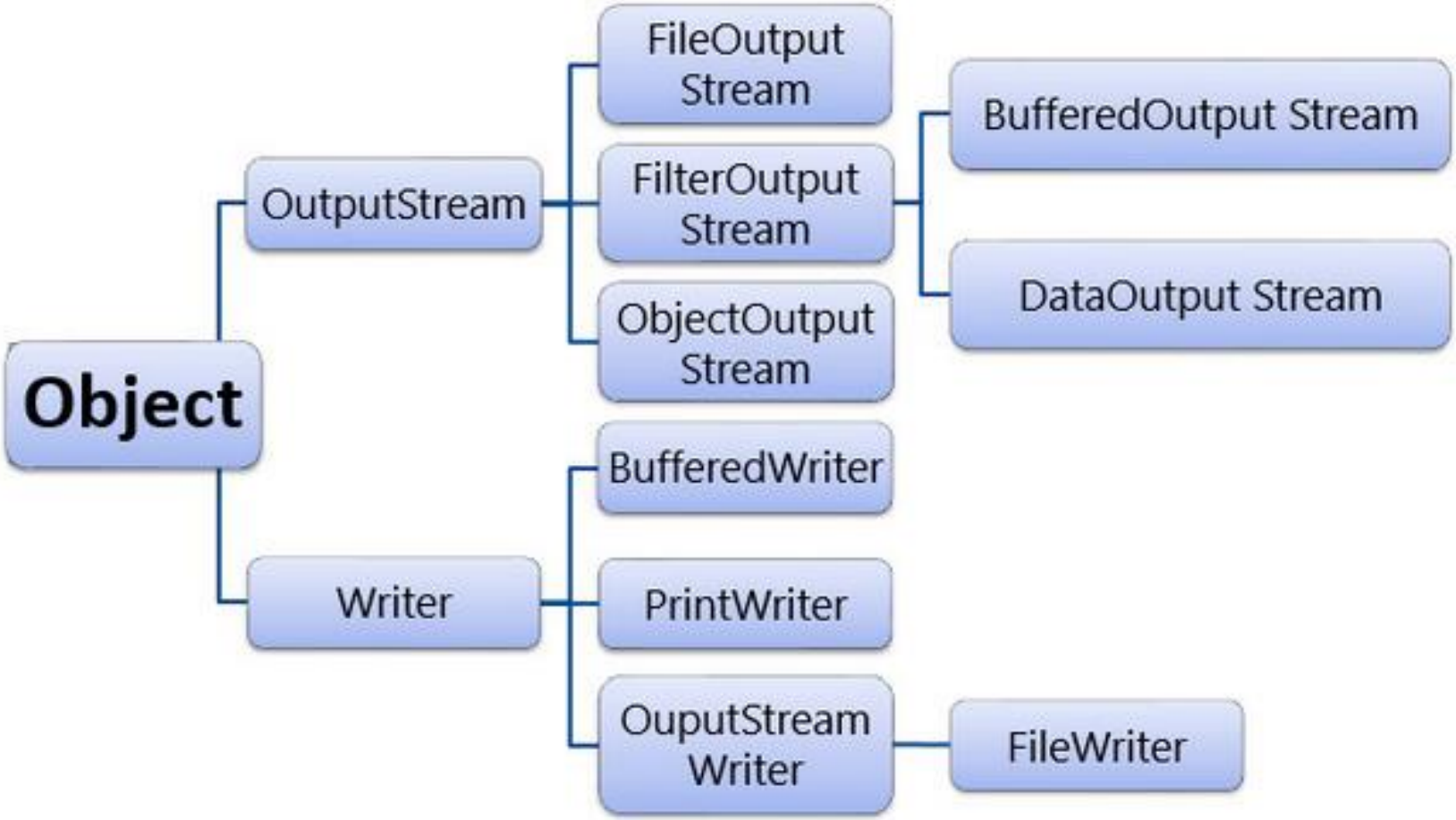
Luồng nhập và luồng xuất

- Luồng nhập (input hoặc source streams)
  - Có thể đọc từ những luồng này.
  - Lớp gốc của tất cảcác luồng nhập:
    - InputStream
    - Reader
- Luồng xuất (output hoặc sink (destination) streams)
  - Có thể ghi từ những luồng này.
  - Các lớp gốc của tất cả các luồng xuất:
    - OutputStream
    - Writer

Kiến trúc Input Stream



Kiến trúc Output Stream



Các thao tác xử lý dữ liệu

- 1.Tạo một luồng và liên kết nó với dữ liệu nguồn (hoặc dữ liệu đích).
- 2.Thao tác dữ liệu (đọc hoặc ghi hoặc cả hai).
- 3.Đóng luồng.

