LẬP TRÌNH ỨNG DỤNG JAVA

JAVA IO

Nguyễn Hoàng Anh – nhanh@fit.hcmuns.edu.vn
Nguyễn Đức Huy – ndhuy@fit.hcmus.edu.vn

Nội dung

- **File**
- **Stream**
  - **Character Stream**
  - **Byte Stream**
- **Zip**
  - **ZipOutputStream**
  - **ZipInputStream**

Java IO

- Data
  - Memory
  - Disk
  - Network
- Các lớp chính dùng để xử lý IO thuộc gói java.io

File

- Đối tượng File có thể được xem như là
  - Môt tập tin
  - Một thư mục

```
File f1=new File("test.txt");
File f2=new File("D:\\GiangDay\\Java");
```

File

- File.separator
- File.separatorChar
- File.pathSeparator
- File.pathSeparatorChar
- File.createTempFile (String prefix, String suffix)
- File.createTempFile(String prefix, String suffix, File dir)
- File.listRoots()

File

```
package javaio;
import java.io.*;
public class FileProcessing {

    public static void createTempFile() {
        File tempFile = null;
        try {
            tempFile = File.createTempFile("MyFile.txt", ".tmp");
            System.out.print("Created temporary file with name ");
            System.out.println(tempFile.getAbsolutePath());
        } catch (IOException ex) {

            System.err.println("Cannot create temp file: " + ex.getMessage());
        } finally {
            if (tempFile != null) {
            }
        }
    }
    public static void main(String[] args) throws IOException {
        FileProcessing.createTempFile();
    }
}
```

```
Created temporary file with name
C:\Users\NHAnh\AppData\Local\Temp\MyFile.txt6059735001335123925.tmp
```

File

```
public class Main {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws IOException {
        // TODO code application logic here
        File[] fs=File.listRoots();
        for (int i=0; i<fs.length; i++){
            System.out.println(fs[i].getPath());
        }
    }
}
```

```
init:
deps-jar:
Compiling 1 source file to D:\GiangDay\2009\JAVA\Demo\JavaIOSample\build\classes
compile:
run:
C:\
D:\
E:\
F:\
G:\
BUILD SUCCESSFUL (total time: 0 seconds)
```

File

- **isFile ()**
- **isDirectory ()**
- **isHidden ()**
- **canRead ()**
- **canWrite ()**
- **canExecute ()**
- **exists ()**

File

- **createNewFile()**
- **delete()**
- **deleteOnExit()**
- **mkdir()**
- **mkdirs()**
- **renameTo (File dest)**
- **System.getProperty(“user.dir”)**

File

```
package javaiosample;

import java.io.File;
import java.io.IOException;
public class Main {
    public static void main(String[] args) throws IOException {
        String path = System.getProperty("user.dir");
        File folder = new File(path + "doc/java/io");
        folder.mkdirs();

    }
}
```

File

```
package javaiosample;
import java.io.File;
import java.io.IOException;

public class Main {

    public static void main(String[] args) throws IOException {
        String path = System.getProperty("user.dir");
        File file = new File(path + "abc.txt");
        file.createNewFile();
    }
}
```

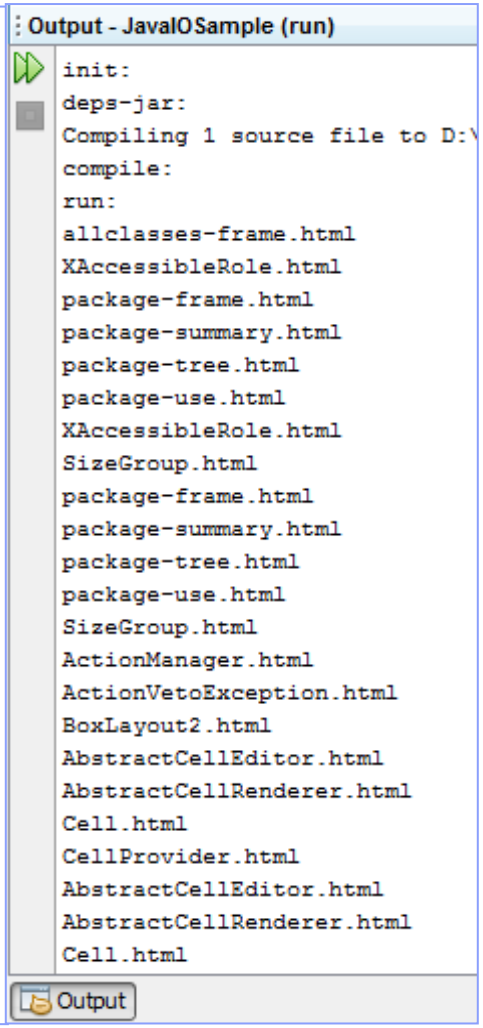
- **setExecutable (boolean exe)**
- **setLasModified(long time)**
- **setReadonly()**
- **setReadable(boolean b)**
- **setWritable(boolean b)**
- **toURI ()**

