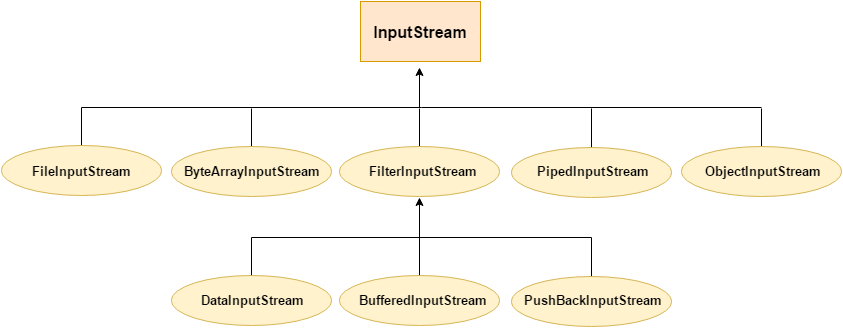
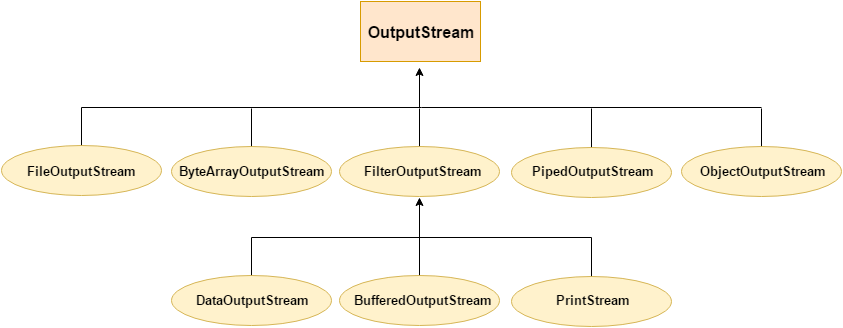
Kiến trúc InputStream:





Input Stream OutputStream là abstract class, không tạo đối tượng được, các stream thực hiện việc di chuyển đưa dãy các byte vào trong chương trình Java từ một nguồn bên ngoài

Các lớp kế thừa từ Stream xử lí các loại dữ liệu khác nhau:

* File(Input/Output)Stream: đọc file => chú ý.
* Filter(Input/Output)Stream: đọc file => không cần để ý.
* ByteArray(Input/Output)Stream: đọc mảng byte như là input stream.
* Piped(Input/Output)Stream: đọc dữ liệu từ một luồng khác, kiểu như ống nước kết nối hai đầu dữ liệu. <https://shareprogramming.net/pipedinputstream-va-pipedoutputstream-trong-java/>
* Object(Input/Output)Stream: hỗ trợ việc đọc/ghi các đối tượng.

<https://openplanning.net/13397/java-objectinputstream>

* Một số hàm chính trong các Input Stream:

| int | [**available**](https://docs.oracle.com/javase/7/docs/api/java/io/InputStream.html#available())()  trả về số byte có thể đọc được |
| --- | --- |
| void | [**close**](https://docs.oracle.com/javase/7/docs/api/java/io/InputStream.html#close())()  đóng luồn. P/s: lưu ý tạo luồng đọc thì phải đóng lại. |
| void | [**mark**](https://docs.oracle.com/javase/7/docs/api/java/io/InputStream.html#mark(int))(int readlimit)=> không cần để ý  Marks the current position in this input stream. |
| boolean | [**markSupported**](https://docs.oracle.com/javase/7/docs/api/java/io/InputStream.html#markSupported())()=> không cần để ý  Tests if this input stream supports the mark and reset methods. |
| abstract int | [**read**](https://docs.oracle.com/javase/7/docs/api/java/io/InputStream.html#read())()  đọc byte kế tiếp trong luồng.  Đối với các lớp nhứ File InputStream sẽ đọc từng kí tự trong file. |
| int | [**read**](https://docs.oracle.com/javase/7/docs/api/java/io/InputStream.html#read(byte%5B%5D))(byte[] b)  giống hàm ở trên chỉ khác là đọc xong thì lưu vào b và trả về số byte đã đọc |
| int | [**read**](https://docs.oracle.com/javase/7/docs/api/java/io/InputStream.html#read(byte%5B%5D,%20int,%20int))(byte[] b, int off, int len)  Tương tự như ở trên chỉ khác là có thêm vị trí đọc ở đâu (tại “off”) và độ dài là bao nhiêu (len). |
| void | [**reset**](https://docs.oracle.com/javase/7/docs/api/java/io/InputStream.html#reset())()=> không xài nhiều  Trở về vị trị đã được hàm mark đánh dấu lần cuối |
| long | [**skip**](https://docs.oracle.com/javase/7/docs/api/java/io/InputStream.html#skip(long))(long n)  bỏ qua n byte trong luồng |

* Một số hàm chính trong các Output Stream:

| void | [**close**](https://docs.oracle.com/javase/7/docs/api/java/io/OutputStream.html#close())()  đóng luồn |
| --- | --- |
| void | [**flush**](https://docs.oracle.com/javase/7/docs/api/java/io/OutputStream.html#flush())()  làm sạch luồn, đẩy tất cả các byte còn trong luồng ra khỏi. |
| void | [**write**](https://docs.oracle.com/javase/7/docs/api/java/io/OutputStream.html#write(byte%5B%5D))(byte[] b)  ghi một chuỗi dữ liệu từ mảng b.  P/s: đối với FileOutputStream sẽ ghi ra được chuôi. |
| void | [**write**](https://docs.oracle.com/javase/7/docs/api/java/io/OutputStream.html#write(byte%5B%5D,%20int,%20int))(byte[] b, int off, int len)  giống trên mà ghi ở vị trí nào trong File với độ dài bao nhiêu |
| abstract void | [**write**](https://docs.oracle.com/javase/7/docs/api/java/io/OutputStream.html#write(int))(int b)  ghi một kí tự byte có mã kí tự là b.w |

* File(Input/Output)Stream:

**package** org.o7planning.fileinputstream.ex;

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.IOException;

**import** java.net.MalformedURLException;

**public** **class** **FileInputStreamEx1** {

**public** **static** **void** **main**(String[] args) **throws** MalformedURLException, IOException {

// Windows Path: C:/Data/test/utf8-file-without-bom.txt

String path = "/Volumes/Data/test/utf8-file-without-bom.txt";

File file = **new** **File**(path);

FileInputStream fis = **new** **FileInputStream**(file);

int code;

**while**((code = fis.read()) != -1) {

char ch = (char) code;

System.out.println(code + " " + ch);

}

fis.close();

}

}

Output:

74 J

80 P

230 æ

151

165 ¥

230 æ

156

172 ¬

45 -

229 å

133

171 «

230 æ

180 ´

178 ²

* ByteArray(Input/Output)Stream:
* Piped(Input/Output)Stream:=> Không cần kĩ lắm

import java.io.IOException;

import java.io.PipedInputStream;

import java.io.PipedOutputStream;

**class** PipeExample {

public static **void** main(String[] args) throws IOException {

final PipedOutputStream output = **new** PipedOutputStream();

final PipedInputStream input = **new** PipedInputStream(output);

Thread thread1 = **new** Thread(() -> {

**try** {

output.write("Hello world, pipe!".getBytes());

} **catch** (IOException e) {

e.printStackTrace();

} finally {

**try** {

output.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

});

Thread thread2 = **new** Thread(() -> {

**try** {

System.out.println("I'm thread 2");

int data = input.read();

**while**(data != -1){

System.out.print((char)data);

data = input.read();

}

} **catch** (IOException e) {

e.printStackTrace();

} finally {

**try** {

input.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

});

thread1.start();

thread2.start();

}

}

*Output:*

*I’m thread 2*

*Hello world, pipe!*

* Object(Input/Output)Stream:

Employee.java

**package** org.o7planning.beans;

**import** java.io.Serializable;

**public** **class** **Employee** **implements** **Serializable** {

**private** **static** **final** long serialVersionUID = 1L;

**private** String fullName;

**private** float salary;

**public** **Employee**(String fullName, float salary) {

this.fullName = fullName;

this.salary = salary;

}

**public** String **getFullName**() {

**return** fullName;

}

**public** **void** **setFullName**(String firstName) {

this.fullName = firstName;

}

**public** float **getSalary**() {

**return** salary;

}

**public** **void** **setSalary**(float lastName) {

this.salary = lastName;

