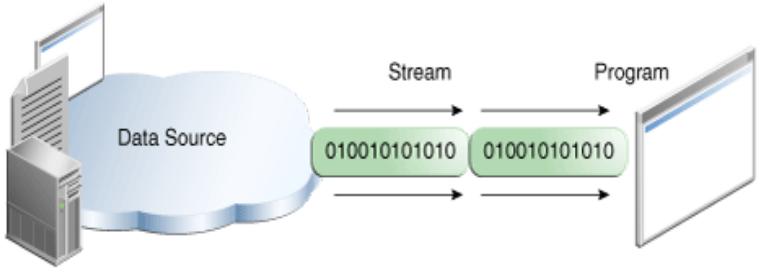
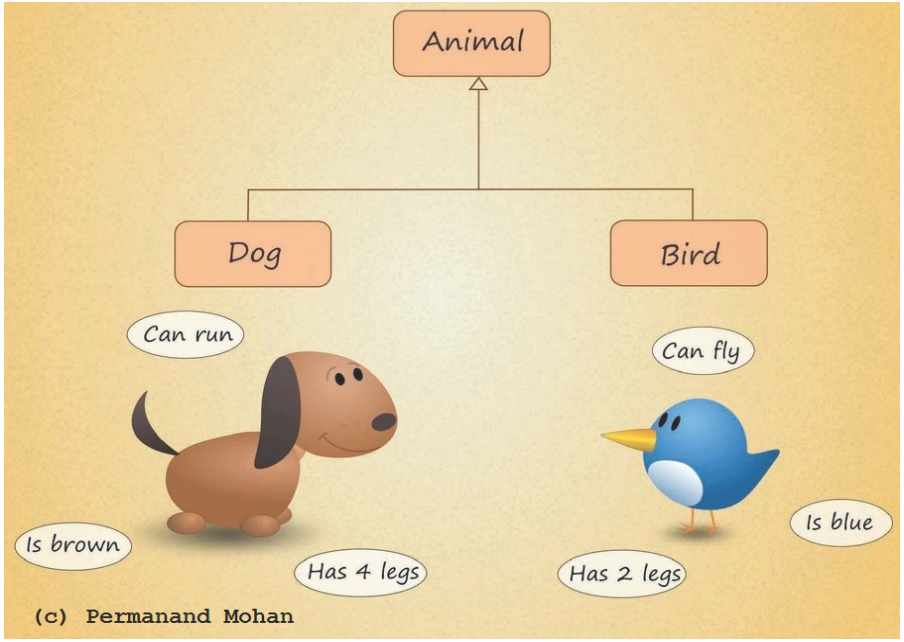




Chapter 4

Vào ra (I/O) trong Java

CT176 – LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

TS. Triệu Thanh Ngoan

ttngoan@cit.ctu.edu.vn

Khoa MMT&TT, Trường CNTT-TT

Mục tiêu

Chương này nhằm giới thiệu
các kỹ thuật vào ra – nhập xuất (I/O) trong Java

Nội dung

- Giới thiệu
- Lớp java.io.File
- Dòng nhập xuất (I/O Stream)
- Các dòng nhập xuất theo byte
- Các dòng nhập xuất theo ký tự
- Nhập xuất đối tượng
- Tập tin truy cập ngẫu nhiên

Giới thiệu về nhập xuất trong Java

- Các gói hỗ trợ nhập xuất trong JDK:
  - **java.io**: nhập xuất chuẩn (standard I/O)
    - Được giới thiệu từ JDK 1.0
    - Nhập xuất thông qua Stream
  - **java.nio**: nhập xuất mới (new I/O)
    - Được giới thiệu từ JDK 1.4
    - Nâng cao hiệu quả việc nhập xuất qua vùng đệm.
    - JDK 1.7 hỗ trợ nhập xuất file nâng cao
- JDK 1.5 giới thiệu thêm lớp java.util.Scanner
  - Hỗ trợ nhập xuất với các kiểu dữ liệu cơ bản, chuỗi.
  - Tách biểu thức thông thường thành các token.

Lớp java.io.File

- Đối tượng **File** biểu diễn 1 tập tin hoặc 1 thư mục.
- Khởi tạo 1 đối tượng
  - `public File(String pathString)`
- Sử dụng đường dẫn (path) theo dạng:
  - Trong Windows: "C:\ViduJava\Hello.java"
  - Trong Unix/Mac: "/ViduJava/Hello.java"
- Ví dụ:
  - `File f1 = new File("data.txt");`
  - `File f2 = new File("C:\\ViDu\\Hello.java");`
  - `File dir1 = new File("C:\\temp");`

Lớp java.io.File

- Một số các phương thức quan trọng
  - `public boolean exists();` *// Có tồn tại hay không*
  - `public long length();` *// Kích thước file*
  - `public boolean isDirectory();` *// Là thư mục?*
  - `public boolean isFile();` *// Là tập tin?*
  - `public boolean canRead();` *// Có thể đọc?*
  - `public boolean canWrite();` *// Có thể ghi?*
  - `public boolean delete();` *// Xóa*
  - `public void deleteOnExit();` *// Xóa khi kết thúc*
  - `public boolean renameTo(File dest);` *// Đổi tên*
  - `public boolean mkdir();` *// Tạo thư mục*
  - `public String[] list();` *// Liệt kê thư mục dạng chuỗi*
  - `public File[] listFiles();` *// Liệt kê thư mục dạng File*