Xử lý nhập xuất dữ liệu sử dụng luồng byte

- Chương trình sử dụng luồng byte (byte stream) để thực hiện nhập xuất theo đơn vị byte.
- Tất cả các lớp của luồng byte đều được dẫn xuất từ lớp **InputStream** và **OutputStream**.
- Có nhiều lớp để thao tác trên luồng byte:
  - **FileInputStream**
  - **FileOutputStream**
- Hầu hết cách thức sử dụng chúng là giống nhau, khác nhau chủ yếu là cách thức được khởi tạo.

Phương thức InputStream

- int **available()** throws IOException :Trả về số byte có thể đọc tiếp
- abstract int **read()** throws IOException :Đọc một byte từ luồng.
  - Nếu cuối luồng sẽ trả về-1
- int **read(byte[] b)** throws IOException : Đọc một mảng byte từ luồng và lưu vào mảng b.
- int **read(byte[] b, int off, int len)** throws IOException : Đọc len byte từ luồng và lưu vào mảng b, bắt đầu tại vị trí off.
- void **close()** throws IOException : Đóng luồng và giải phóng tất cả tài nguyên hệ thống nếu có liên kết với luồng này.
- **FileInputStream**: dùng để đọc các byte từ một tập tin.

Phương thức OutputStream

- void abstract **write(int b)** throws IOException : Ghi byte dữ liệu b.
- Void **write(byte[] b)** throws IOException : Ghi mảng byte b.
- void **write(byte[] b, int off, int len)** : Ghi len byte từ mảng byte b, bắt đầu tại vị trí off.
- void **close()** throws IOException : Đóng luồng xuất và giải phóng tất cả tài nguyên hệ thống liên quan đến luồng này.
- void **flush()** throws IOException : Đẩy dữ liệu còn trên luồng xuống đích.
- **FileOutputStream** dùng để ghi các byte xuống tập tin.

Ví dụ về luồng byte

```
public class CopyFile {
    public static void main(String args[]) {
        FileInputStream in = null;
        FileOutputStream out = null;
        try{
         in = new
  FileInputStream("src/demo/input.txt");
         out = new
  FileOutputStream("src/demo/output.txt");
        }catch (FileNotFoundException ex) {
            System.out.println(ex);
           }
```

```
int c;
    try {
       while ((c = in.read()) != -1) {
         out.write(c);
       }
       if (in != null) {
        in.close();
       }
       if (out != null) {
         out.close();
       }
      } catch (IOException ex) {
          System.out.println(ex);
          }
   }
}
```

Khi nào không sử dụng luồng byte?

- Luồng byte biểu diễn một loại nhập xuất ở mức thấp mà ta nên tránh.
  - Nếu dữ liệu là dữ liệu ký tự, thì phương pháp tốt nhất là sử dụng luồng ký tự.
  - Ngoài ra, còn có nhiều luồng khác thích hợp cho những kiểu dữ liệu phức tạp.
- Các luồng byte chỉ nên sử dụng nhập xuất cho các kiểu nhập xuất cơ bản.
- Tất cả các luồng khác đều dựa trên luồng byte.

Luồng ký tự

- Java hỗ trợ đọc và thao tác trên luồng đối với các ký tự Unicode.
- Luồng ký tự (character stream): Thực hiện các thao tác nhập xuất theo ký tự.
- Tất cả các lớp của luồng ký tự đều được dẫn xuất từ lớp **Reader** và **Writer.**
- Các lớp thao tác trên file của luồng ký tự:
  - **FileReader**
  - **FileWriter**.

Phương thức của lớp Writer

- void **write(int c)** throws IOException
  - Ghi ký tự c được biểu diễn bằng số nguyên.
- void **write(char cbuf[])** throws IOException
  - Ghi mảng ký tự cbuf.
- void **write(char[] cbuf, int off, int len)** throws IOException
  - Ghi len ký tự trong mảng cbuf, bắt đầu tại vị trí off.
- void **write(String str)** throws IOException
  - Ghi một chuỗi ký tự str.
- void **write(String str, int off, int len)** throws IOException
  - Ghi len ký tự trong chuỗi str, bắt đầu tại vị trí off.
- void **flush()** throws IOException
  - Đẩy dữ liệu còn trên luồng xuống đích.
- void **close()** throws IOException
  - Đóng luồng.