}

}

WriteEmployeeDataEx.java

**package** org.o7planning.objectinputstream.ex;

**import** java.io.File;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import** java.io.ObjectOutputStream;

**import** java.io.OutputStream;

**import** java.util.Date;

**import** org.o7planning.beans.Employee;

**public** **class** **WriteEmployeeDataEx** {

// Windows: C:/Data/test/employees.data

**private** **static** String file\_path = "/Volumes/Data/test/employees.data";

**public** **static** **void** **main**(String[] args) **throws** IOException {

File outFile = **new** **File**(file\_path);

outFile.getParentFile().mkdirs();

Employee e1 = **new** **Employee**("Tom", 1000f);

Employee e2 = **new** **Employee**("Jerry", 2000f);

Employee e3 = **new** **Employee**("Donald", 1200f);

Employee[] employees = **new** **Employee**[] { e1, e2, e3 };

OutputStream os = **new** **FileOutputStream**(outFile);

ObjectOutputStream oos = **new** **ObjectOutputStream**(os);

System.out.println("Writing file: " + outFile.getAbsolutePath());

oos.writeObject(**new** **Date**());

oos.writeUTF("Employee data"); // Some informations.

oos.writeInt(employees.length); // Number of Employees

**for** (Employee e : employees) {

oos.writeObject(e);

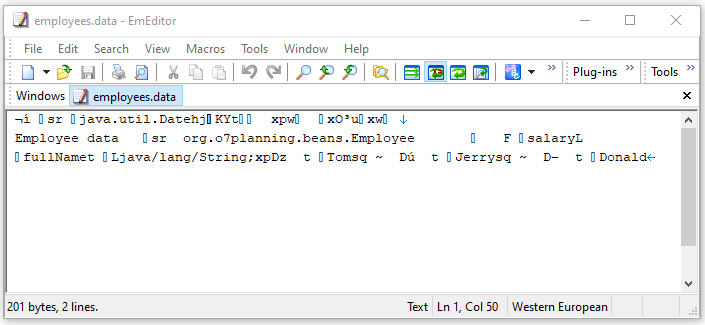
}

oos.close();

System.out.println("Finished!");

}

}



ReadEmployeeDataEx.java

**package** org.o7planning.objectinputstream.ex;

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.IOException;

**import** java.io.InputStream;

**import** java.io.ObjectInputStream;

**import** java.util.Date;

**import** org.o7planning.beans.Employee;

**public** **class** **ReadEmployeeDataEx** {

// Windows: C:/Data/test/employees.data

**private** **static** String file\_path = "/Volumes/Data/test/employees.data";

**public** **static** **void** **main**(String[] args) **throws** IOException, ClassNotFoundException {

File inFile = **new** **File**(file\_path);

InputStream is = **new** **FileInputStream**(inFile);

ObjectInputStream ois = **new** **ObjectInputStream**(is);

System.out.println("Reading file: " + inFile.getAbsolutePath());

System.out.println();

Date date = (Date) ois.readObject();

String info = ois.readUTF();

System.out.println(date);

System.out.println(info);

System.out.println();

int employeeCount = ois.readInt();

**for**(int i=0; i< employeeCount; i++) {

Employee e = (Employee) ois.readObject();

System.out.println("Employee Name: " + e.getFullName() +" / Salary: " + e.getSalary());

}

ois.close();

}

}

Output:

Reading file: /Volumes/Data/test/employees.data

Sat Mar 20 18:54:24 KGT 2021

Employee data

Employee Name: Tom / Salary: 1000.0

Employee Name: Jerry / Salary: 2000.0

Employee Name: Donald / Salary: 1200.0

GameSetting.java

**package** org.o7planning.beans;

**import** java.io.IOException;

**import** java.io.ObjectInputStream;

**public** **class** **GameSetting** **implements** **java**.io.Serializable {

**private** **static** **final** long serialVersionUID = 1L;

**private** int sound;

**private** int bightness;

**private** String difficultyLevel;

**private** String userNote;

**public** **GameSetting**(int sound, int bightness, String difficultyLevel, String userNote) {

this.sound = sound;

this.bightness = bightness;

this.difficultyLevel = difficultyLevel;

this.userNote = userNote;

}

**public** int **getSound**() {

**return** sound;

}

**public** int **getBightness**() {

**return** bightness;

}

**public** String **getDifficultyLevel**() {

**return** difficultyLevel;

}

**public** String **getUserNote**() {

**return** userNote;

}

// Do not change name and parameter of this method.

**private** **void** **readObject**(ObjectInputStream in) **throws** IOException, ClassNotFoundException {

ObjectInputStream.GetField fields = in.readFields();

this.sound = fields.get("sound", 50);

this.bightness = fields.get("bightness", 50);

// Edit fields

this.difficultyLevel = (String) fields.get("difficultyLevel", "Easy"); // Default

**if** (this.difficultyLevel == null) {

this.difficultyLevel = "Easy";

}

this.userNote = (String) fields.get("userNote", "Have fun!"); // Default

**if** (this.userNote == null) {

this.userNote = "Have fun!";

}

}

}

ObjectInputStream\_readFields.java

**package** org.o7planning.objectinputstream.ex;

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import** java.io.InputStream;

**import** java.io.ObjectInputStream;

**import** java.io.ObjectOutputStream;

**import** java.io.OutputStream;

**import** java.util.Date;

**import** org.o7planning.beans.GameSetting;

**public** **class** **ObjectInputStream\_readFields** {

// Windows: C:/Data/test/game\_setting.data

**private** **static** String file\_path = "/Volumes/Data/test/game\_setting.data";

**public** **static** **void** **main**(String[] args) **throws** IOException, ClassNotFoundException {

GameSetting setting = **new** **GameSetting**(10, 80, null, null);

writeGameSetting(setting);

readGameSetting();

}

**private** **static** **void** **writeGameSetting**(GameSetting setting) **throws** IOException {

File file = **new** **File**(file\_path);

file.getParentFile().mkdirs();

OutputStream os = **new** **FileOutputStream**(file);

ObjectOutputStream oos = **new** **ObjectOutputStream**(os);

// Write a String

oos.writeUTF("Game Settings, Save at " + **new** **Date**());

// Write Object

oos.writeObject(setting);

oos.close();

}

**private** **static** **void** **readGameSetting**() **throws** IOException, ClassNotFoundException {

File file = **new** **File**(file\_path);

file.getParentFile().mkdirs();

InputStream is = **new** **FileInputStream**(file);

ObjectInputStream ois = **new** **ObjectInputStream**(is);

// Read a String

String info = ois.readUTF();

// Read fields

GameSetting setting = (GameSetting) ois.readObject();

System.out.println("sound: " + setting.getSound());

System.out.println("bightness: " + setting.getBightness());

System.out.println("difficultyLevel: " + setting.getDifficultyLevel());

System.out.println("userNote: " + setting.getUserNote()); // null.

ois.close();

}

}

Output:

sound: 10

bightness: 80

difficultyLevel: Easy

userNote: Have fun!

ObjectInputStream\_readUnshared​.java

**package** org.o7planning.objectinputstream.ex;

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import** java.io.InputStream;

**import** java.io.ObjectInputStream;

**import** java.io.ObjectOutputStream;

**import** java.io.OutputStream;

**import** java.util.ArrayList;

**public** **class** **ObjectInputStream\_readUnshared**​ {

// Windows: C:/Data/test/test1.data

**private** **static** String file\_path = "/Volumes/Data/test/test.data";

**public** **static** **void** **main**(String[] args) **throws** IOException, ClassNotFoundException {

writeUnsharedTest();

readUnsharedTest();

}

**private** **static** **void** **writeUnsharedTest**() **throws** IOException {

File file = **new** **File**(file\_path);

file.getParentFile().mkdirs();

ArrayList<String> list = **new** **ArrayList**<String>();

list.add("One");

list.add("Two");

OutputStream os = **new** **FileOutputStream**(file);

ObjectOutputStream oos = **new** **ObjectOutputStream**(os);

oos.writeUnshared(list); // Write the first time

oos.writeUnshared(list); // Write the second time

oos.close();

}

@SuppressWarnings({ "unchecked" })

**private** **static** **void** **readUnsharedTest**() **throws** IOException, ClassNotFoundException {

File file = **new** **File**(file\_path);

ArrayList<String> list = **new** **ArrayList**<String>();

list.add("One");

list.add("Two");

InputStream is = **new** **FileInputStream**(file);

ObjectInputStream ois = **new** **ObjectInputStream**(is);

ArrayList<String> list1 = (ArrayList<String>) ois.readUnshared();

ArrayList<String> list2 = (ArrayList<String>) ois.readUnshared();

System.out.println("list1 == list2? " + (list1 == list2));

ois.close();

}

}

Output:

list1 == list2? false