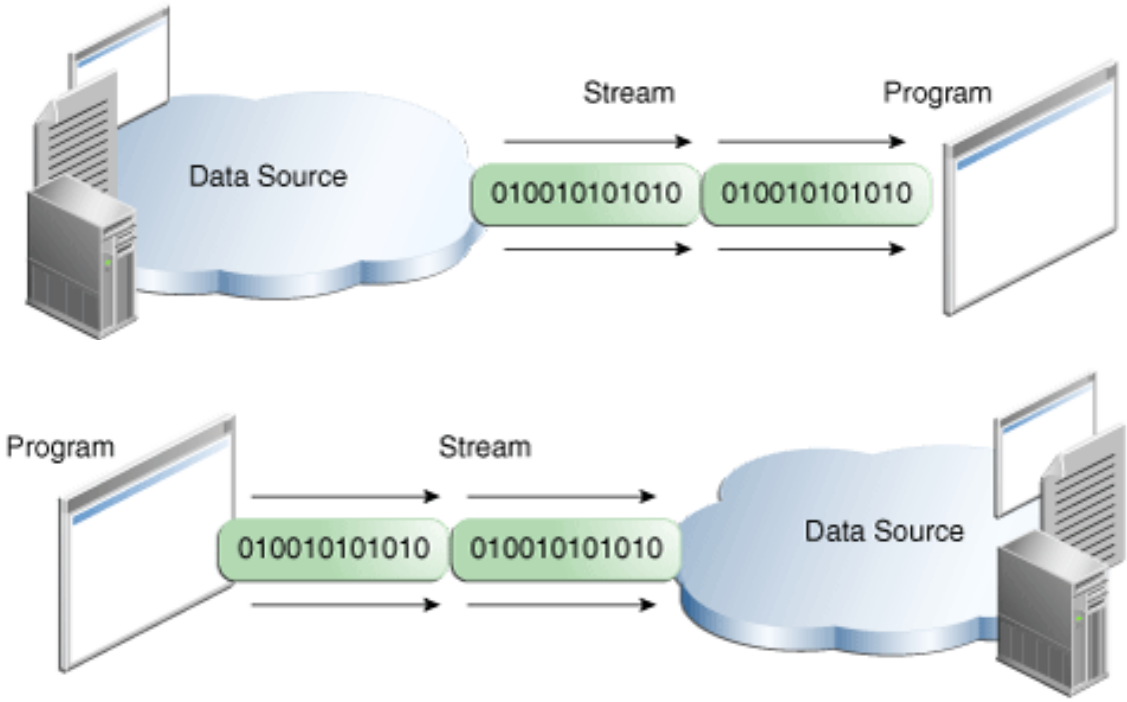
Ví dụ về lớp java.io.File

```
import java.io.File;
public class ListDirectoryRecursive {
   public static void main(String[] args) {
      File dir = new File("C:\\ViduJava");
      listRecursive(dir);
   }

   public static void listRecursive(File dir) {
      if (dir.isDirectory()) {
         File[] items = dir.listFiles();
         for (File item : items) {
            System.out.println(item.getAbsolutePath());
            if (item.isDirectory()) listRecursive(item);
         }
      }
   }
}
```

Liệt kê nội dung thư mục C:\ViduJava

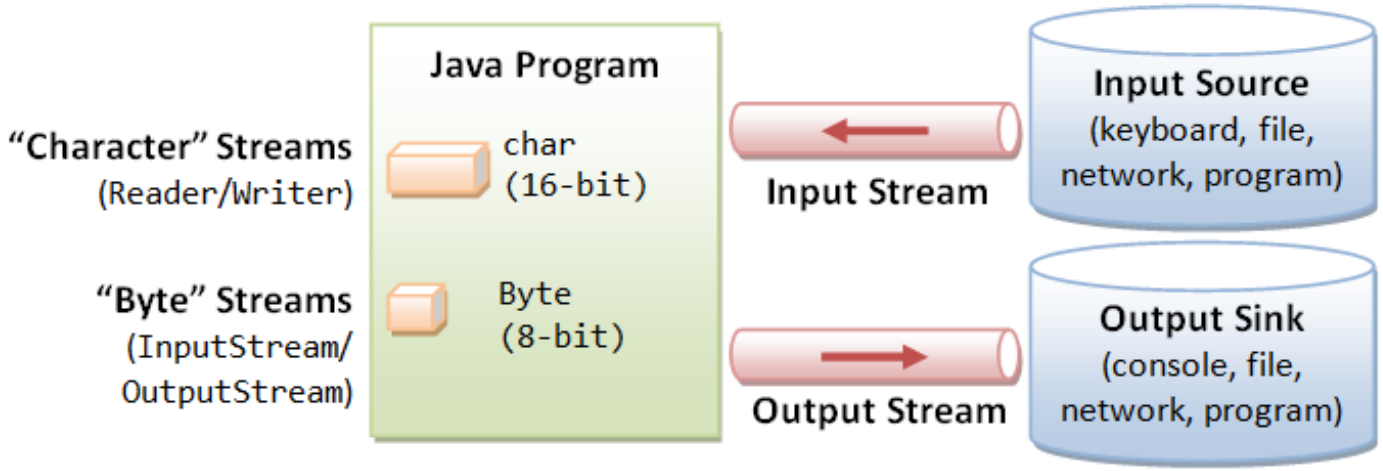
Khái niệm về Stream



Trong Java, việc nhập xuất - vào ra (I/O) được thực hiện thông qua các stream.

