Bài 13

Work with File(s) API
JAVA File & IO

Agenda

- Definition of file
- Kinds of file: **Text-Readable** & **Serializable-Unreadable**
- Manipulation
  - Create/Delete/Copy/Upload/Filter with file
  - 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.

Byte Stream

- This mainly incorporates with byte data. When an input is provided and executed with byte data, then it is called the file handling process with a byte stream.

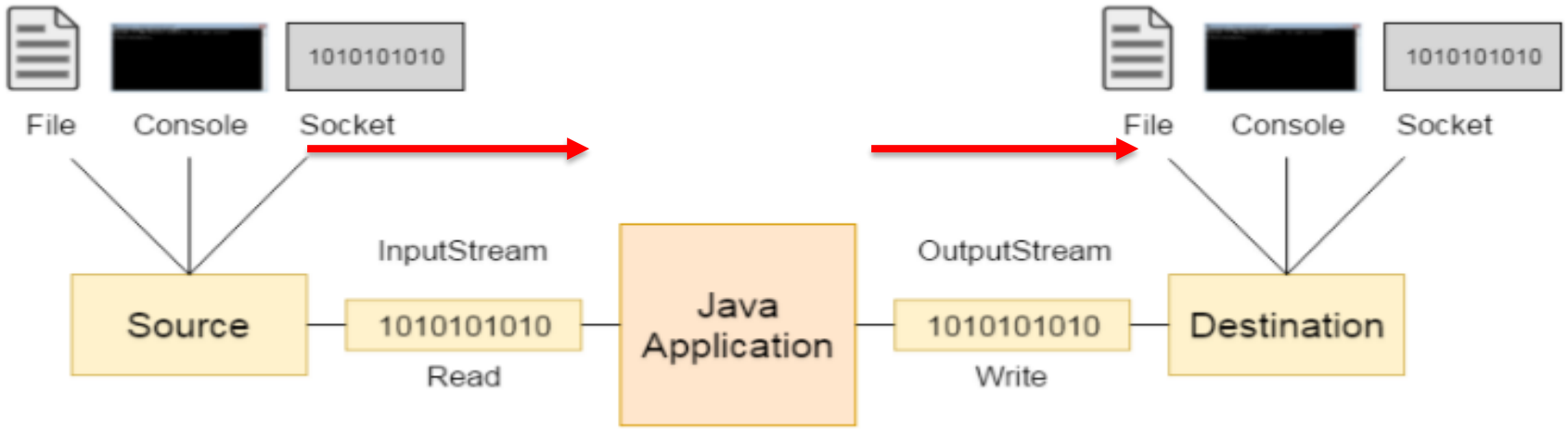
Charater Stream

- Character Stream is a stream which incorporates with characters. Processing of input data with character is called the file handling process with a character stream.

InputStream vs OutputStream

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

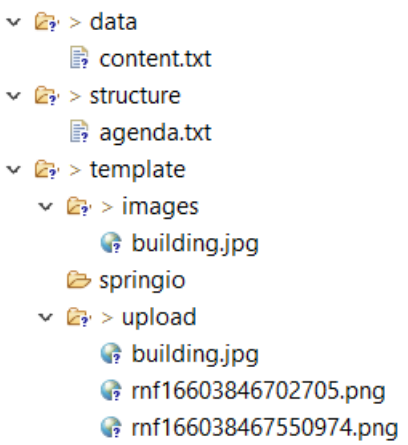
- Perform operation



File API

- Class File support some of basic behaviors to help developers performing easy way
  - Belong to java.io package
  - Manipulate files, folder
  - Directory is a file
  - File could access system-file such as directory.
- Manipulation
  - Open/Close
  - Read/Write
  - Rename/Remove

File IO



❖ Create a new file

```
public static File createNewFile(String pathName) {
    File file = new File(pathName);
    File parent = file.getParentFile();
    if (!parent.exists()) {
        parent.mkdirs();
    }
    if (!file.exists()) {
        try {
            boolean isSuccess = file.createNewFile();
            System.out.println("File " + file.getName()
                                + " is created " + (isSuccess ? "successful" : "fail"));
        } catch (IOException e) {
            e.printStackTrace();
        }
    } else {
        System.out.println("File " + file.getName() + " is already existed");
    }
    return file;
}
```

text\\data\\content.txt

Condition while create a new FILE

Root file or directory make sure that existing in system

File IO

- Create a new directory

```
public static File createNewDir(String pathName) {
    File dir = new File(pathName);
    if (!dir.exists()) {
        boolean isSuccess = dir.mkdirs();
        System.out.println("Folder " + dir.getName()
                           + " is created " + (isSuccess ? "successful" : "fail"));
    } else {
        System.out.println("Folder " + dir.getName() + " is already existed");
    }
    return dir;
}
```

