Bài 13

Work with File(s) API
JAVA File & IO

Abstraction

- Definition of file
- Kinds of file: **Text & Serializable**
- The way to write/read file **text** type and **serializable** file

Overview

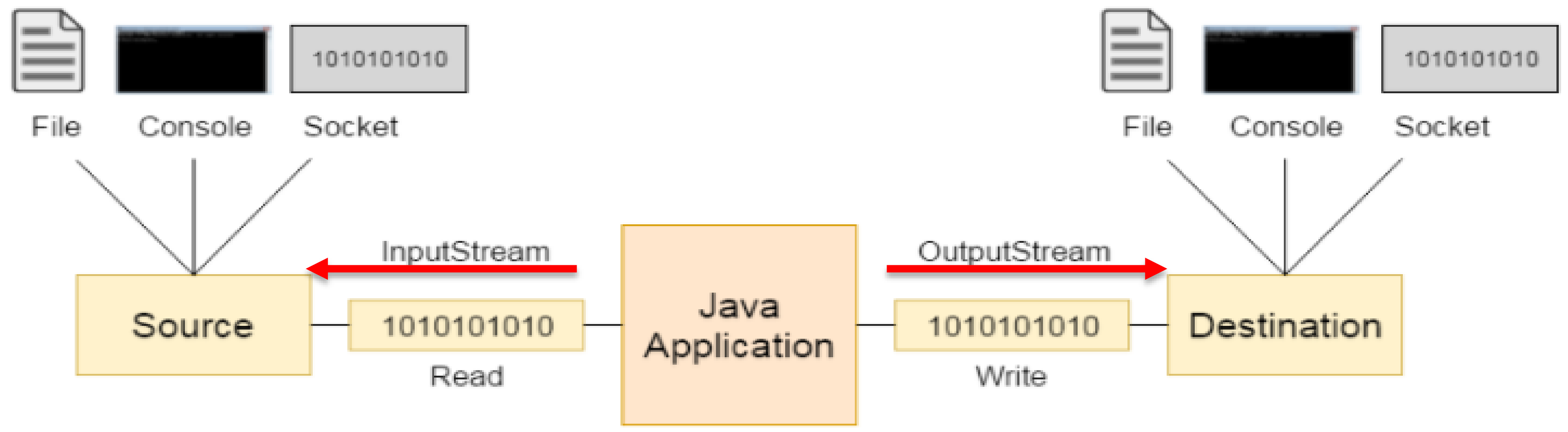
- JAVA IO
  - Java I/O (Input and Output) is used to access the input and produce the output.
  - Java uses the concept of **stream** to make I/O operation fast. The java.io package contains all the classes required for input and output operations.
  - We can perform file handling in java by Java I/O API.
- Stream
  - A stream is a sequence of data. Java stream is composed of **bytes**.
  - It's called a stream because it is like a **stream of water** that continues to flow.
- Basic Stream
  - System.out : syn
  - System.in : syn
  - System.err : arcsyn

Example: Input from console using stream

```
int i = System.in.read();
System.out.println((char)i);
```

InputStream vs OutputStream

- OutputStream
  - Java application uses an output stream to **write data into a destination**, it may be a file, an array, peripheral device or socket.
- InputStream
  - Java application uses an input stream to **read data from a source**, it may be a file, an array, peripheral device or socket.
- Perform operation



FILE

- Class File support some of basic functions to help dev performing easy way
  - Belong to java.io package
  - Accees file, open file
  - Get info from file and directory(folder)
  - Directory is a file

File IO

❖ Create a new file

```
File file = new File("test.txt");
if(!file.exists()){
    try {
        file.createNewFile();
        System.out.println("Create file sucessfull");
    } catch (IOException e) {
        System.out.println("cant create file" + file.getName());
    }
}
```

❖ Create a new directory

```
File dir = new File("programing/test.txt");
if(!dir.isDirectory()){
    dir.mkdirs();
    // dir.mkdir
}
```

- We can create a folder by **mkdir()** or structure folder by **mkdirs()**

❖ **Condition while create a new FILE**

- **Root file or directory make sure that existing in system**

File IO

- **Ex 01**
  - Create a list file in path “…/pathproject/**root**/”
  - Create a list floder in path “…/pathproject/**root**/”
    - Quantity: Input n from keyboard
    - Filename:
      - Get the current time to name of file
      - System.currentTimeMillis() to avoid same filename