Stream là 1 dòng liên tục, có thứ tự (chỉ đi theo 1 chiều) dùng để chuyển dữ liệu giữa chương trình và nguồn dữ liệu (các thiết bị ngoại vi).

Các dòng nhập xuất chuẩn



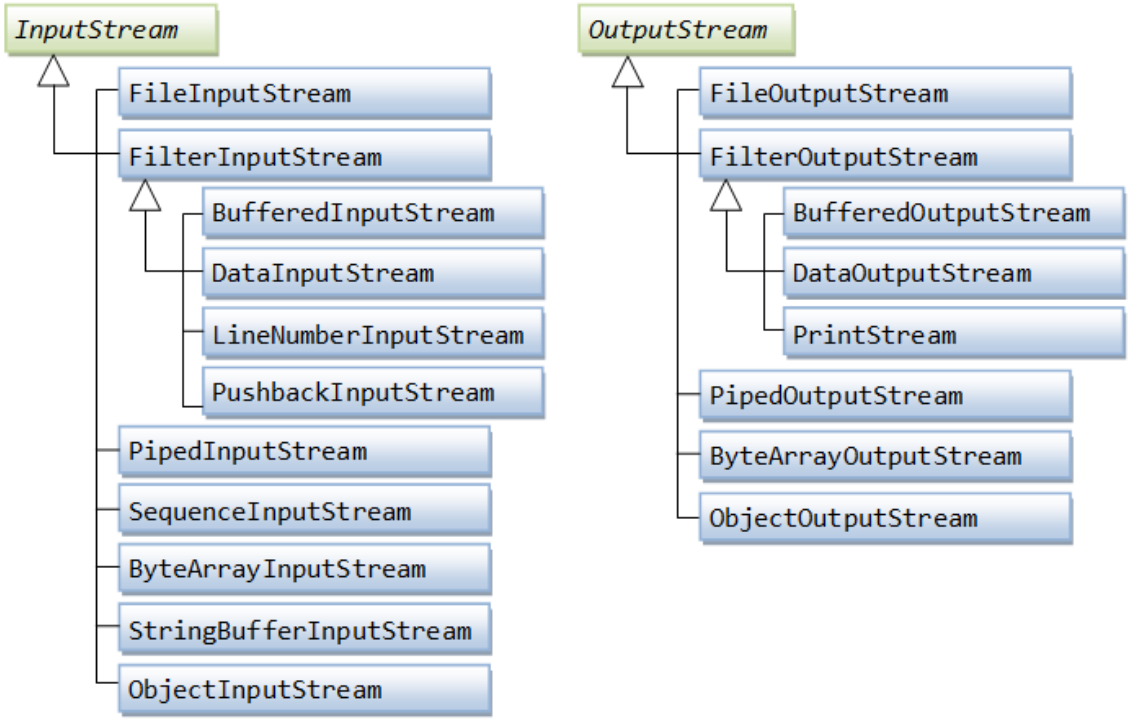
Java phân biệt 2 loại dòng nhập xuất:

- Theo byte (byte-based I/O): xử lý dữ liệu thô hay nhị phân.
- Theo ký tự (character-based I/O): xử lý dữ liệu text.

Các dòng nhập xuất chuẩn

- Để hỗ trợ các thao tác xuất nhập, gói java.io cung cấp các lớp định nghĩa các dạng dòng xuất nhập khác nhau.
  - Reader: Stream đọc vào dữ liệu dạng ký tự.
  - Writer: Stream xuất dữ liệu dạng ký tự.
  - InputStream: Hỗ trợ đọc dữ liệu dạng byte.
  - OutputStream: Viết dữ liệu dạng byte.

Các dòng nhập xuất theo byte



- Sử dụng để đọc/ghi (read/write) các byte dữ liệu thô (raw data) từ/đến các thiết bị ngoại vi.
- Thừa kế từ 2 lớp cha là InputStream và OutputStream.

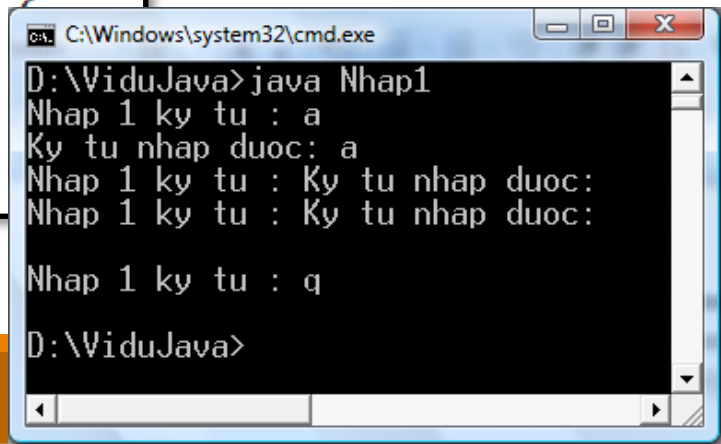
Lớp InputStream

- `public abstract int read() throws IOException;`
  - Đọc 1 ký tự từ thiết bị ngoại vi nối với InputStream.
  - Kết quả nhận về là thứ tự của ký tự trong bảng mã ASCII (**0-255**).
  - Kết quả là -1 nếu cuối dòng (hết dữ liệu trong stream).
- `public int read(byte[] b) throws IOException;`
  - Đọc nhiều ký tự, kết quả lưu vào mảng byte b[]
  - Trị trả về là số lượng byte nhận được.
- `public int read(byte[] b, int offset, int length) throws IOException;`
  - Đọc n ký tự, lưu vào mảng b[] từ vị trí *offset* chiều dài là *length*.
- `public int available() throws IOException;`
- `public long skip(long n) throws IOException;`
- `public void close() throws IOException;`