File

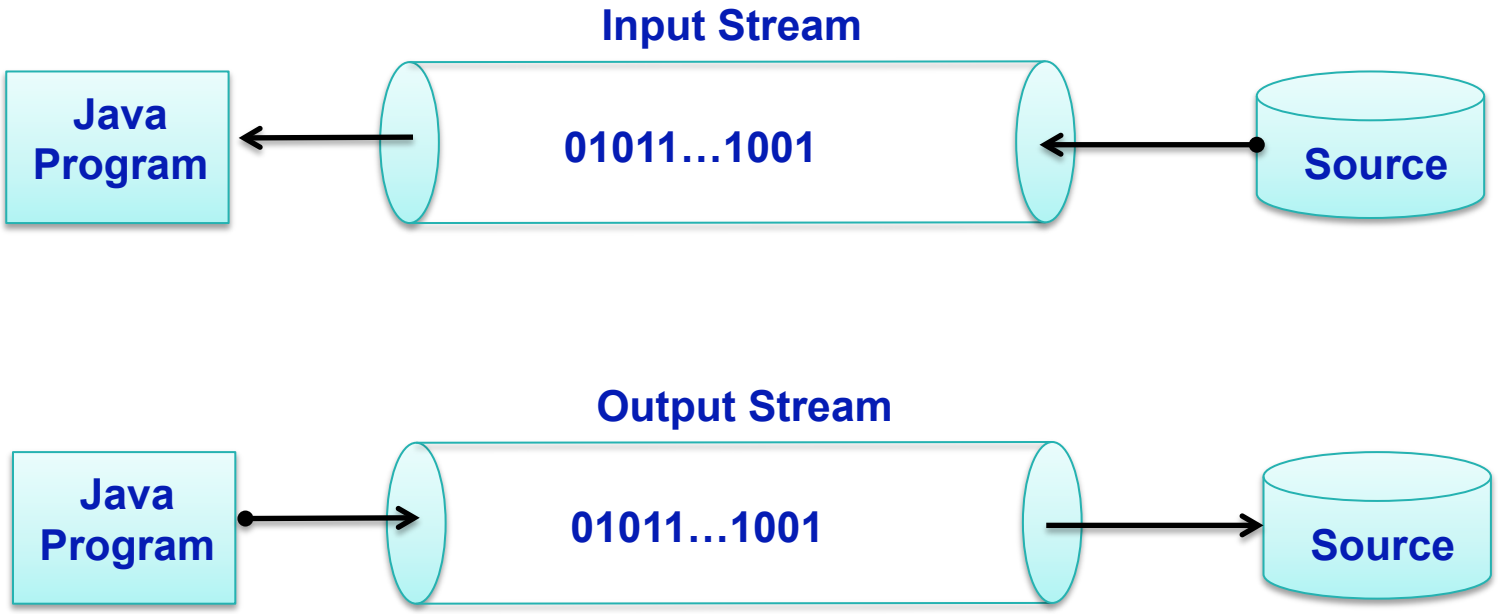
- **getName()**
- **getParentFile()**
- **getPath() , toString ()**
- **length() , lastModified()**
- **list() , list(FilenameFilter filter)**
- **listFiles()**
- **listFies(FileFilter filter)**

File

```
public class Main {

    private static void Files(ArrayList<File> af, File folder) {
        for (File file : folder.listFiles()) {
            if (file.isFile()) {
                af.add(file);
            }
            if (file.isDirectory()) {
                Files(af, file);
            }
        }
    }
    public static void main(String[] args) throws IOException {
       ArrayList<File> al=new ArrayList<File>();
       File folder=new File("doc");
       Files(al,folder);
       for(File file :al){
           System.out.println(file.getName());
       }
    }
}
```