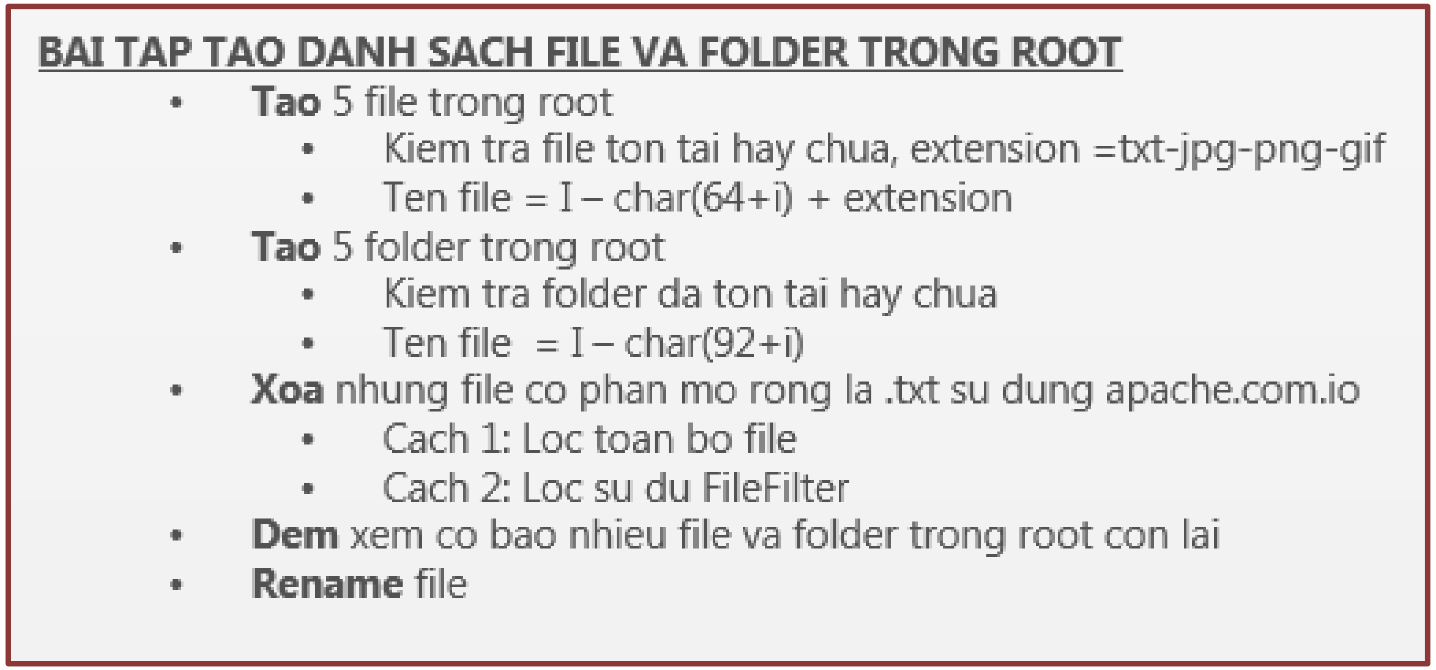
File IO: Useful Methods

File operation

- String getName;
- String getPath
- String getAbsolutePath
- String getCanonicalPath
- String getParent
- boolean renameTo[newName]
- long lastModified
- long length
- boolean delete

Check file

- boolean exists
- boolean canWrite
- boolean canRead



BAI TAP TAO DANH SACH FILE VA FOLDER TRONG ROOT

- **Tao** 5 file trong root
  - Kiem tra file ton tai hay chua, extension =txt-jpg-png-gif
  - Ten file = I – char(64+i) + extension
- **Tao** 5 folder trong root
  - Kiem tra folder da ton tai hay chua
  - Ten file = I – char(92+i)
- **Xoa** nhung file co phan mo rong la .txt su dung apache.com.io
  - Cach 1: Loc toan bo file
  - Cach 2: Loc su du FileFilter
- **Dem** xem co bao nhieu file va folder trong root con lai
- **Rename** file