Ví dụ về InputStream

```
import java.io.*;
public class Nhap1  {
    public static void main(String[] args)  {
        InputStream  is = System.in;   // Ban phim
        while(true) {
        try {
            System.out.print("Nhap 1 ky tu : ");
            int  ch = is.read();
            if (ch==-1 || ch=='q')   break;
            System.out.println("Ky tu nhap duoc: "+ (char)ch);
         }
         catch(IOException  e)
             {  System.out.println("Co loi ve nhap xuat");  }
      }
    }
}
```

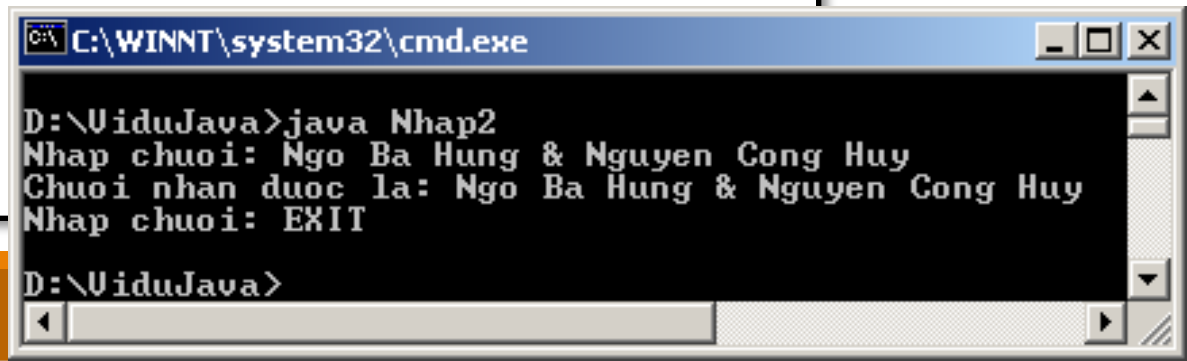
- Nhập từng ký tự từ bàn phím.
- Hiển thị ra màn hình ký tự đó.
- Kết thúc khi nhập vào ký tự q



Ví dụ về InputStream

```
import java.io.*;
public class Nhap2 {
    public static void main(String args[]) {
        while (true) {
            System.out.print("Nhap chuoi: ");
            byte b[] = new byte[100];     // Tao vung dem de nhap chuoi
            try {
                int n = System.in.read(b);              // Nhap n ky tu
                String str = new String(b, 0, n-2);     // Doi byte[] -> String
                if (str.equals("EXIT")) break;      // Kiem tra Dk de thoat
                System.out.println("Chuoi nhan duoc la: " + str);
            }
            catch (IOException ie) {
                System.out.print("Error: "+ie);
            }
        }
    }
}
```

- Nhập 1 chuỗi ký tự từ bàn phím.
- Hiển thị ra màn hình chuỗi ký tự đó.
- Kết thúc khi nhập vào chuỗi "EXIT"



Lớp OutputStream

- `public abstract void write(int ch) throws IOException;`
  - Xuất **ký tự** ch vào thiết bị ngoại vi nối với OutputStream.
- `public void write(byte[] b) throws IOException;`
  - Xuất hết mảng byte b[]
- `public void write(byte[] b, int offset, int length) throws IOException;`
  - Xuất từ mảng b[] từ vị trí *offset* chiều dài là *length*.
- `public void flush() throws IOException;`
- `public void close() throws IOException;`

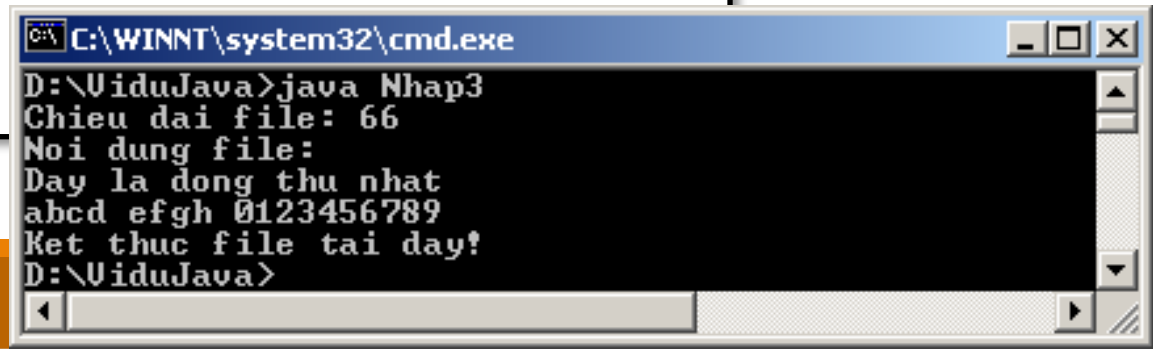
Lớp FileInputStream và FileOutputStream

- **Lớp FileInputStream**
  - Sử dụng để đọc nội dung từ file
  - Thừa kế từ lớp InputStream
  - Có các phương thức như InputStream.
- **Lớp FileOutputStream**
  - Sử dụng để ghi nội dung vào file
  - Thừa kế từ lớp OutputStream
  - Có các phương thức như OutputStream.
- Truy xuất file (đọc/ghi) trực tiếp, không qua vùng đệm.
- Không hiệu quả về mặt hiệu suất (chậm).