Stream



Java sử dụng Stream để Read và Write data

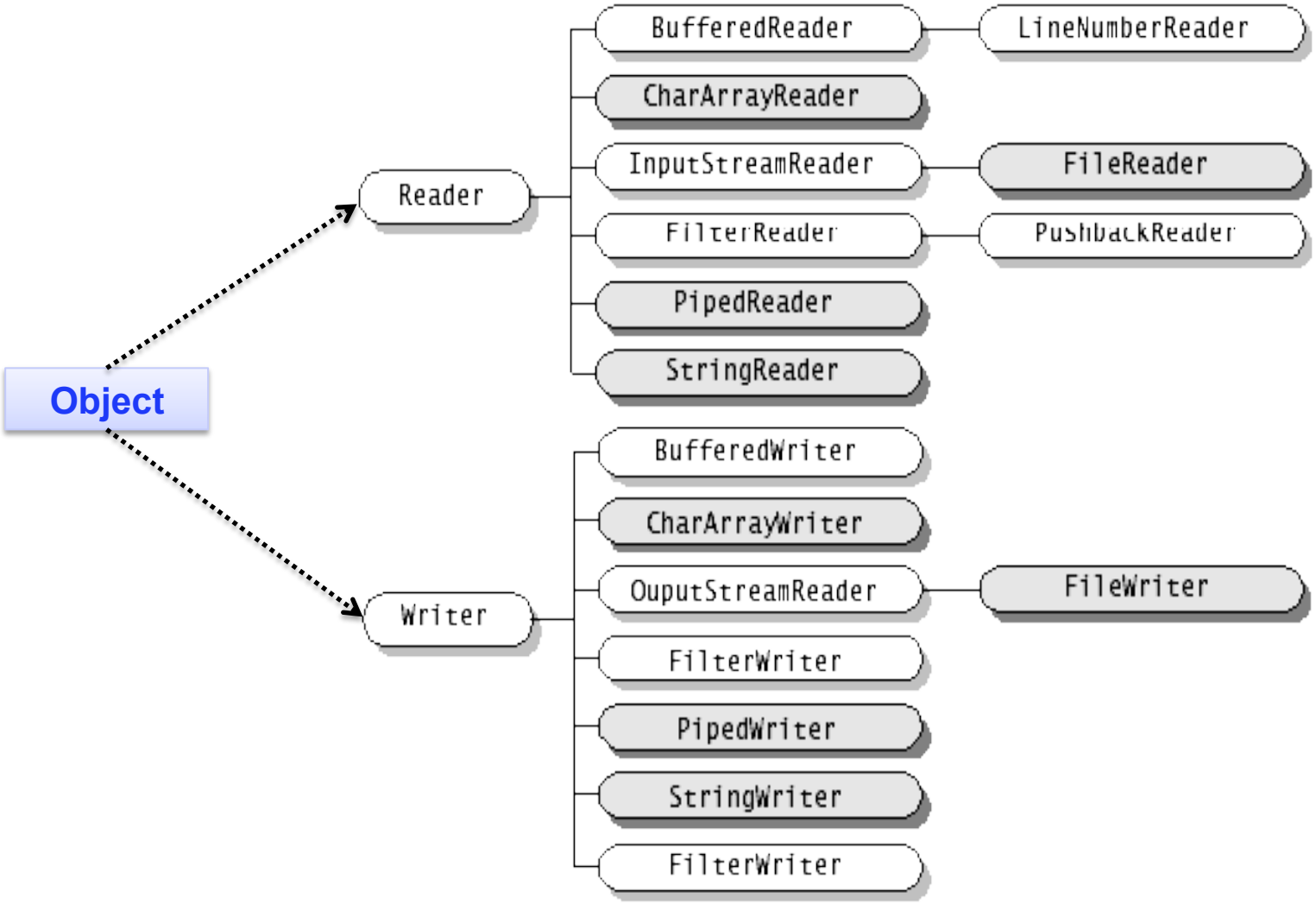
Stream

- **Stream** là một dãy tuần tự các byte có chiều dài không xác định.
- **Input Stream** là các stream thực hiện việc di chuyển đưa dãy các byte vào trong chương trình Java từ một nguồn bên ngoài.
- **Output Stream** đưa dãy các byte từ chương trình Java đến các nơi bên ngoài

Stream

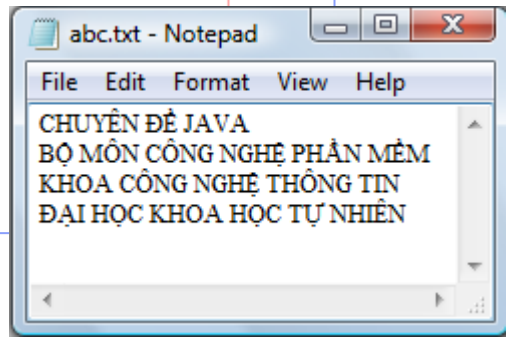
- Package **java.io** bao gồm các **Stream** Class:
  - ***Character Streams:***
    - Được sử dụng cho **16-bit characters**
    - Sử dụng **Read** & **Write classes**
  - ***Byte Streams*** :
    - Được sử dụng cho **8-bit bytes**
    - Sử dụng **InputStream** & **OutputStream classes**

Character Stream



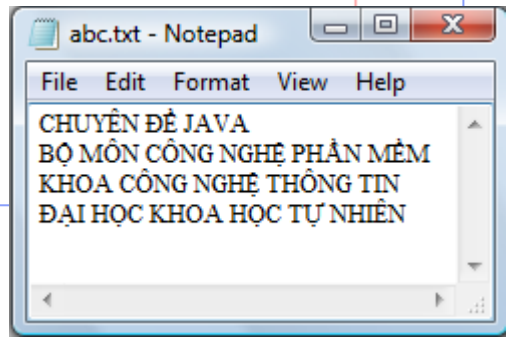
Character Stream

```
public static void main(String[] args) throws IOException {
    FileOutputStream fos = new FileOutputStream("abc.txt");
    BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(fos, "UTF-8"));
    String[] strs = new String[]{"CHUYÊN ĐỀ JAVA",
                                 "BỘ MÔN CÔNG NGHỆ PHẦN MỀM",
                                 "KHOA CÔNG NGHỆ THÔNG TIN",
                                 "ĐẠI HỌC KHOA HỌC TỰ NHIÊN"
                                 };
    for (String str : strs) {
        bw.write(str);
        bw.newLine();
    }
    bw.close();
}
```

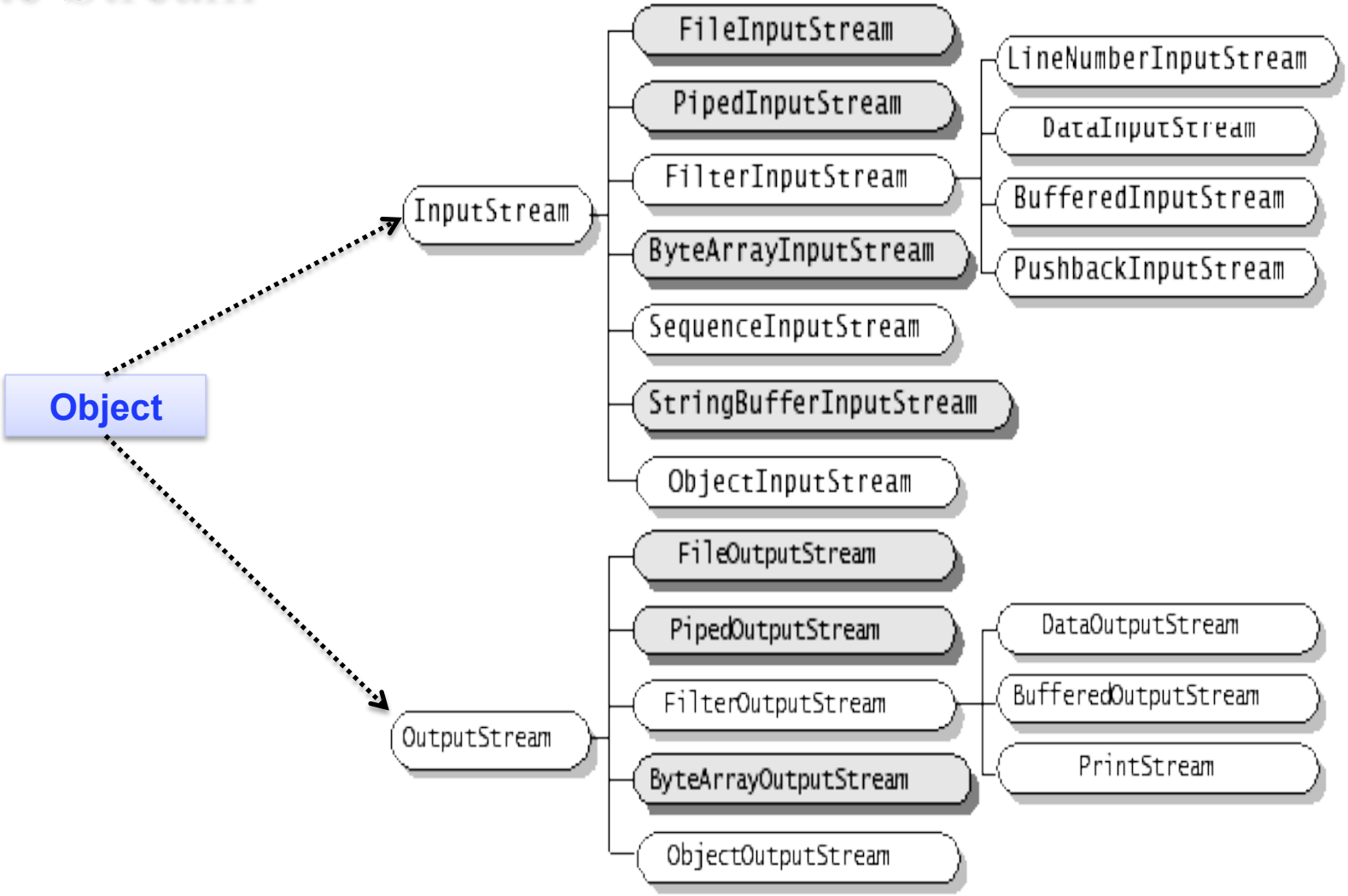


Character Stream

```
public static void main(String[] args) throws IOException {
    FileInputStream fis = new FileInputStream("abc.txt");
    BufferedReader br = new BufferedReader(new InputStreamReader(fis, "UTF-8"));
    String str = null;
    do {
        str = br.readLine();
        System.out.println(str);
    } while (str != null);

}
```