File IO: Useful Methods

- Directory operation
  - boolean mkdir
  - Boolean mkdirs
  - String [] list

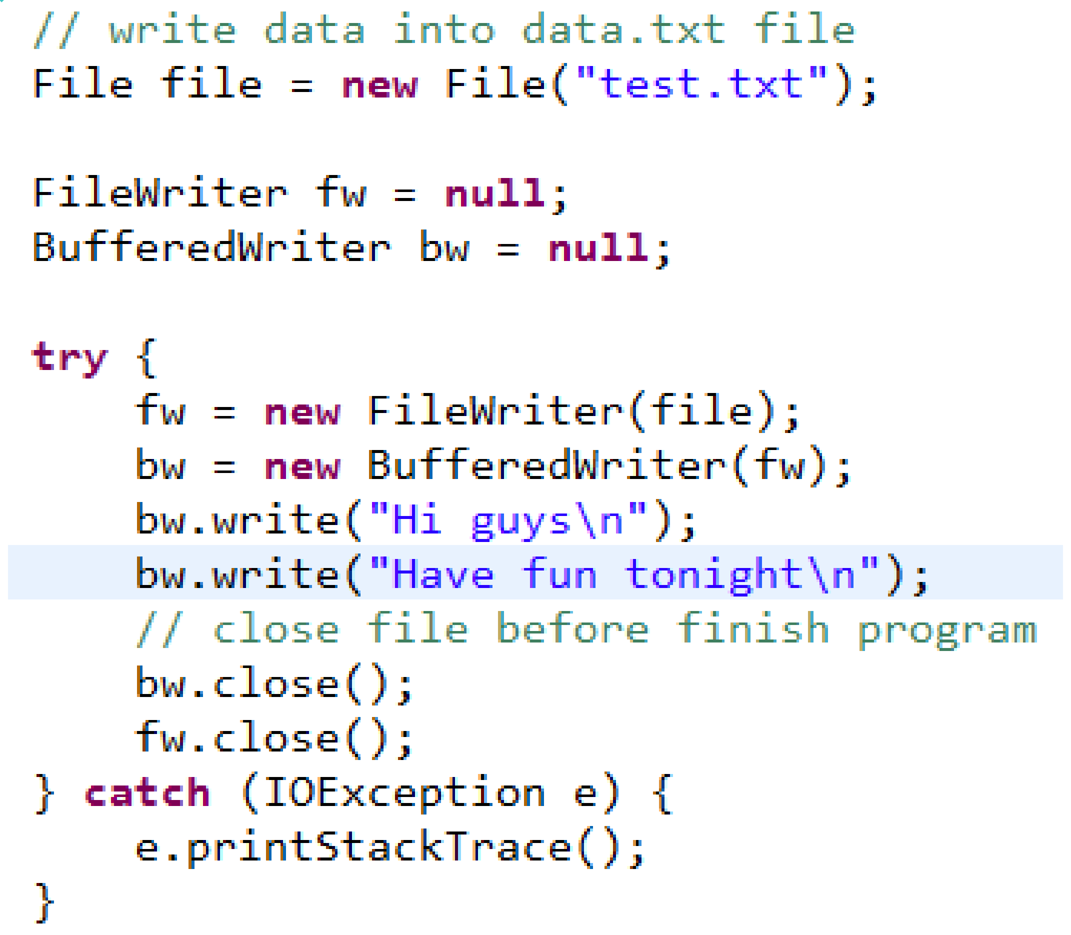
File IO: File Filter

- File class support some methods to get list children file and directory in parent directory. It depends on operating system
  - File [] listRoots : static
  - File [] **listFiles**
  - File [] listFiles[FilenameFilter filter]
  - File [] listFiles[FileFilter filter]
  - String [] list : return path
  - String [] list[FilenameFilter filter]
- FileFilter interface
- FileNameFilter interface

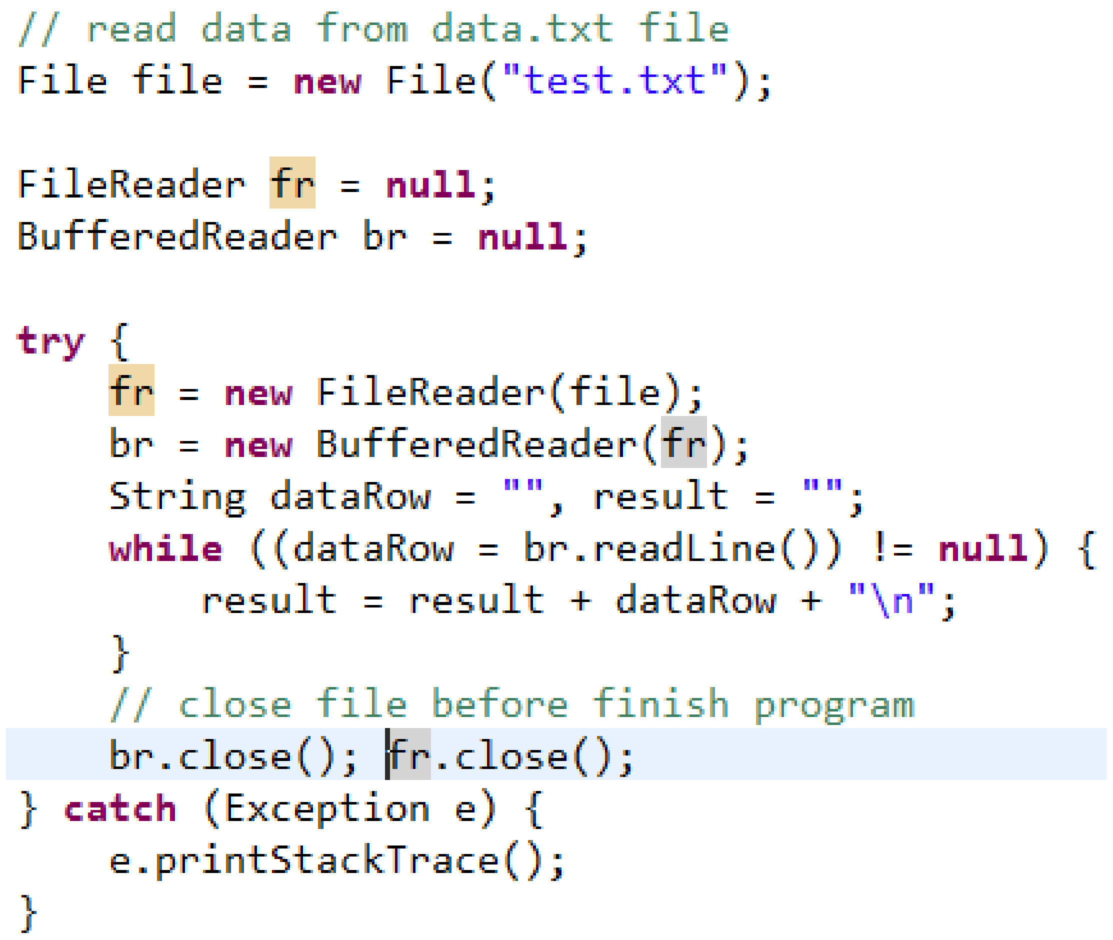
File IO: File Filter

- List all file, folder directly child in “springdata” folder.
- List all root directory in your system
- List all path of file and folder in “springdata” folder
- List all **file which extension is “.config” in F folder**
- Rename file

File IO: Read || Write

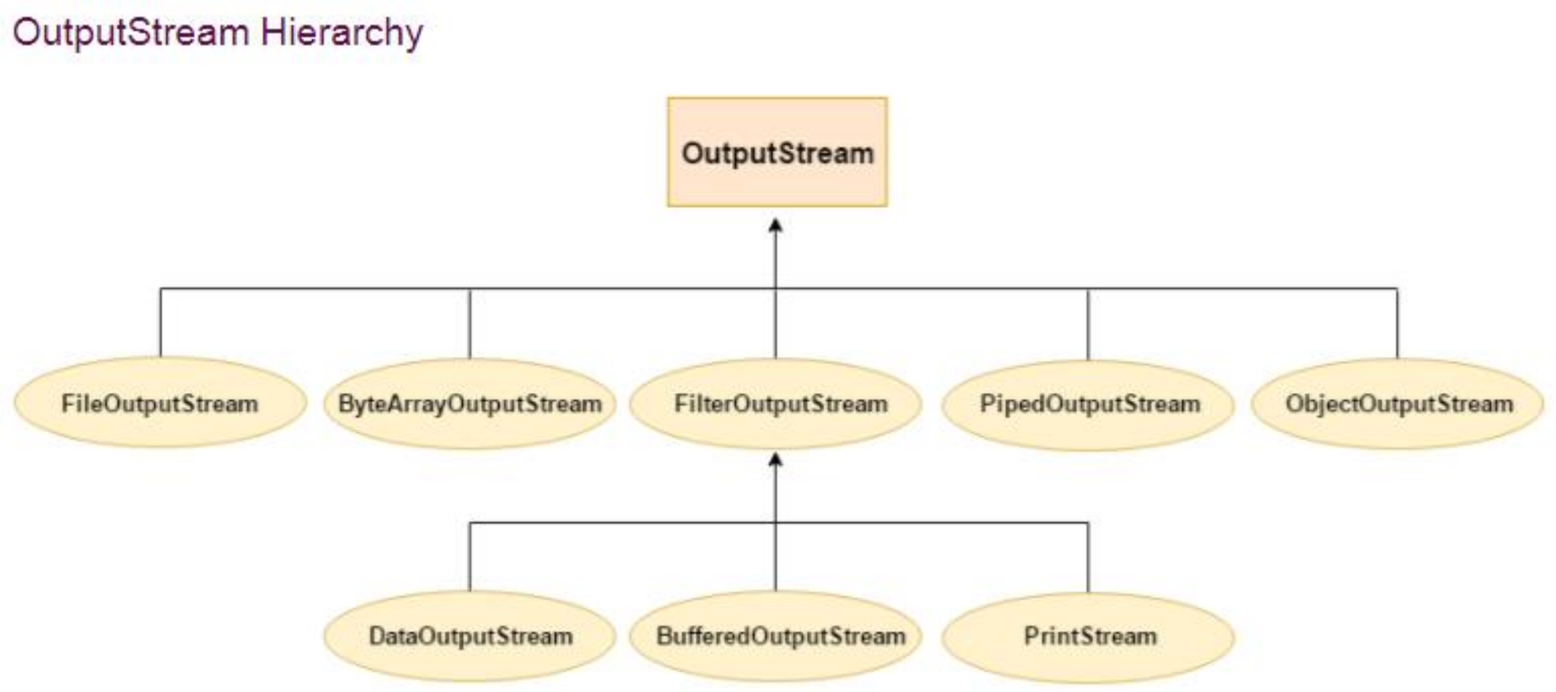


```
// write data into data.txt file
File file = new File("test.txt");

FileWriter fw = null;
BufferedWriter bw = null;

try {
    fw = new FileWriter(file);
    bw = new BufferedWriter(fw);
    bw.write("Hi guys\n");
    bw.write("Have fun tonight\n");
    // close file before finish program
    bw.close();
    fw.close();
} catch (IOException e) {
    e.printStackTrace();
}
```



```
// read data from data.txt file
File file = new File("test.txt");

FileReader fr = null;
BufferedReader br = null;

try {
    fr = new FileReader(file);
    br = new BufferedReader(fr);
    String dataRow = "", result = "";
    while ((dataRow = br.readLine()) != null) {
        result = result + dataRow + "\n";
    }
    // close file before finish program
    br.close(); fr.close();
} catch (Exception e) {
    e.printStackTrace();
}
```