Ví dụ về FileInputStream

```
import java.io.*;
public class Nhap3 {
    public static void main(String args[]) {
        try {
            FileInputStream  f1 = new FileInputStream("C:/Test.txt");
            int  len = f1.available();
            System.out.println("Chieu dai file: " + len);
            System.out.println("Noi dung file:");
            for(int i=0; i<len; i++)
                System.out.print((char)f1.read());
            f1.close();
            }
        catch (IOException ie)
            {  System.out.println("Loi khi truy xuat file"); }
    }
}
```

- Đọc nội dung 1 file.
- Hiển thị ra màn hình nội dung đó.



Ví dụ về FileOutputStream

```
import java.io.*;
public class Ghi {
    public static void main(String args[]) {
        try {
            FileOutputStream  f1 = new FileOutputStream("C:/Test1.txt");
            int ch='@'; f1.write(ch);
            byte  b1[] = new byte[10];
            int m=0;
            for(int  i= '0'; i<='9'; i++)      b1[m++]=(byte)i;
            f1.write(b1); m=0; f1.write('\r'); f1.write('\n');
            byte  b2[] = new byte[50];
            for(int  j= 'A'; j<='Z'; j++)     b2[m++]=(byte)j;
            f1.write(b2, 0, m);
            f1.close();
            }
        catch (IOException ie)
            {   System.out.println("Loi khi truy xuat file"); }
    }
}
```

Ghi 1 file với các dạng dữ liệu khác nhau

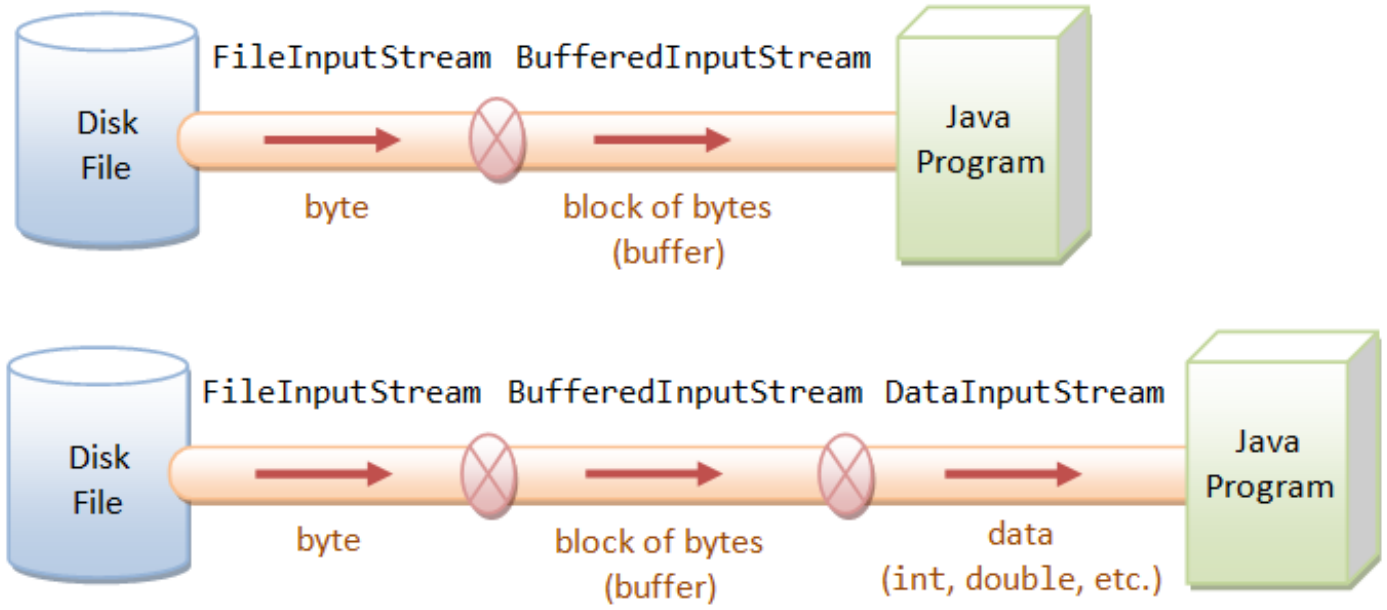
Ví dụ về File Copy

```
import java.io.*;
public class FileCopyNoBuffer {
   public static void main(String[] args) {
      String filedoc= "test-read.jpg";
      String fileghi= "test-write.jpg";
      long batdau, tongtg;
      File f= new File(filedoc);
      System.out.println("Kich thuoc file " + f.length() + " bytes");
      try {
         FileInputStream  in  = new FileInputStream(filedoc);
         FileOutputStream out = new FileOutputStream(fileghi);
         batdau= System.nanoTime();
         int ch;
         while ((ch= in.read()) != -1) {
            out.write(ch);
         }
         tongtg= System.nanoTime() - batdau;
         System.out.println("Thoi gian copy: ” + (tongtg/ 1000000.0) + " ms");
      }
      catch (IOException ex) {
         ex.printStackTrace();
      }
   }
}
```

Đọc từng ký tự từ file test-read.jpg ghi vào file có tên là test-write.jpg

Không có hiệu suất.