Byte Stream


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

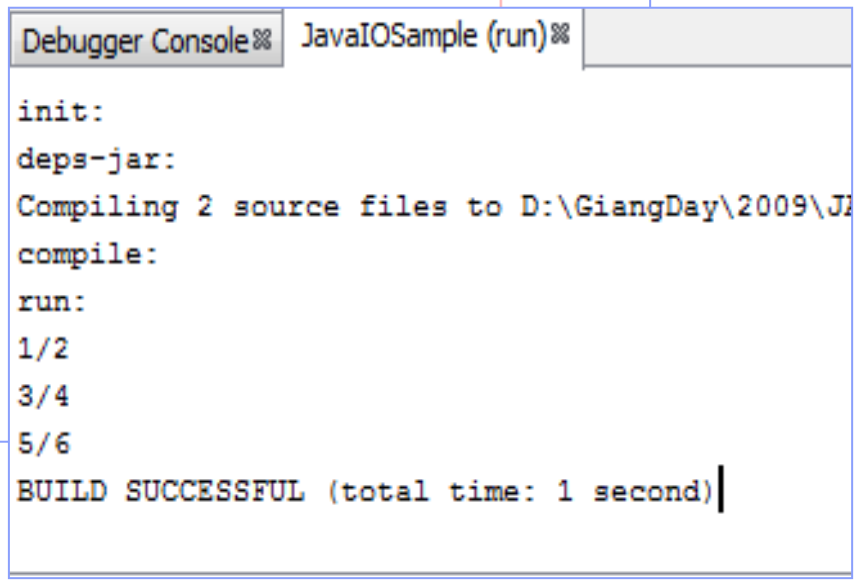
Byte Stream

```
public static void main(String[] args) throws IOException, ClassNotFoundException {
    ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("abc.obj"));
    PhanSo[] arr = new PhanSo[3];
    arr[0] = new PhanSo(1, 2);
    arr[1] = new PhanSo(3, 4);
    arr[2] = new PhanSo(5, 6);
    oos.writeObject(arr.length);
    for (PhanSo ps : arr) {
        oos.writeObject(ps);
    }
    oos.close();
}
```

```
public class PhanSo implements Serializable {
    private int _tuSo;
    private int _mauSo;
    public PhanSo() {...}
    public PhanSo(int tuSo, int mauSo) {...}
    public int getTuSo() {...}
    public void setTuSo(int tuSo) {...}
    public int getMauSo() {...}
    public void setMauSo(int mauSo) {...}
    public void xuat() {...}
}
```

Byte Stream

```
public static void main(String[] args) throws IOException, ClassNotFoundException {
   ObjectInputStream ois = new ObjectInputStream(new FileInputStream("abc.obj"));
    Object obj = null;
    int n = ((Integer)ois.readObject()).intValue();
    for (int i = 0; i < n; i++) {
        obj = ois.readObject();
        ((PhanSo) obj).xuat();
    }
    ois.close();
}
```



Java IO

- Reader và InputStream định nghĩa các API tương tự nhau nhưng cho 2 kiểu dữ liệu khác nhau

```
int read()
int read(char cbuf[])
int read(char cbuf[], int offset, int length)
```

Reader

```
int read()
int read(byte cbuf[])
int read(byte cbuf[], int offset, int length)
```

InputStream

Java IO

- Writer và OutputStream định nghĩa các API tương tự nhau nhưng cho 2 kiểu dữ liệu khác nhau

```
int write()
int write(char cbuf[])
int write(char cbuf[], int offset, int length)
```

Writer

```
int write()
int write(byte cbuf[])
int write(byte cbuf[], int offset, int length)
```

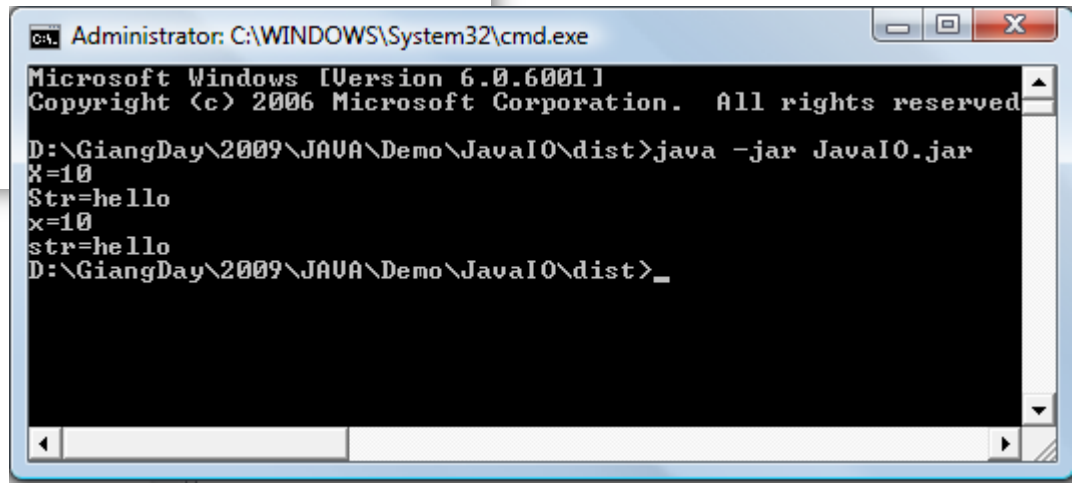
OutputStream

Console

- Output **: System.out**
  - Đọc từ keyboard
- Input : **System.in**
  - Xuất ra màn hình console
- Error : **System.err**
  - Xuất lỗi ra màn hình console
- Console: **System.console()**

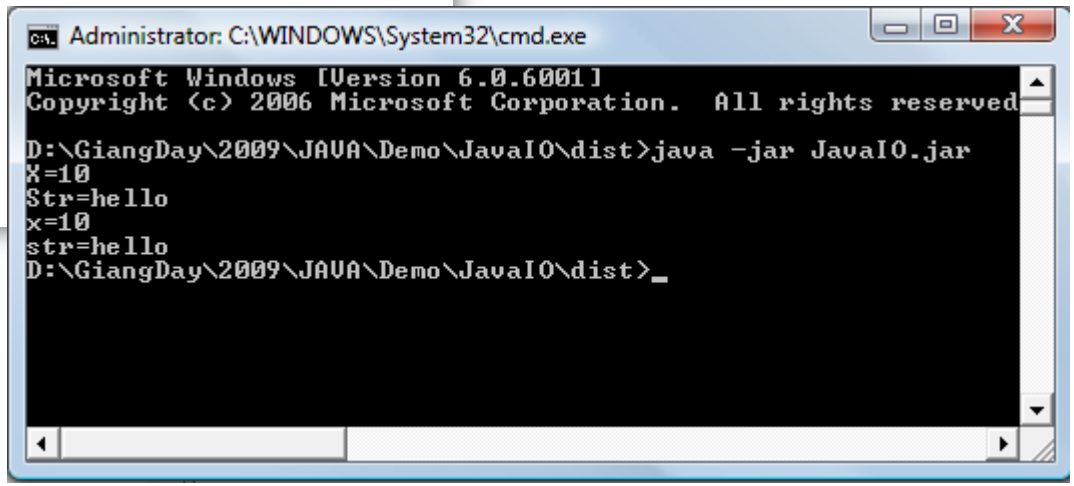
Console

```
public class ConsoleIODemo {

    public static void test1() {
        System.out.print("X=");
        int x = Integer.parseInt(System.console().readLine());
        System.out.print("Str=");
        String str = System.console().readLine();
        System.out.println("x=" + x);
        System.out.print("str=" + str);

    }
}
```

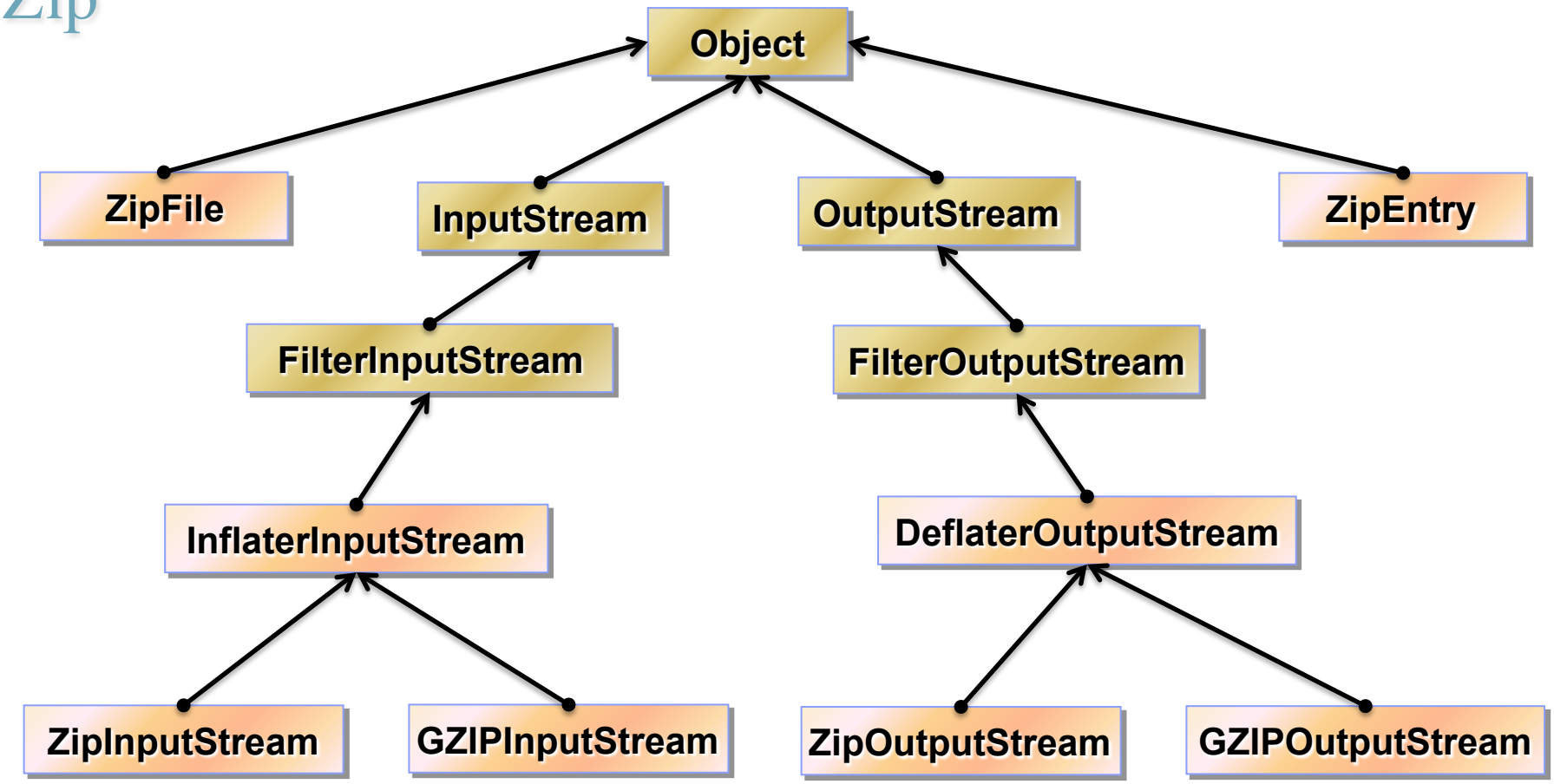


Console

```
public class ConsoleIODemo {

    public static void test2(){
        InputStreamReader inp=new InputStreamReader(System.in);
        BufferedReader br=new BufferedReader(inp);
        int x;
        try {
            x = Integer.parseInt(br.readLine());
            System.out.print("Str=");
            String str = br.readLine();
            System.out.println("x=" + x);
            System.out.print("str=" + str);
        } catch (IOException ex) {
           ex.printStackTrace();
        }
    }
}
```



Zip


Object
ZipFile
InputStream
OutputStream
ZipEntry
FilterInputStream
FilterOutputStream
InflaterInputStream
DeflaterOutputStream
ZipInputStream
GZIPInputStream
ZipOutputStream
GZIPOutputStream

Zip

- InflaterInputStream:
  - Giải nén dữ liệu dạng zip hoặc gzip
- DeflaterOutputStream
  - Nén dữ liệu dạng zip hoặc gzip
- ZipFile
  - Đọc các ZipEntry từ một file zip
- ZipEntry
  - Thể hiện một phần tử của file Zip

Zip

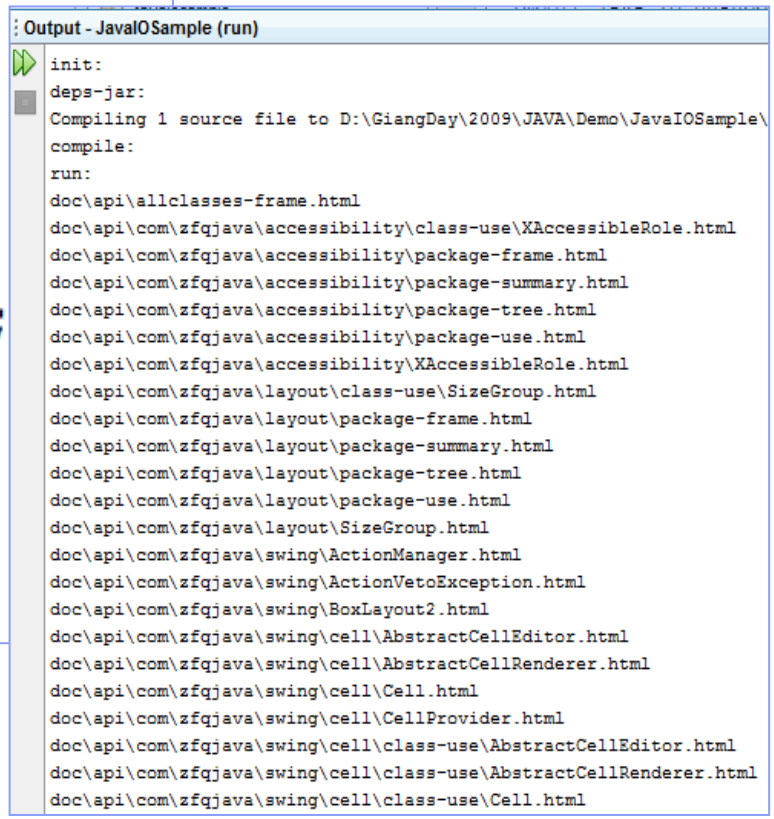
- **ZipInputStream**
  - Đọc các file trong file zip, giải nén file zip
- **GZIPInputStream**
  - Đọc các file trong file gzip, giải nén file gzip
- **ZipOutputStream**
  - Ghi file theo định dạng zip
- **GZIPOutputStream**
  - Ghi file theo định dạng gzip

Zip

- **ZipEntry**
  - **getName()**
  - **getSize()**
- **ZipFile**
  - **getEntry(String name)**
  - **entries()**
  - **getName ()**
  - **getSize ()**

Zip

```
public static void main(String[] args) throws IOException  {
    ZipFile zf = new ZipFile("doc.zip");
     Enumeration entries = zf.entries();
     while(entries.hasMoreElements()){
         ZipEntry entry=(ZipEntry)entries.nextElement();
         System.out.println(entry.getName());
     }
}
```



ZipOutputStream

- **ZipOutputStream (OutputStream out)**
- **pushNextEntry(ZipEntry entry)**
- **Write (byte[] b)**
- **Write(byte[] b, int off, int len)**
- **flush ()**
- **close(), closeEntry()**
- **setComment()**

ZipOutputStream

```
public static void ZipFile(File file) throws FileNotFoundException, IOException {
    byte[] data = new byte[1024];
    ZipOutputStream zos = new ZipOutputStream(
                                    new FileOutputStream(file.getName() + ".zip")
                                                );
    FileInputStream fis = new FileInputStream(file);
    zos.putNextEntry(new ZipEntry(file.getPath()));
    int count;
    while ((count = fis.read(data, 0, 1024)) != -1) {
        zos.write(data, 0, count);
    }
    zos.closeEntry();
    zos.flush();
    zos.close();
}
```

ZipOutputStream

```
private static void Files(ArrayList<File> af, File folder) {
    for (File file : folder.listFiles()) {
        if (file.isFile()) {
            af.add(file);
        }
        if (file.isDirectory()) {
            Files(af, file);
        }
    }
}
```

ZipOutputStream

```
public static void ZipFolder(File folder) throws FileNotFoundException, IOException {
    ArrayList<File> af = new ArrayList<File>();
    MyZip.Files(af, folder);
    ZipOutputStream zos = new ZipOutputStream(new FileOutputStream(folder.getName() + ".zip"));
    FileInputStream fis = null;
    byte[] data = new byte[1024];
    for (int i = 0; i < af.size(); i++) {
        File file = af.get(i);
        fis = new FileInputStream(file);
        zos.putNextEntry(new ZipEntry(file.getPath()));
        int count;
        while ((count = fis.read(data, 0, 1024)) != -1) {
            zos.write(data, 0, count);
        }
        zos.closeEntry();
        fis.close();

    }
    zos.flush();
    zos.close();
}
```

ZipInputStream

- **ZipInputStream (InputStream out)**
- **getNextEntry()**
- **read (byte[] b)**
- **read(byte[] b, int off, int len)**
- **flush ()**
- **close(), closeEntry()**
- **setComment()**

ZipInputStream

```
public static void UnZip(File file) throws FileNotFoundException, IOException {
    ZipInputStream zis = new ZipInputStream(new FileInputStream(file));
    ZipEntry entry;
    while ((entry = zis.getNextEntry()) != null) {
        int count;
        byte data[] = new byte[1024];
        String path = System.getProperty("user.dir") + File.separator + entry.getName();
        String[] s = path.split("\\\\");
        String dirs = "";
        for (int i = 0; i < s.length - 1; i++) {
            dirs = dirs + File.separator + s[i];
        }
        new File(dirs).mkdirs();
        File fout = new File(path);
        fout.createNewFile();
        FileOutputStream fos = new FileOutputStream(fout);
        while ((count = zis.read(data, 0, 1024)) != -1) {
            fos.write(data, 0, count);
        }
        fos.close();
    }
    zis.close();
}
```

ZipInputStream

```
public static void AppendZipFile(File zip, File fileAppend) throws FileNotFoundException, IOException {
    ArrayList<File> af = new ArrayList<File>();
    MyZip.Files(af, fileAppend);
    FileInputStream fis = null;
    byte[] data = new byte[1024];
    ZipFile zf = new ZipFile(zip);
    Enumeration entries = zf.entries();
    ZipOutputStream zos = new ZipOutputStream(new FileOutputStream(zip.getName()+"_1.zip"));
    while (entries.hasMoreElements()) {
        ZipEntry entry = (ZipEntry) entries.nextElement();
        InputStream is = zf.getInputStream(entry);
        zos.putNextEntry(entry);
        int count;
        while ((count = is.read(data, 0, 1024)) != -1) {
            zos.write(data, 0, count);
        }
        zos.closeEntry();
        is.close();
    }
```

ZipInputStream

```
    for (int i = 0; i < af.size(); i++) {
        File file = af.get(i);
        fis = new FileInputStream(file);
        zos.putNextEntry(new ZipEntry(file.getPath()));
        int count;
        while ((count = fis.read(data, 0, 1024)) != -1) {
            zos.write(data, 0, count);
        }
        zos.closeEntry();
        fis.close();

    }
    zos.flush();
    zos.close();
}
```

Tham khảo

- **http://java.sun.com/j2se/1.4.2/docs/api/java/io/package-summary.html**
- **http://java.sun.com/j2se/1.3/docs/api/java/util/zip/package-summary.html**