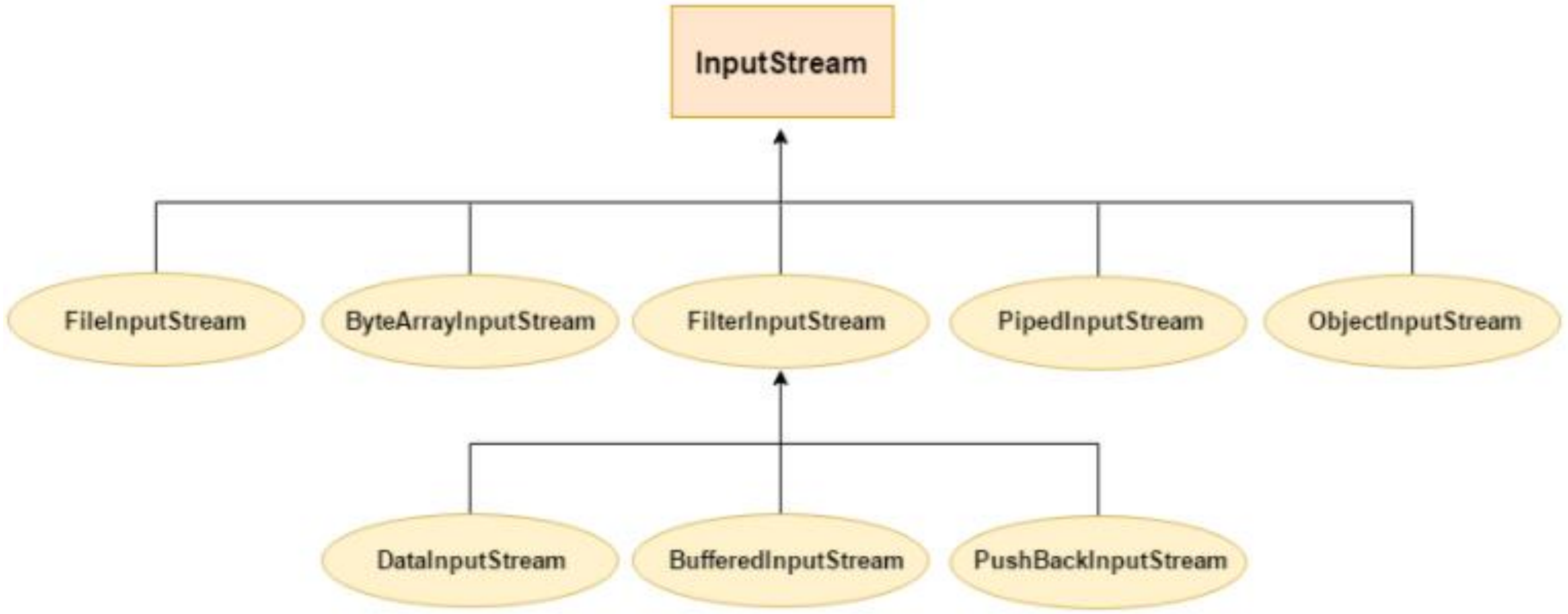
Stream Hierachy



- Write data to destination file

Stream Hierachy

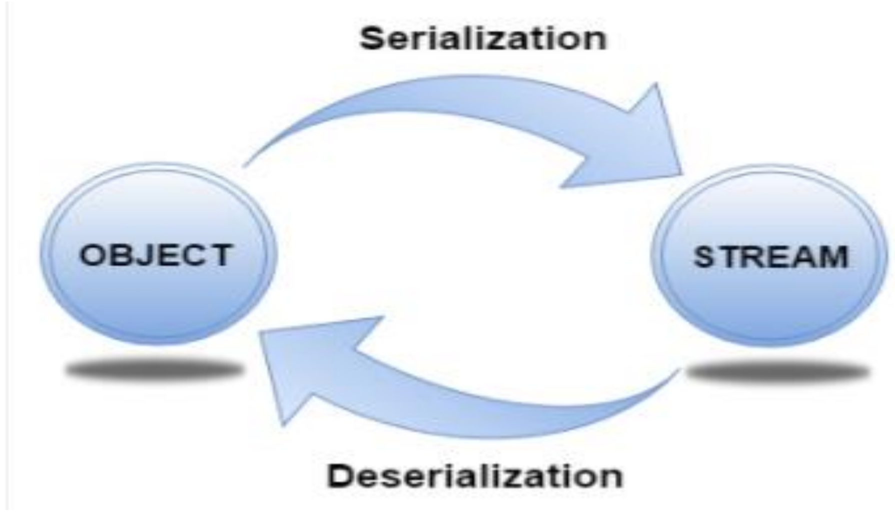
- InputStream Hierarchy

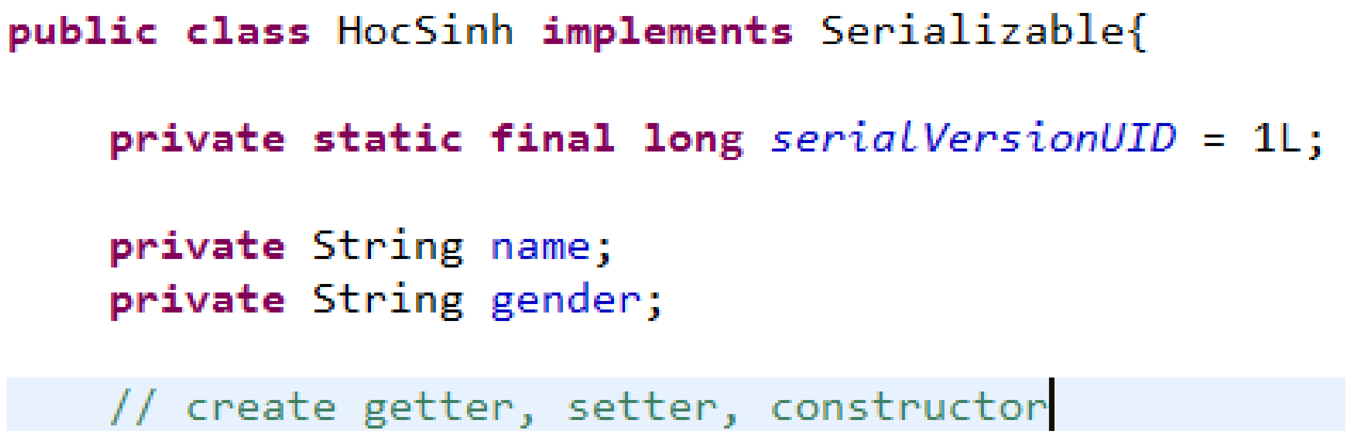


- Read data from source

Serialization