Chuyển hướng các dòng nhập xuất



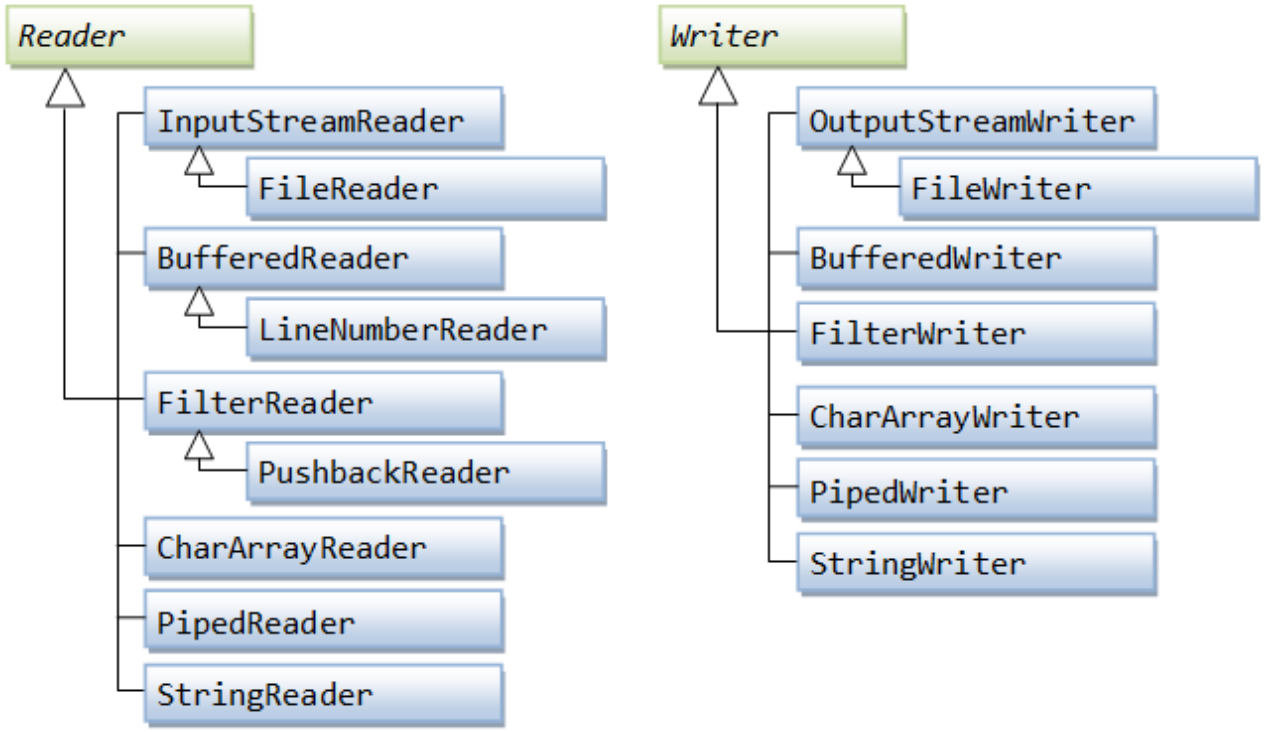
Các dòng nhập xuất chuẩn thường được chuyển thành các dòng khác (thuộc lớp con) với mục đích như lọc dữ liệu, thêm vùng đệm, chuyển định dạng…

Ví dụ về Buffered Stream

```
import java.io.*;
public class FileCopyBufferedStream {
   public static void main(String[] args) {
      String filedoc= "test-read.jpg";
      String fileghi= "test-write.jpg";
      long batdau, tongtg;
      File f= new File(filedoc);
      System.out.println("Kich thuoc file " + f.length() + " bytes");
      try {
         BufferedInputStream  in  = new BufferedInputStream( new
                                        FileInputStream(filedoc));
         BufferedOutputStream out = new BufferedOutputStream( new
                                        FileOutputStream(fileghi));
         batdau= System.nanoTime();
         int ch;
         while ((ch= in.read()) != -1)
            out.write(ch);
         tongtg= System.nanoTime() - batdau;
         System.out.println("Thoi gian copy: " + (tongtg/ 1000000.0) + " ms");
       } catch (IOException ex) {
         ex.printStackTrace();
      }
   }
}
```

So sánh thời gian copy với chương trình FileCopyNoBuffer

Các dòng nhập xuất theo ký tự



- Được dùng cho việc xử lý nhập xuất theo ký tự Unicode (kích thước 16 bits).
- Reader và Writer và các lớp con được sử dụng cho việc chuyển đổi ký tự được lưu theo các bảng mã khác nhau (UTF-8, UTF-16, BIG5, …).

Các dòng nhập xuất theo ký tự

- Lớp Reader
  - `public abstract int read() throws IOException;`
    - Ký tự nhận được có giá trị từ **0 - 65535**
  - `public int read(char[] chars, int offset, int length) throws IOException;`
  - `public int read(char[] chars) throws IOException;`
- Lớp Writer
  - `public void abstract void write(int aChar) throws IOException;`
  - `public void write(char[] chars, int offset, int length) throws IOException;`
  - `public void write(char[] chars) throws IOException;`
- Thông thường sẽ dùng các lớp con của lớp Reader và Writer để truy xuất.