Phương thức của lớp Reader

- int **read()** throws IOException
  - Đọc một ký tự.
  - Trảvề -1 nếu cuối luồng.
- int **read(char cbuf[])** throws IOException
  - Đọc các ký tự từ luồng và lưu vào mảng cbuf.
- int **read(char[] cbuf, int off, int len)** throws IOException
  - Đọc len ký tự từ luồng và lưu chúng vào mảng cbuf, bắt đầu tại vị trí off.
- Boolean **ready()** throws IOException
  - Trả về true nếu số ký tự để đọc tiếp vẫn còn trong luồng.
- long **skip(long n)** throws IOException
  - Bỏ qua n ký tự (để đọc các ký tự sau).
- void **close()** throws IOException
  - Đóng luồng.

Ví dụ về luồng ký tự

```
public class CopyCharacters {
   public static void main(String[]
   args) throws IOException {
      FileReader reader = null;
      FileWriter writer = null;
      try {
         reader = new
   FileReader("src/demo/input.txt");
         writer = new
   FileWriter("src/demo/output.txt");
```

```
int c;
        while ((c = reader.read()) != -1) {
          writer.write(c);
        }
      } finally {
        if (reader != null) {
          reader.close();
        }
        if (writer != null) {
          writer.close();
        }
      }
    }
}
```

Luồng Byte và Luồng KýTự

- Luồng ký tự thường là "wrappers" của luồng byte.
- Luồng ký tự sử dụng luồng byte để thực hiện nhập xuất vật lý. Trong khi đó luồng ký tự xử lý chuyển đổi giữa ký tự và byte.
  - **FileReader** dùng **FileInputStream**
  - **FileWriter** dùng **FileOutputStream**
- Dùng luồng ký tự có thể thao tác được cho luồng byte.
- Có thể chuyển từ luồng byte sang luồng ký tự nhờ:
  - **InputStreamReader** và **OutputStreamWriter**.

Line-Oriented I/O

- Thông thường nhập xuất trên ký tự thường xảy ra là một chuỗi các ký tự hơn là các ký tự riêng lẻ.
  - Thông dụng nhất là một dòng: một chuỗi các ký tự với một tín hiệu kết thúc dòng.
  - Tín hiệu kết thúc dòng có thể là \r (carriage-return) hoặc \n (line-feed).

Ví dụ

```
public class CopyLines {
  public static void main(String [] args) throws
   IOException{
      String file = "output.txt";
      Scanner in = new Scanner(System.in);
      FileWriter fw = new FileWriter(file);
      while(true){
         String input = in.nextLine();
         if("DONE".equalsIgnoreCase(input.trim())){
         break;
       }
       fw.write(input+"\n");
       }
       in.close();
       fw.close();
}
}
```

Ví dụ xử lý dãy số

- Nằm trong gói **dayso**
- Có file input.txt chứa dãy số
- Đọc dãy số ra và tính toán rồi lưu vào file output.txt

```
public int[] doc(){
    int[] t;
    String str=null;
    Scanner in = null;
    try {
        FileInputStream f = new
   FileInputStream("src/dayso/input.txt");
        in = new Scanner(f,"UTF-8");
        str=in.nextLine();
    } catch (FileNotFoundException ex) {
         System.out.println("khong thay file");
      }
    String[] temp = null;
    temp = str.split("\\s+");
    t=new int[temp.length];
    for(int i=0;i<temp.length;i++)
        t[i]=Integer.parseInt(temp[i]);
    return t;
    }
```

```
public void viet(String st) {
    FileOutputStream f;
    PrintWriter out;
    try {
        f = new
  FileOutputStream("src/dayso/output.txt");
        out = new PrintWriter(f);
        out.println(st);
        if(out!=null)
           out.close();
    } catch (FileNotFoundException ex) {
        System.out.println("Khong thay
  file");
         }
     }
```

Luồng đệm (buffered stream)

- Nếu một I/O không có bộ đệm, nghĩa là mỗi yêu cầu đọc hoặc ghi được xử lý trực tiếp trên thiết bị.
- Để giảm các chi phí trên, nền tảng Java hỗ trợ luồng nhập xuất có bộ đệm.
  - Luồng nhập có bộ đệm (buffered input stream) đọc dữ liệu từ một vùng nhớ được xem như một bộ đệm; chỉ ghi vào khi nào bộ đệm rỗng.
  - Luồng xuất có bộ đệm (buffered output stream) ghi dữ liệu tới bộ đệm; chờ cho đến khi bộ đệm đầy mới ghi tới đích.