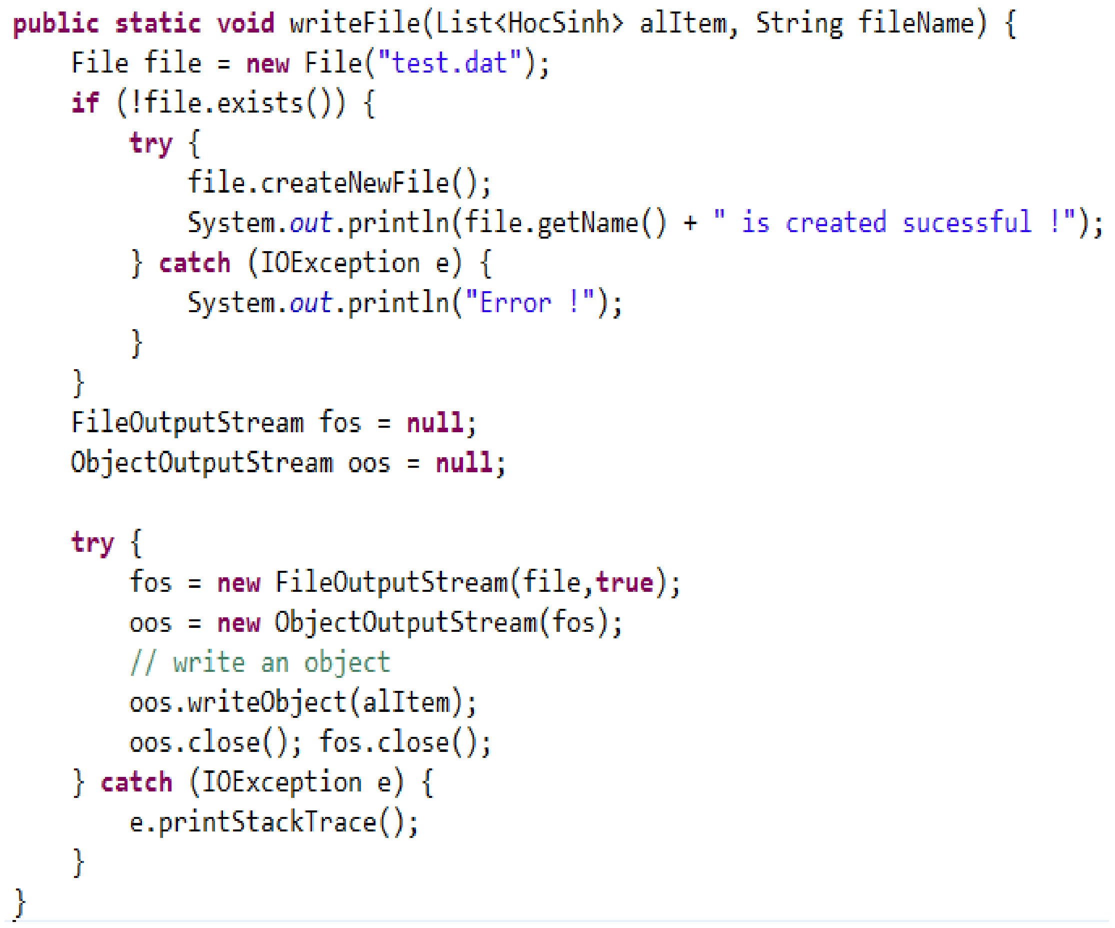
- Serialization in java is a mechanism of writing the state of an object into a byte stream.
- It is mainly used in Hibernate, RMI, JPA, EJB and JMS technologies.
- It must be **implemented** by the class whose **object** you want to persist.



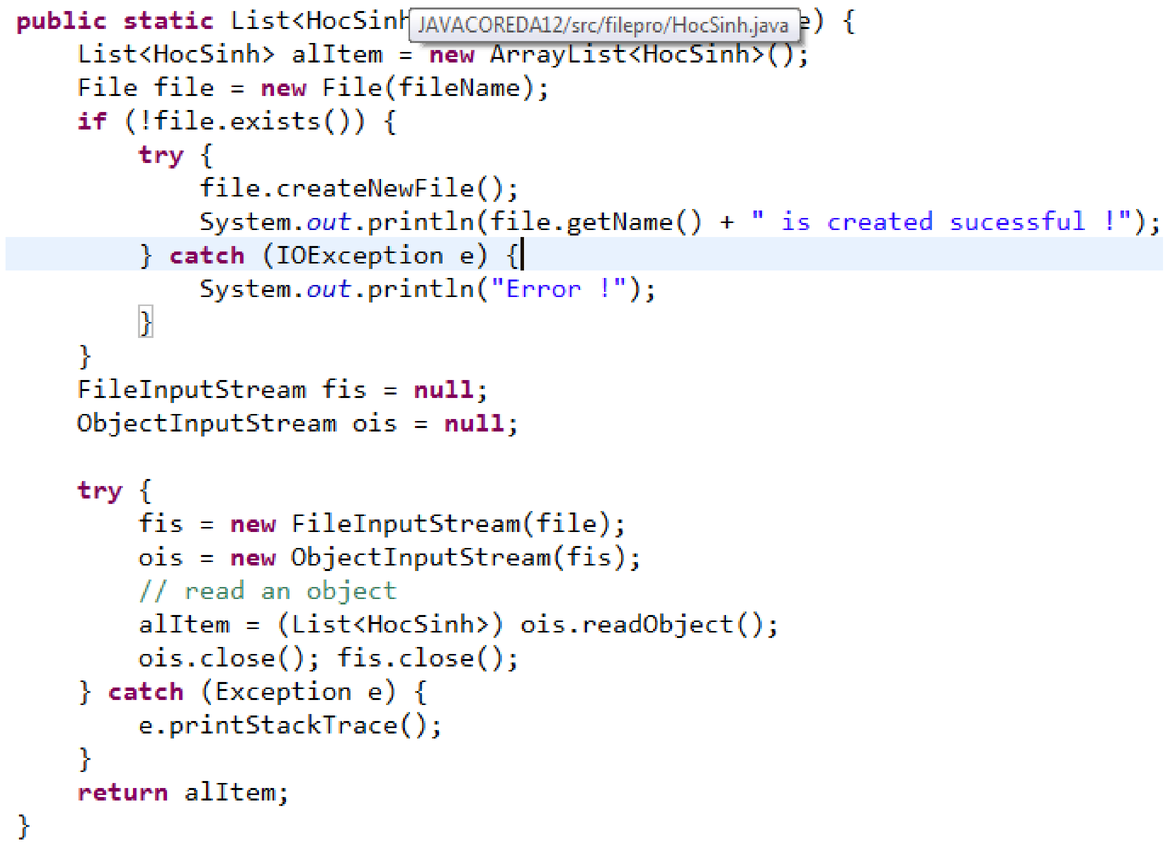


```
public class HocSinh implements Serializable{

    private static final long serialVersionUID = 1L;

    private String name;
    private String gender;

    // create getter, setter, constructor
```

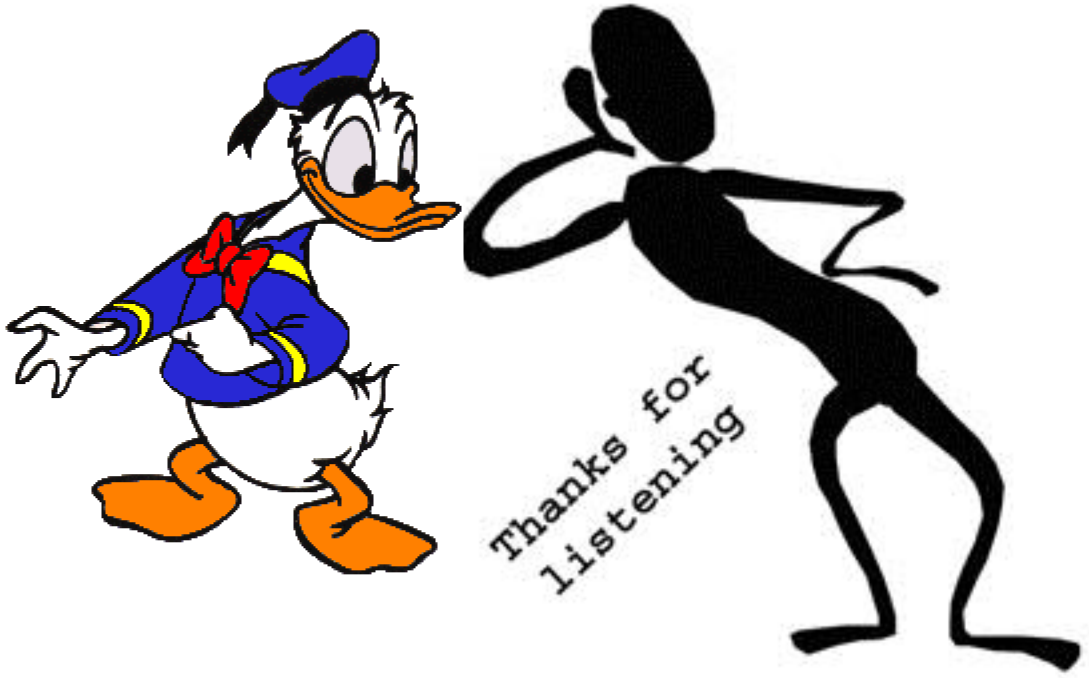
File IO: Read || Write Object



```java
public static void writeFile(List<HocSinh> alItem, String fileName) {
    File file = new File("test.dat");
    if (!file.exists()) {
        try {
            file.createNewFile();
            System.out.println(file.getName() + " is created sucessful !");
        } catch (IOException e) {
            System.out.println("Error !");
        }
    }
    FileOutputStream fos = null;
    ObjectOutputStream oos = null;

    try {
        fos = new FileOutputStream(file,true);
        oos = new ObjectOutputStream(fos);
        // write an object
        oos.writeObject(alItem);
        oos.close(); fos.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```



```java
public static List<HocSinh JAVACOREDA12/src/filepro/HocSinh.java e) {
    List<HocSinh> alItem = new ArrayList<HocSinh>();
    File file = new File(fileName);
    if (!file.exists()) {
        try {
            file.createNewFile();
            System.out.println(file.getName() + " is created sucessful !");
        } catch (IOException e) {
            System.out.println("Error !");
        }
    }
    FileInputStream fis = null;
    ObjectInputStream ois = null;

    try {
        fis = new FileInputStream(file);
        ois = new ObjectInputStream(fis);
        // read an object
        alItem = (List<HocSinh>) ois.readObject();
        ois.close(); fis.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
    return alItem;
}
```



END