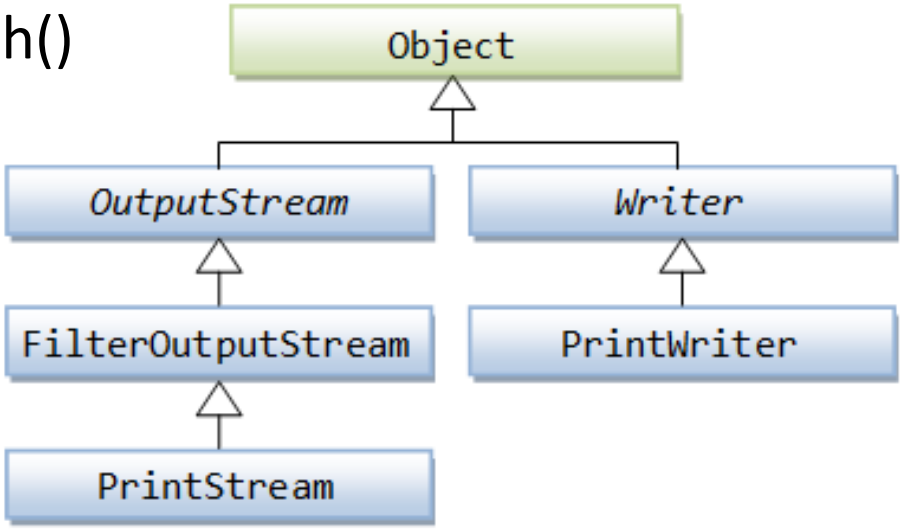
Lớp BufferedReader

- **public String readLine() throws IOException;**
  - Đọc 1 dòng lưu vào chuỗi
  - Kết thúc bởi \n (Unix), \r\n (Windows), \r (Mac)

```
import java.io.*;
public class BufferedFileReader  {
   public static void main(String[] args) {
      String strFilename = "data.txt";
      try {
         BufferedReader in = new BufferedReader(new FileReader(strFilename));
         String inLine;
         while ((inLine = in.readLine()) != null) {
            System.out.println(inLine);
         }
      }
      catch (IOException ex) {
         ex.printStackTrace();
      }
   }
}
```

Lớp PrintStream và PrintWriter

- Lớp PrintStream
  - System.out (màn hình): là đối tượng thuộc lớp PrintStream.
  - Có các hàm print(), println(): hiển thị ra màn hình
- Lớp PrintWriter
  - Có các hàm print(), println()
  - Cần thực thi thêm hàm flush()



Ví dụ về PrintWriter

```
import java.io.*;
public class WriteFilePrintWriter {
   public static void main(String[] args) {
      String strFilename = "data.txt";
      try {
         BufferedReader kb = new BufferedReader(new
                                                InputStreamReader(System.in));
         FileOutputStream f1 = new FileOutputStream (strFilename);
         PrintWriter pw = new PrintWriter(f1);
         System.out.println("Nhap noi dung can ghi vao file");
         System.out.println("Ket thuc voi <CR>.<CR>");
         while(true) {
                  String str = kb.readLine ();
                  if(str.equals(".")) break;
                  pw.println(str); pw.flush();
         }
         f1.close();
         System.out.println("Da ghi file thanh cong !!!");
       }
      catch (IOException ex) {
         ex.printStackTrace();
      }
   }
}
```

Nhập xuất đối tượng

- **Tuần tự hóa đối tượng (Object Serialization)**
  - Là quá trình biểu diễn trạng thái hiện tại của đối tượng thành 1 dòng các bit tuần tự để ghi vào các thiết bị ngoại vi.
  - Lớp tạo ra đối tượng phải được ***implements*** từ ***interface*** java.io.Serializable hoặc java.io.Externalizable
- 2 lớp **ObjectInputStream** và **ObjectOutputStream** để đọc và ghi đối tượng
  - `public final Object readObject() throws IOException, ClassNotFoundException;`
  - `public final void writeObject(Object obj) throws IOException;`

Ví dụ về nhập xuất đối tượng

- Ghi 2 đối tượng String và Date vào file

```
ObjectOutputStream out = new ObjectOutputStream(
            new BufferedOutputStream( new FileOutputStream("object.ser")));
out.writeObject("The current Date and Time is "); // write a String object
out.writeObject(new Date());                      // write a Date object
out.flush();
out.close();
```

- Đọc 2 đối tượng String và Date từ file

```
ObjectInputStream in =  new ObjectInputStream(
           new BufferedInputStream( new FileInputStream("object.ser")));
String str = (String)in.readObject();
Date d = (Date)in.readObject(new Date());  // downcast
in.close();
```

Ví dụ về nhập xuất đối tượng

```
import java.io.Serializable;
class Diem implements Serializable {
        private int        x, y;
        public Diem() { x = y = 0; }
        public Diem(int x, int y) { this.x = x; this.y = y; }
        void hienThi() { System.out.print("(" + x + "," + y + ")"); }
        // Cac ham khac
}
```

```
import java.io.*;
class GhiDoiTuong {
    public static void main(String[] args) {
        Diem a = new Diem(1,1); Diem ds[] = new Diem[4];
        ds[0] = new Diem(); ds[1] = new Diem(2,5);
        ds[2] = new Diem(10,20); ds[3] = new Diem(50,50);
        try {
            ObjectOutputStream f = new ObjectOutputStream(
                        new BufferedOutputStream( new FileOutputStream("data1.ser")));
            f.writeObject(a);
            f.writeObject(ds);
            f.flush(); f.close();
            System.out.println("Da ghi file doi tuong thanh cong");
         }
        catch(IOException e) { e.printStackTrace(); }
    }
}
```

- Ghi đối tượng a (lớp Diem)
- Ghi mảng đối tượng ds[]