Các lớp của luồng đệm

- Một chương trình có thể chuyển một luồng không bộ đệm thành luồng có bộ đệm (buffered stream).
- Có 4 lớp luồng đệm dùng để "wrap" các luồng không bộ đệm:
- **BufferedInputStream** và **BufferedOutputStream** là các luồng byte có bộ đệm.
- **BufferedReader** và **BufferedWriter** là các luồng ký tự có bộ đệm.

```
public class Test1 {
    public static void main(String [] args) {
      String fileName = “input.txt";
      String line = null;
      try {
        FileReader fileReader = new
  FileReader(fileName);
        BufferedReader reader = new
  BufferedReader(fileReader);
        while((line = reader.readLine()) != null){
          System.out.println(line);
          }
        reader.close();
        }catch(FileNotFoundException ex) {
     System.out.println("Unable to open file “);
        }catch(IOException ex) {
         System.out.println("Error reading file”);
        }    }}
```

```
public class Test2{
  public static void main(String [] args) {
    String fileName = "output.txt";
    try {
      FileWriter fileWriter = new
  FileWriter(fileName);
      BufferedWriter writer = new
  BufferedWriter(fileWriter);
      writer.write("Hello there,");
      writer.write("here is some text.");
      writer.newLine();
      writer.write("We are writing");
      writer.write("the text to the file.");
      writer.close();
   }catch(IOException ex) {
     System.out.println("Error writing to file
  “); }    }}
```

Flushing Buffered Streams

- Vài trường hợp dữ liệu không chứa đủ bộ đệm. Vì vậy, phải dùng flush để ghi hết những gì còn lại trong bộ đệm ra.
- Một vài lớp luồng xuất có bộ đệm hỗ trợ autoflush.
  - Khi chức năng autoflush được thiết lập, cần phải thiết lập sự kiện cụ thể để bộ đệm ghi ra.
  - Ví dụ, autoflush trong đối tượng PrintWriter, bộ đệm ghi ra mỗi khi có lệnh println hoặc format.
- Muốn ghi ra tại thời điểm bất kỳ, ta dùng phương thức flush().

```
public class DemoBufferedStream {
 private final String[] tens = {"Tran An An","Ly
  Hoang Ba","Vu thu Van"};
 private final float[] diemLTHDT = {8.5f,4f, 6.5f};
 private final float[] diemLTM = {5f,7f, 5.5f};
 public void writeTo(String filename) throws
  IOException {
        PrintWriter out = null;
        out = new PrintWriter(new
  BufferedOutputStream(new
  FileOutputStream(filename)),true);
   for(String ten: tens)
     out.print(ten+":");   out.println();
   for(float d1: diemLTHDT)
     out.print(d1+":");    out.println();
   for(float d1: diemLTM)
     out.print(d1+":");
   if(out!=null)
     out.close(); }
```

```
public void readFrom(String filename)
  throws IOException {
    Scanner in = null;
    in = new Scanner(new
  BufferedInputStream(new
  FileInputStream(filename)));
    String line = null;
    String[] ttens=null;
    float[] tdiemLTHDT= new float[3];
    float[] tdiemLTM=new float[3];
    if(in.hasNextLine()){
       line = in.nextLine();
       ttens= line.split(":");
         }
```

```
if(in.hasNextLine()){
    line = in.nextLine();
    String[] tdiem= line.split(":");
    int i=0;
    for(String d:tdiem){
        tdiemLTHDT[i]=Float.parseFloat(d);
        i++;
    }
}
if(in.hasNextLine()){
    line = in.nextLine();
    String[] tdiem= line.split(“:");
    int i=0;
    for(String d:tdiem){
        tdiemLTM[i]=Float.parseFloat(d);
        i++;
    }
}
```

```
if(in!=null)
   in.close();
System.out.println("Du lieu doc ra");
for(String t: ttens)
   System.out.print(t+":");
System.out.println();
for(float d1: tdiemLTHDT)
   System.out.print(d1+":");
   System.out.println();
for(float d1: tdiemLTM)
   System.out.print(d1+":");
}
public static void main(String[] args) throws
   IOException{
   DemoBufferedStream dm = new
   DemoBufferedStream();
   dm.writeTo("src/demo/input.txt");
   dm.readFrom("src/demo/input.txt");
     }}
```

Ví dụ đọc/ghi ma trận

- Nằm trong gói **matran**
- Có file input.txt : 2 dòng đầu cấp của ma trận, các dòng tiếp là 2 ma trận
- Đọc ma trận ra thực hiện sau đó lưu kết quả vào file output.txt (lưu thêm)

```
public int[][] doc1(){
      int m,n;
      int[][] a=null;
      FileInputStream f=null;
      Scanner in=null;
      try{
      f=new
   FileInputStream("src/matran/input.txt");
            in=new Scanner(f, "UTF-8");
            String t=in.nextLine();
            String[] t1=t.split("\\s+");
            m=Integer.parseInt(t1[0]);
            n=Integer.parseInt(t1[1]);
```

```
    a=new int[m][n];
    for(int i=0;i<m;i++){
       t=in.nextLine();
       t1=null;
       t1=t.split("\\s+");
       for(int j=0;j<n;j++)
  a[i][j]=Integer.parseInt(t1[j]);
    }
    in.close();
}catch(FileNotFoundException e){
       System.out.println("Can't find a
  file");
         }
         return a;
    }
```

```
private  void viet(String st, boolean append){
    String path = "src/matran/output.txt";
    File file = new File(path);
    try{
       if (!file.exists()) {
           file.createNewFile();
        }
        FileWriter fw = new
  FileWriter(file.getAbsoluteFile(), true);
        BufferedWriter bw = new
  BufferedWriter(fw);
        bw.write(st);
        bw.write("\n");
        bw.close();
    }catch(IOException ex){
    }
    }
```

```
public void vietMT(int[][] a){
        int m,n;
        m=a.length;
        n=a[0].length;
        String t;
        for(int i=0;i<m;i++){
          t="";
          for(int j=0;j<n;j++)
             t+=a[i][j]+" ";
          viet(t,true);
        }
    }
```

Các luồng chuẩn trên nền Java

- Có 3 luồng chuẩn:
  - Luồng nhập chuẩn - **System.in**
  - Luồng xuất chuẩn - **System.out**
  - Luồng xuất lỗi chuẩn - **System.err**
- System.out, System.err được định nghĩa như các đối tượng PrintStream.

```
public class DemoStandardStream{
   public static void main(String[] args) {
      int[] a= {23,5,78,-4,0,456};
      Scanner in = null;
      try{
        in = new Scanner(System.in);
        System.out.print("Vi tri: ");
        int i=in.nextInt();
        System.out.println("Phan tu vi
  tri:"+i+" la: "+a[i]);
      }catch(ArrayIndexOutOfBoundsException
  e){
  System.err.println("Loi:"+e.toString());
      }finally{
            if(in!=null)
                in.close();
        }
    }}
```

Luồng dữ liệu

- Luồng dữ liệu (data streams) hỗ trợ nhập xuất nhị phân trên các kiểu dữ liệu cơ sở (boolean, char, byte, short, int, long, float, và double) và String.
- Tất cả các luồng dữ liệu được thực hiện từ giao diện **DataInput** hoặc từ **DataOutput**.
- Hầu hết việc nhập xuất trên luồng dữ liệu thì dùng lớp **DataInputStream** và **DataOutputStream**.
- Những dữ liệu được ghi bởi DataOutputStream sẽ đọc được bởi DataInputStream

Phương thức DataOutputStream

- Constructor: DataOutputStream(OutputStream out)
- public final void writeByte(int 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 writeUTF(String s) throws IOException
- …

Ví dụ DataOutputStream

```
public class Test3{
  public static void main(String[] args){
    try {
      DataOutputStream out =
      new DataOutputStream(new
  BufferedOutputStream(new
  FileOutputStream("src/demo/out.txt")));
      String testString = "Day la vi du
  viet vao file.\n";
      out.writeUTF(testString);
      int writeSize1 = out.size();
      System.out.println("writeUTF writes
  " + writeSize1 + " bytes.");
```

```
out.writeChars(testString);
int writeSize2 =out.size()-
  writeSize1;
System.out.println("writeChars
  writes " + writeSize2 +
  "bytes.\n");
    out.close();
} catch ( IOException ex ) {
    ex.printStackTrace();
}
}}
```

```
public class DemoDataStream{
  private final String[] tens = {"Tran An An","Ly
  Hoang Ba","Vu thu Van"};
  private final float[] diemLTHDT = {8.5f,4f, 6.5f};
  private final float[] diemLTM = {5f,7f, 5.5f};
  public void writeTo(String filename) throws
  IOException {
        DataOutputStream out = null;
        try{
        out = new DataOutputStream(new
  BufferedOutputStream(new
  FileOutputStream(filename)));
        for(int i=0;i<tens.length;i++){
            out.writeUTF(tens[i]);
            out.writeFloat(diemLTHDT[i]);
            out.writeFloat(diemLTM[i]);         }
        }finally{
        if(out!=null)
            out.close();     }    }}
```

Phương thức DataInputStream

- Constructor: DataInputStream(InputStream in)
- public final byte readByte() throws IOException
- public final char readChar() throws IOException
- public final short readShort() throws IOException
- public final int readInt() throws IOException
- public final long readLong() throws IOException
- public final String readUTF() throws IOException
- …

```
public class DataInputStreamDemo {
  public static void main(String[] args)
  throws IOException {
    InputStream is = null;
    DataInputStream dis = null;
    try{
      is = new
  FileInputStream("src/demo/input.txt");
      dis = new DataInputStream(is);
      int length = dis.available();
      byte[] buf = new byte[length];
      dis.readFully(buf);
      for (byte b:buf){
         char c = (char)b;
         System.out.print(c);
         }
```

```
}catch(Exception e){
     e.printStackTrace();
}finally{
     if(is!=null)
        is.close();
     if(dis!=null)
        dis.close();
      }   }}
```

Tuần tự hóa dữ liệu

- Tính bền vững (persistence) là khả năng một đối tượng duy trì sự tồn tại độc lập sau thời gian sống của chương trình tạo ra nó.
- Java cung cấp cơ chế được gọi là tuần tự hóa đối tượng (Object **Serialization**) để tạo đối tượng bền vững.
- Khi một đối tượng được tuần tự hóa, nó sẽ được chuyển thành tuần tự các byte dạng thô, biểu diễn đối tượng.

Luồng đối tượng (Object Streams)

- Luồng đối tượng (Object Streams) hỗ trợ việc đọc, ghi các đối tượng.
- Nếu đối tượng thực hiện giao diện **Serializable** thì ta có thể sử dụng luồng đối tượng để đọc, ghi đối tượng đó.
- Hai lớp hỗ trợ luồng đối tượng:
  - **ObjectInputStream**
  - **ObjectOutputStream**
- Hai lớp này tương ứng thực hiện các giao diện:
  - **ObjectInput**
  - **ObjectOutput**

Luồng đối tượng

- Bất kỳ đối tượng nào mà ta muốn tuần tự hóa (serialize) thì bắt buộc phải hiện thực giao diện Serializable.
- Để tuần tự hóa một đối tượng, gọi phương thức **writeObject** của lớp **ObjectOutputStream**.
- Để khôi phục lại đối tượng đã được tuần tự hóa trước đó (deserialize), gọi phương thức **readObject** của lớp **ObjectInputStream**.
- Các đối tượng được tuần tự hóa có thể được ghi vào file, truyền qua mạng hoặc có thể chuyển sang các luồng khác.

Ví dụ viết/đọc 1 đối tượng

- Nằm trong gói **object**
- Ghi 1 đối tượng (Book) vào file
- Đọc 1 đối tượng (Book) từ file.
  - Tạo lớp Book `implements Serializable`
  - Tạo lớp ReadWriteObject, thực thi :
    - Đọc dữ liệu cho 1 quyển sách
    - Mở luồng viết (sach.dat)
    - Viết đối tượng, đóng luồng viết
    - Mở luồng đọc (sach.dat)
    - Đọc ra (bb), viết ra màn hình

```
public class Book implements Serializable {
    private String maSach;    private String tenSach;
    private String tacgia;  private String nhaxuatban;
    private double gia;       private int soluong;
    public Book() {}
    public Book(String maSach,String tenSach,  String
   tacgia, String nhaxuatban, double gia, int soluong) {
         this.maSach = maSach;
         this.tenSach = tenSach;
         this.tacgia = tacgia;
         this.nhaxuatban = nhaxuatban;
         this.gia = gia;
         this.soluong = soluong;
    }
// getter and setter….
public String toString(){
      return maSach+"  "+ tenSach+"  "+ tacgia+"  "+
   nhaxuatban+"  "+ gia+"  "+ soluong;
    }}
```

```
public class ReadWriteObject {
    public static void main(String[] args){
        Scanner in = new Scanner(System.in);
        System.out.println("Nhập dữ liệu cho sách");
        System.out.print("Ma sach: ");
        String ma = in.nextLine();
        System.out.print("Ten sach: ");
        String ten = in.nextLine();
        System.out.print("Ten tac gia: ");
        String tentg = in.nextLine();
        System.out.print("ten nha XB: ");
        String nxb = in.nextLine();
        System.out.print("Don gia: ");
        double gia =
  Double.parseDouble(in.nextLine());
        System.out.print("So luong: ");
        int sl = Integer.parseInt(in.nextLine());
        Book b = new Book(ma,ten,tentg,nxb,gia,sl);
```

```
try {
        FileOutputStream f = new FileOutputStream("book.dat");
        ObjectOutputStream out = new ObjectOutputStream(f);
        out.writeObject(b);
        out.close();
    } catch (IOException e) {
            System.out.println("Error Write file");
         }
       Book bb = null;
       try {
         FileInputStream f = new FileInputStream("book.dat");
         ObjectInputStream in1 = new ObjectInputStream(f);
         bb = (Book) in1.readObject();
         in1.close();
      } catch (ClassNotFoundException e) {
          System.out.println("Class not found");
      } catch (IOException e) {
          System.out.println("Error Read file");
      }
         System.out.println("Du lieu doc tu file:");
         System.out.println(bb.toString());
    }}
```

Ví dụ viết/đọc 1 mảng đối tượng

- Nằm trong gói **bookarray**
- **Ghi** 1 mảng các đối tượng (Book) vào file
- **Đọc** 1 mảng các đối tượng (Book) từ file.
  - Tạo lớp Book `implements Serializable`
  - Tạo lớp ProcessBook, gồm :
    - Book[] creat()
    - void write(Book[] b)
    - Book[] read()
    - void show(Book[] b)
  - Tạo lớp Menu để chạy chương trình

```
class ProcessBook {
  public Book[] creat() {
    int n;
    Scanner in = new Scanner(System.in);
    System.out.print("Nhap so sach: ");
    n = Integer.parseInt(in.nextLine());
    Book[] b = new Book[n];
    for (int i = 0; i < n; i++) {
      System.out.print("Ma sach: ");
      String ma = in.nextLine();
      System.out.print("Ten sach: ");
      String ten = in.nextLine();
```

```
System.out.print("Ten tac gia: ");
String tentg = in.nextLine();
System.out.print("ten nha XB: ");
String nxb = in.nextLine();
System.out.print("Don gia: ");
double gia =
  Double.parseDouble(in.nextLine());
System.out.print("So luong: ");
int sl = Integer.parseInt(in.nextLine());
b[i] = new Book(ma,ten,tentg,nxb,gia,sl);
}
return b;
}
```

```
public void write(Book[] b) {
   try {
      FileOutputStream f = new
   FileOutputStream("sach.dat");
      ObjectOutputStream out = new
   ObjectOutputStream(f);
       out.writeObject(b);
       out.close();
   } catch (IOException e) {
      System.out.println("Loi ghi file");          }
   }
public Book[] read() {
    Book[] b = null;
    try {
      FileInputStream f = new
   FileInputStream("sach.dat");
      ObjectInputStream in = new
   ObjectInputStream(f);
      b = (Book[]) in.readObject();
      in.close();
```

```
}catch (ClassNotFoundException e) {
    System.out.println("khong co lop");
} catch (IOException e) {
    System.out.println("Loi doc file");
  }
 return b;
}
public void show(Book[] b) {
   System.out.println("STT Ma sach   Ten
  sach      Tac gia   Nha XB   Don gia So
  luong");
   for (int i = 0; i < b.length; i++)
      System.out.println((i + 1) +
  b[i].toString());
}}
```

Lớp File

- Lớp File dùng cho việc thao tác trên file và thư mục.
- Một số phương thức của lớp File:
  - boolean exists(); // kiểm tra sự tồn tại của file
  - boolean isDirectory(); // kiểm tra xem file có phải là thư mục
  - String getParent(); // lấy thư mục cha
  - long length(); // lấy kích cỡ file (byte)
  - long lastModified(); // lấy ngày sửa file gần nhất
  - String[] list(); // lấy nội dung của thư mục
  - boolean createNewFile(); //tạo file

Viết thêm 1 đối tượng

- Viết **đè** vào file

```
FileOutputStream fos = new
   FileOutputStream(filename);
ObjectOutputStream oos = new
   ObjectOutputStream(fos);
```

- Viết **tiếp** vào file

```
FileOutputStream fos = new
   FileOutputStream(filename,true);
ObjectOutputStream oos = new
   ObjectOutputStream(fos) {
   @Override
   protected void writeStreamHeader() throws
   IOException { reset(); }
              };
```

Ví dụ viết/đọc thêm 1 đối tượng

- Nằm trong gói **bookmore**
- Ghi thêm 1 đối tượng (Book) vào file
- Đọc 1 danh sách đối tượng (Book) từ file.
  - Tạo lớp Book `implements Serializable`
  - Tạo lớp ProcessBook, gồm :
    - void Input()
    - boolean hasObject(File f)
    - ObWrite(Book b, boolean append)
    - ArrayList<Book> readAppend()
    - show(ArrayList<Book> list)
  - Tạo lớp Menu để chạy chương trình

```
class ProcessBook {
    public void Input() {
        Scanner in = new Scanner(System.in);
        System.out.println("= Them moi sach ===");
        System.out.print("Ma sach: ");
        String ma = in.nextLine();
        System.out.print("Ten sach: ");
        String ten = in.nextLine();
        System.out.print("Ten tac gia: ");
        String tentg = in.nextLine();
        System.out.print("ten nha XB: ");
        String nxb = in.nextLine();
        System.out.print("Don gia: ");
        double gia =
  Double.parseDouble(in.nextLine());
        System.out.print("So luong: ");
        int sl = Integer.parseInt(in.nextLine());
        Book b = new Book(ma,ten,tentg,nxb,gia,sl);
        ObWrite(b, true);
    }
```

```
public static boolean hasObject(File f) {
        FileInputStream fi;
        boolean check = true;
        try {
            fi = new FileInputStream(f);
            ObjectInputStream in = new
  ObjectInputStream(fi);
            if (in.readObject() == null) {
                check = false;
            }
            in.close();
        } catch (FileNotFoundException e) {
            check = false;
        } catch (IOException e) {
            check = false;
        } catch (ClassNotFoundException e) {
            check = false;
        }
        return check;
    }
```

```
public void ObWrite(Book b, boolean append) {
    String path = "sach.dat";
    File file = new File(path);
    OutputStream out;
    try {
        ObjectOutputStream ois;
        if (!hasObject(file)) {
            out = new FileOutputStream(file);
            ois = new ObjectOutputStream(out);
        } else {
            out = new FileOutputStream(file, append);
            ois = new ObjectOutputStream(out) {
         @Override
protected void writeStreamHeader() throws IOException {
                    reset();
                } };
        }
        ois.writeObject(b);
        ois.flush();    ois.close();
        out.flush();    out.close();
    } catch (IOException ex) {
        System.out.println("Khong tim thay file");  }
    }
```

```
public static ArrayList<Book> readAppend() {
    String path = "sach.dat";
    File f = new File(path);
    ArrayList<Book> list = new ArrayList<>();
    ObjectInputStream obin = null;
    try {
        FileInputStream in = new FileInputStream(f);
        obin = new ObjectInputStream(in);
        Object obj = null;
        while ((obj = obin.readObject()) != null) {
            list.add((Book) obj);
        }
    } catch (Exception ex) {
    } finally {
        if (obin != null) {
            try {
                obin.close();
            } catch (IOException ex) {
                System.out.println("Loi doc file");
            }
        }    }
    return list;
}
```

```
public void show(ArrayList<Book> list) {
    System.out.println("STT Ma sach   Ten
  sach       Tac gia   Nha XB   Don gia So
  luong");
    for (int i = 0; i < list.size(); i++)
       System.out.println((i+1) +
  list.get(i).toString());
     }
}
```

```
public class Menu {
    public static void main(String[] args) {
        ProcessBook pb = new ProcessBook();
        ArrayList<Book> list= null;
        while (true) {
            System.out.println("1. Them moi sach");
            System.out.println("2. Xem danh sach sach");
            System.out.println("\n 0. Thoat ");
            System.out.print("\n HAY CHON 1,2 hoac 0: ");
            Scanner in = new Scanner(System.in);
            int choice = in.nextInt();
            switch (choice) {
                case 1:
                    pb.Input();
                    break;
                case 2:
                    list = pb.readAppend();
                    pb.show(list);
                    break;
                case 0: System.out.println("\n Bye!!!");
                        System.exit(0);
                        break;
           default:System.out.println("\n HAY CHON 1,2 hoac 0 ");
                        break;
             }
    }  }}
```

Ví dụ thao tác với đối tượng sách

- Nằm trong gói **qlsach**
- Lớp Book (**entity**) `implements Serializable`
- Lớp IOFile (**entity**) : viết/đọc danh sách vào/từ file
  - outFile(List ob, String s)
  - inFile(List ob, String s)
- Lớp DSBook (**controller**) : thao tác với danh sách
  - themSach() throws IOException
  - suaSach() throws IOException
  - xoaSach() throws IOException
  - sort(int mode)

- searchByMa(String ma)
- searchByGia(double from, double to)
- inDanhSach()
- inDSSapXep()
- inDSTimKiem()
- getAllNxb()
- thongkeNxb()
- getAllTacgia()
- thongkeTacgia()

- Lớp SortByMa
- Lớp SortByDonGia
- SortByTenTacGia
- Lớp menu để chạy chương trình

Ví dụ với 3 đối tượng (ViDu3DT)

- Bài toán quản lý sách: có 3 đối tượng gồm Loại sách, Nhà xuất bản và Sách. Trong đó
- Loại sách: mã loại, tên loại sách
- Nhà xuất bản: mã nhà xuất bản, tên nhà xuất bản, địa chỉ nhà xuất bản và điện thoại
- Sách: mã sách, tên sách, tên tác giả, giá sách, số lượng sách, **loại sách** và **nhà xuất bản**
- 3 đối tượng được lưu trên 3 file: loaisach.dat, nhaxb.dat và sach.dat
- Xây dựng chương trình: Quản lý sách