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**
  - boolean exists
  - boolean canWrite
  - boolean canRead
- Directory operation
  - boolean mkdir
  - boolean mkdirs

File IO: File Filter

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

File IO: Read || Write

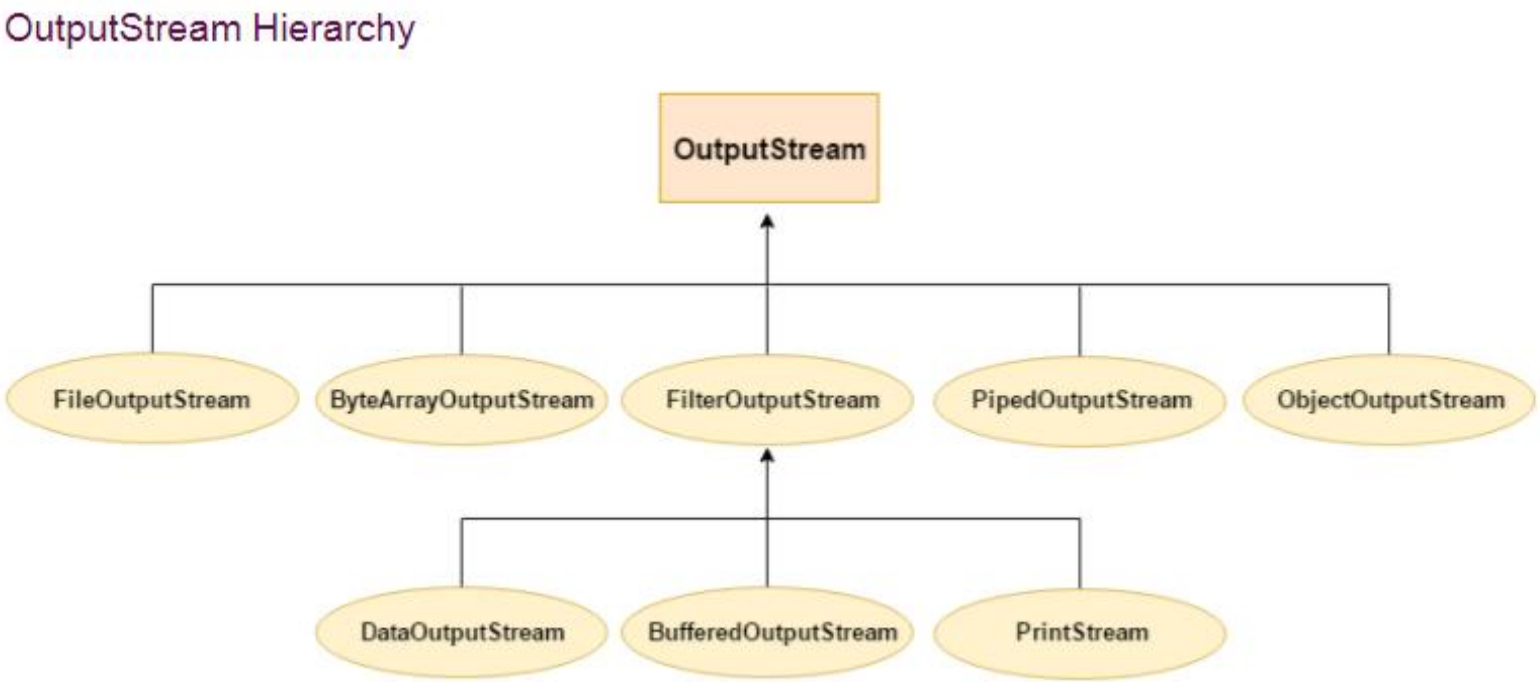
```java
// write data into data.txt file
File file = new File("test.txt");

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();
}
```

```java
// 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();
}
```

File IO: Read || Write

```
public static <DataRow extends AbstractFile> void writeFile(
        String title, List<DataRow> dataRows, File file) {
    FileWriter fw = null;
    BufferedWriter bw = null;
    try {
        fw = new FileWriter(file);
        bw = new BufferedWriter(fw);

        // write-in data
        bw.write(title + "\n");
        bw.write("==========================\n");
        for (DataRow dataRow: dataRows) {
            bw.write(dataRow.toLine());
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        FileUtils.close(bw, fw);
    }
}
```

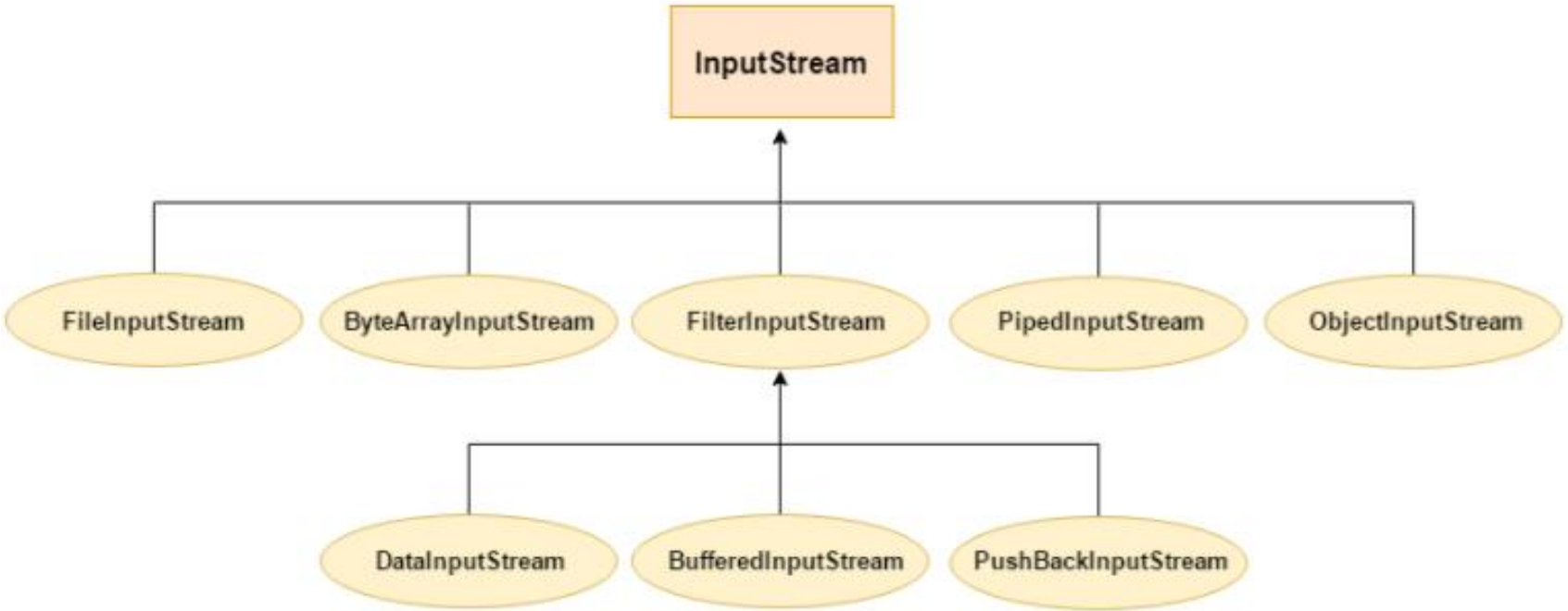
Stream Hierachy



- Write data to target file

Stream Hierachy

- InputStream Hierarchy

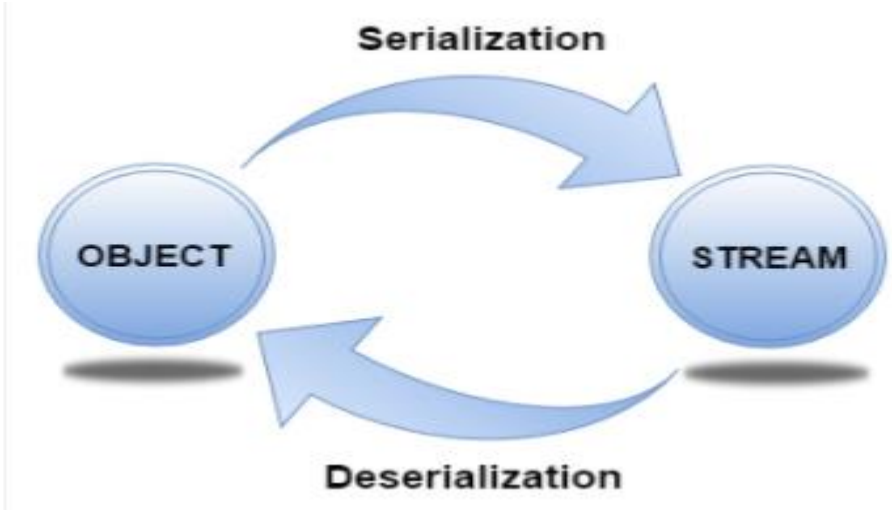


- Read data from source

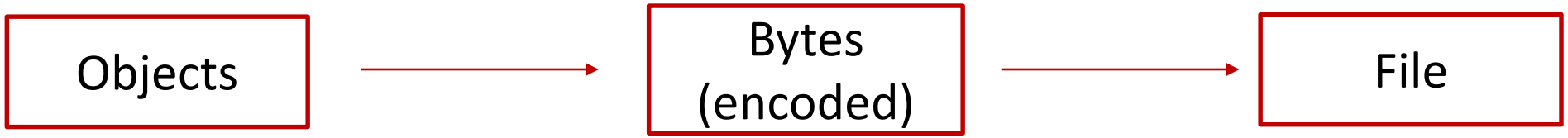
Serialization

- Serialization in java is a mechanism of convert file, object into a byte stream. The objects in the file is converted to the bytes for security purposes. For this, we need to implement java.io.Serializable interface. It has no method to define

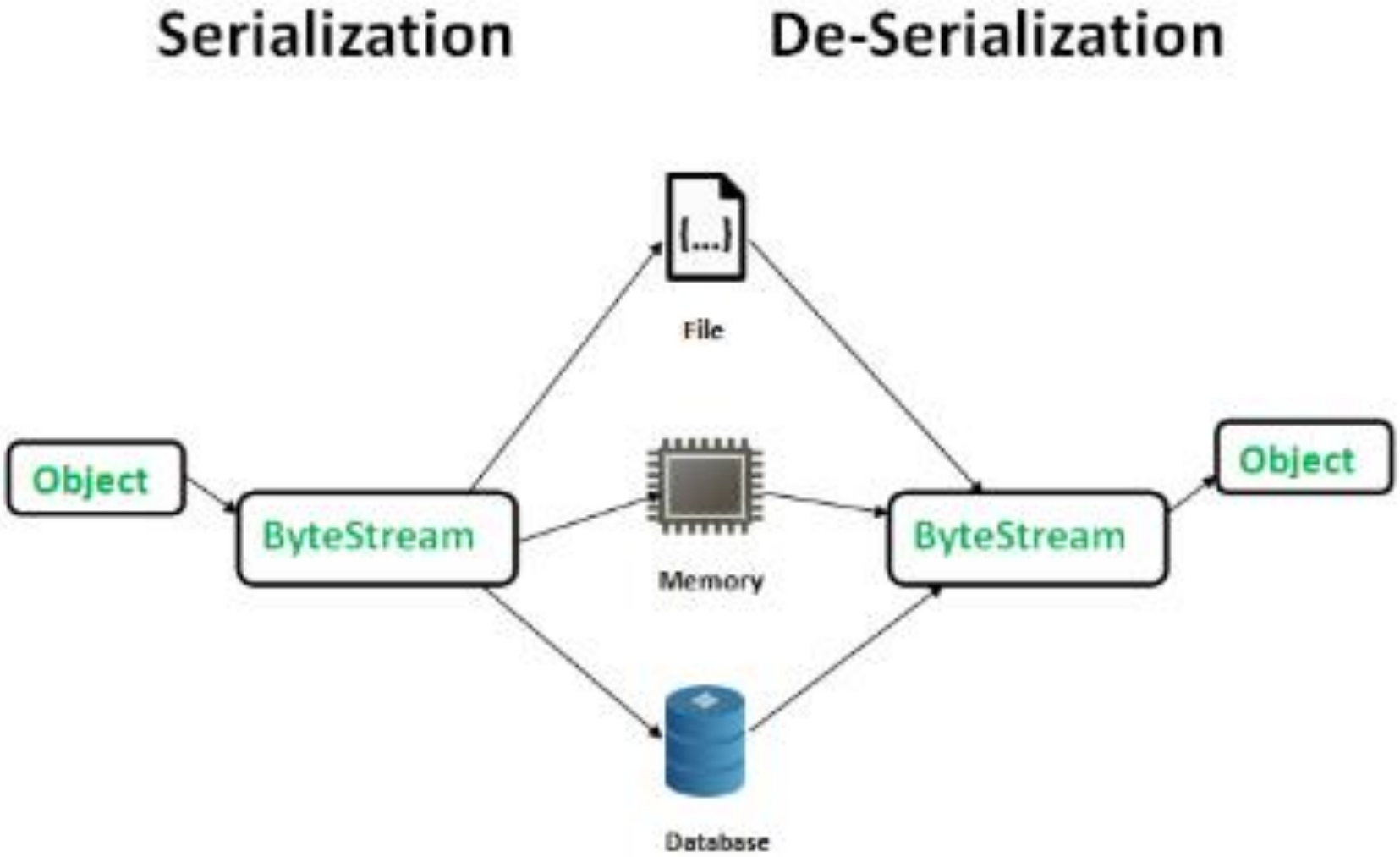
- It is mainly used in Hibernate, RMI, JPA, EJB and JMS technologies.
- It must be **implemented** by the class whose **object** you want to persist.




```java
public class HocSinh implements Serializable{

    private static final long serialVersionUID = 1L;

    private String name;
    private String gender;

    // create getter, setter, constructor
```



Serialization

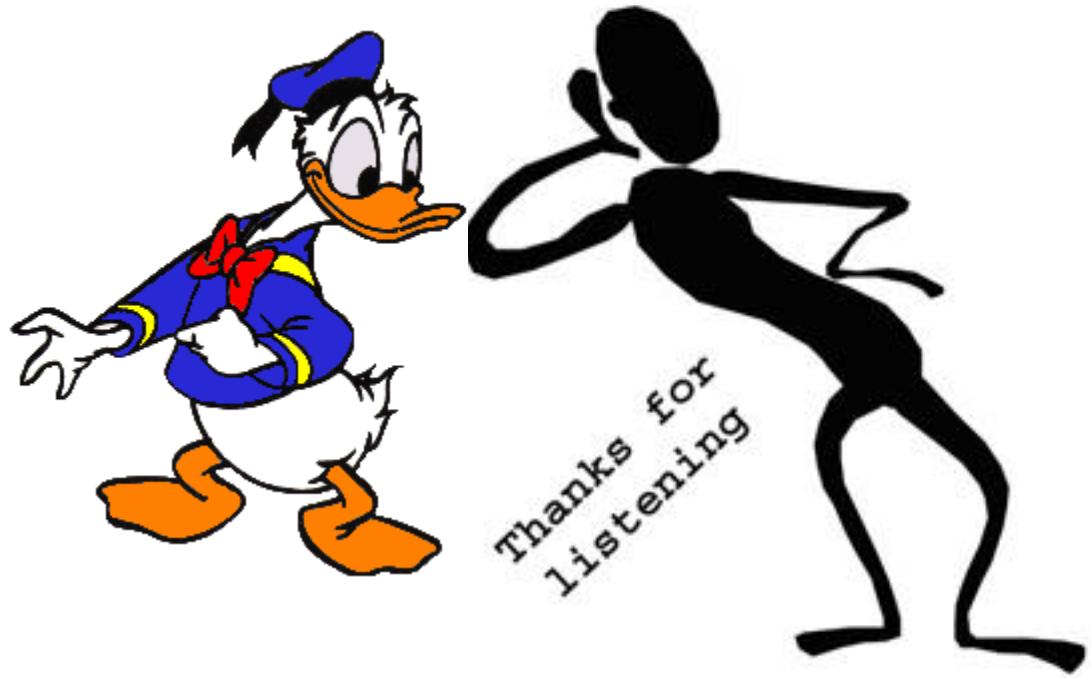


File IO: Read || Write Object

```
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();
    }
}
```

File IO: Read || Write Object

```
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