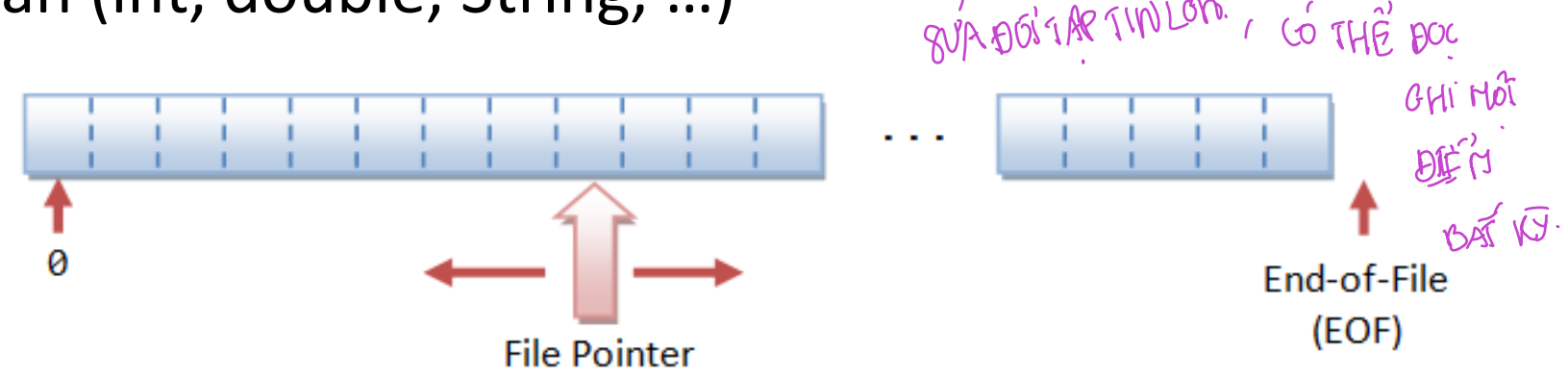
Ví dụ về nhập xuất đối tượng

```
import java.io.*;
class DocDoiTuong {
      public static void main(String[] args) {
         try {
             ObjectInputStream f = new ObjectInputStream(
                          new BufferedInputStream( new
FileInputStream("data1.ser")));
             Diem a;
             Diem ds[];
             a = (Diem)f.readObject();
             ds = (Diem[])f.readObject();
             f.close();
             System.out.println("Noi dung file la");
             a.hienThi();
             for(Diem d: ds)
                  d.hienThi();
         }
         catch(ClassNotFoundException | IOException e) {
              e.printStackTrace();
          }
      }
}
```

- Đọc đối tượng a (lớp Diem)
- Đọc mảng đối tượng ds[]

Tập tin truy cập ngẫu nhiên

- Các dòng nhập xuất chỉ cho phép 1 chiều (đọc – ghi).
- Lớp **RandomAccessFile** là dòng 2 chiều, hỗ trợ cả việc đọc và ghi tại vị trí ngẫu nhiên trong file.
- Khi mở file, con trỏ file sẽ di chuyển đến đầu file (vị trí 0).
- Con trỏ có thể di chuyển đến vị trí bất kỳ trong file để đọc và ghi 1 byte hoặc nhóm các bytes theo các dạng dữ liệu chuẩn (int, double, String, …)

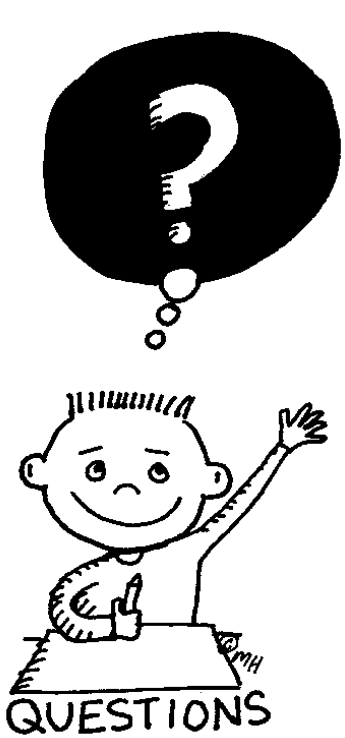


Tập tin truy cập ngẫu nhiên

- Khởi tạo đối tượng lớp RandomAccessFile
  - `RandomAccessFile f1 = new RandomAccessFile("filename", "r");`
  - `RandomAccessFile f2 = new RandomAccessFile("filename", "rw");`
- Một số các phương thức
  - `public void seek(long pos) throws IOException;`
  - `public int skipBytes(int numBytes) throws IOException;`
  - `public long getFilePointer() throws IOException;`
  - `public long length() throws IOException;`
  - `public int readInt() throws IOException;`
  - `public double readDouble() throws IOException;`
  - `public String readLine() throws IOException;`
  - `public void writeInt(int i) throws IOException;`
  - `public void writeDouble(double d) throws IOException;`
  - `public void writeChars(String s) throws IOException;`
  - ...

Tổng kết

- Nhập xuất (vào ra, I/O) trong Java được thực hiện thông qua các dòng (stream).
- Các dòng nhập xuất theo byte thừa kế từ 2 lớp chính là InputStream và OutputStream, hỗ trợ việc truy xuất theo từng byte dữ liệu thô hoặc nhị phân.
- Các dòng nhập xuất theo ký tự thừa kế từ 2 lớp chính là Reader và Writer, hỗ trợ việc truy xuất theo từng ký tự được lưu dưới dạng text của nhiều bảng mã khác nhau.
- Đối tượng cũng có thể được ghi và đọc tuần tự từ các thiết bị ngoại vi (file).
- Có nhiều lớp khác nhau hỗ trợ việc truy xuất tập tin.



Question?

CT176 – LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG