Low Level Input and Output

Sungchul Lee

Learning Object

- Java.io Package
  - File Class
  - FileInputStream
  - FileOutputStream
- Type of File I/O
  - Low Level Input and Output

. Limited memory
. Permanently store data
. Some data is needed immediately, others are needed months/weeks/.. from now

Why?

Built-in Package for Input and Output

- The **java.io** contains all the classes required for input and output
- Write data on File
  - **FileOutputStream** is used to handle raw binary data.
  - **DataOutputStream** lets an application write primitive data types
  - **PrintWriter** is used to send characters to a text file.
- Read data from File
  - **FileInputStream** is used to read bytes from file
  - **DataInputStream** allows an application to read primitive data
  - **FileReader** is meant for reading streams of characters
  - **BufferedReader** reads text from a character-input stream

The **File** Class

- Need to know the file before writing and reading data on/from file
- The File class is useful for getting information about a file
  - For example:
  - It can tell you whether the file exists, is readable, is writeable, is hidden, and is a directory.
  - It can give you the file name, the parent directory's name, the time last modified, and the file length.
  - If the file is a directory, it can give you a list of the files in the directory.
- You should import the java.io package to use File class: **import** java.io.*;

Generate File Objects

- Syntax:

```
import java.io.*;
File <variable_name> = new File(<filename>);
File <variable name>=new File(<directory>, <file name>);
```

- To operate on a file, we must first create a

File object (from java.io package) and associate this file object to **a specific file name**.

File Class - Example

- Open the file “sample.dat” in the current directory (Project folder)

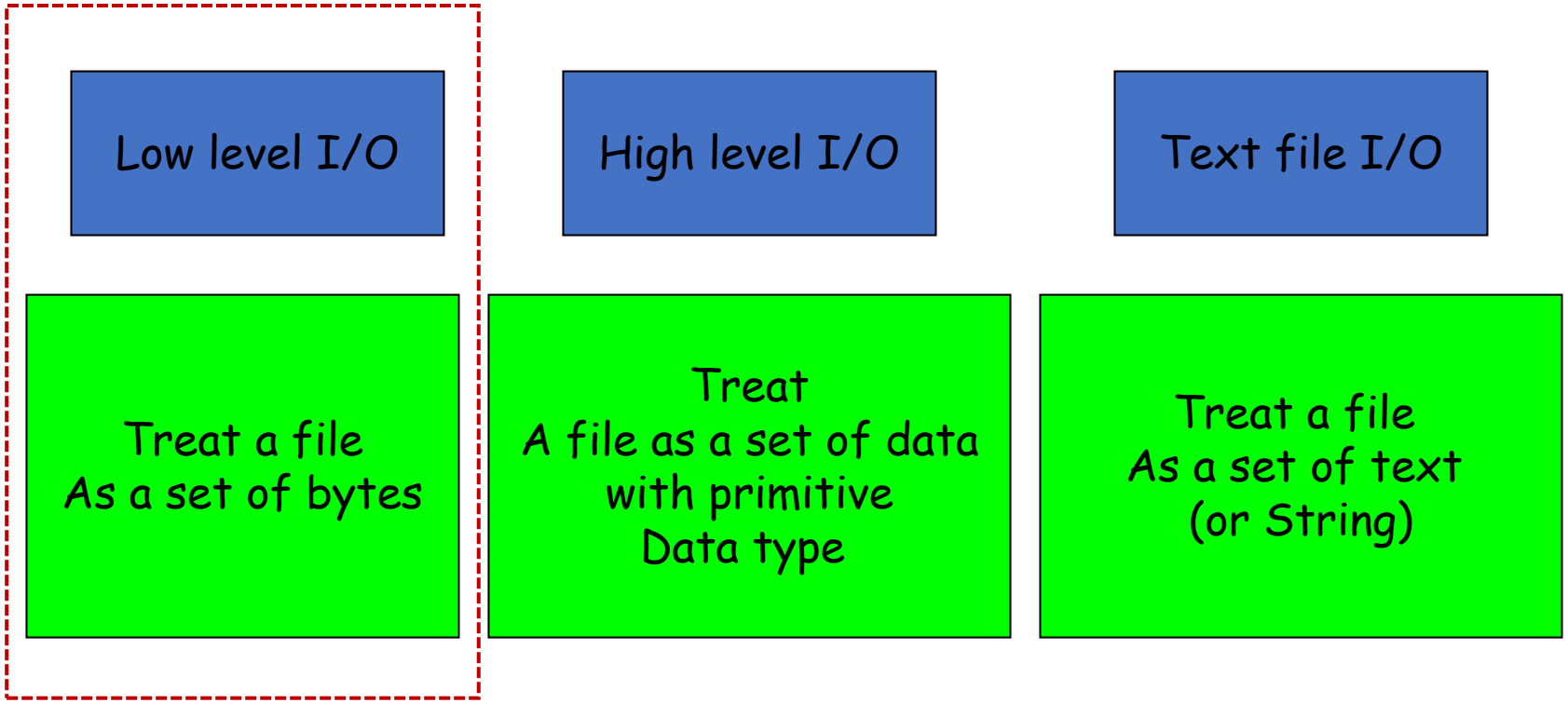
```
File file = new File (“sample.dat”)
```

- Open the file “test.dat” in the directory "C:/Users/lee/Downloads/test.dat“

```
File file = new File("C:/Users/lee/Downloads/test.dat");
```

Type of File I/O

- Three type of file I/O



Low-Level File I/O

- Most of data is stored in binary foramt to compress
- To read data from or write data to a binary foramt file, we must create one of the Java **stream** objects and attach it to the file.
- A stream is a sequence of data items, usually 8-bit bytes.
  - A pipe connecting files and original program
- A stream either produces or consumes information
- Java has two types of streams
  - Input stream to read a byte of data
  - Output stream to write a byte of data

Streams for Low-Level File I/O

- **FileOutputStream** and **FileInputStream** are two stream objects that facilitate file access.
- **FileOutputStream** allows us to output a **sequence of bytes**; values of data type byte.
- **FileInputStream** allows us to read in an **array of bytes**.

TestBinary.bson

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  54 68 69 73 20 77 69 6C 6C 20 62 65 20 61 20 34  This will be a 4
00000010  30 47 42 20 62 79 74 65 20 73 74 72 65 61 6D 21  0GB byte stream!
00000020  64 00 00 00 03 48 65 61 64 65 72 00 4E 00 00 00  d....Header.N...
00000030  03 53 75 62 48 65 61 64 65 72 31 00 21 00 00 00  .SubHeader1.!...
00000040  02 4E 61 6D 65 00 05 00 00 00 42 6F 6E 64 00 10  .Name.....Bond..
00000050  4C 69 63 65 6E 73 65 00 07 00 00 00 00 03 53 75  License.......Su
00000060  62 48 65 61 64 65 72 32 00 10 00 00 00 08 49 73  bHeader2......Is
00000070  41 63 74 69 76 65 00 01 00 00 0A 50 61 79 6C 6F  Active.....Paylo
00000080  61 64 00 00                                       ad..
```

How to Write Low Level Data

- Step 1: Create a File object
  - File name and directory path
- Step 2: Create a **FileOutputStream** object
  - Use file object
- Step 3:Get data ready
  - Your data
- Step 4:**Write data** to **output stream**
- Step 5: **Close** the file

OutputStream Methods

- Syntax:
  - FileOutputStream outStream = new FileOutputStream(File Object);
  - outStream.method()

Method	Description
void close( )	Closes the output stream. Further write attempts will generate an **IOException**.
void flush( )	Causes any output that has been buffered to be sent to its destination. That is, it flushes the output buffer.
void write(int *b*)	Writes a single byte to an output stream.
void write(byte[ ] *buff*)	Writes a complete array of bytes to an output stream.

How to Read Low Level Data

- Step 1: Create a File object
  - File name and directory path
- Step 2: Create a FileInputStream object
  - File object
- Step 3: Declare an array to keep input data, allocate memory for this array
  - byte array
- Step 4: Read data and process data if needed
- Step 5: Close the file

InputStream Methods

- Syntax:
  - FileInputStream inStream = new FileInputStream(File Object);
  - instream.method()

Method	Description
int available( )	Returns the number of bytes of input currently available for reading.
void close( )	Closes the input source. Further read attempts will throw an **IOException**.
int read( )	Returns an integer representation of the next available byte of input. If the end of the stream is encountered, -1 is returned.
int read(byte[] *buffer*)	Attempts to read up to *buffer*.length bytes into buffer and returns the number of bytes that were read.

Practice

1. Make a new project (Reference: Create Project and Class File)
   - Project name: Low_Level_IO
2. Create a new Class File
   - Class name: Main
3. Coding:
   - Note:
     - **import java.io.File;**
     - **import java.io.FileInputStream;**
     - **import java.io.FileOutputStream;**
     - **import java.io.IOException;**

Coding – Main (import and File Class)

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

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

    //set up file and stream
    File file = new File("ch.dat");
    //"C:/Users/lee/Downloads/test.dat"
```

Coding – Main (FileOutputStream class)

```
// Generate Output Stream object
FileOutputStream outStream = new FileOutputStream( file );

//Prepare data to save
byte[] byteArray = {100,20};
//write data to the stream
outStream.write( byteArray );
outStream.flush();
//output done, so close the stream
outStream.close();
```

Coding – Main (FileInputStream class)

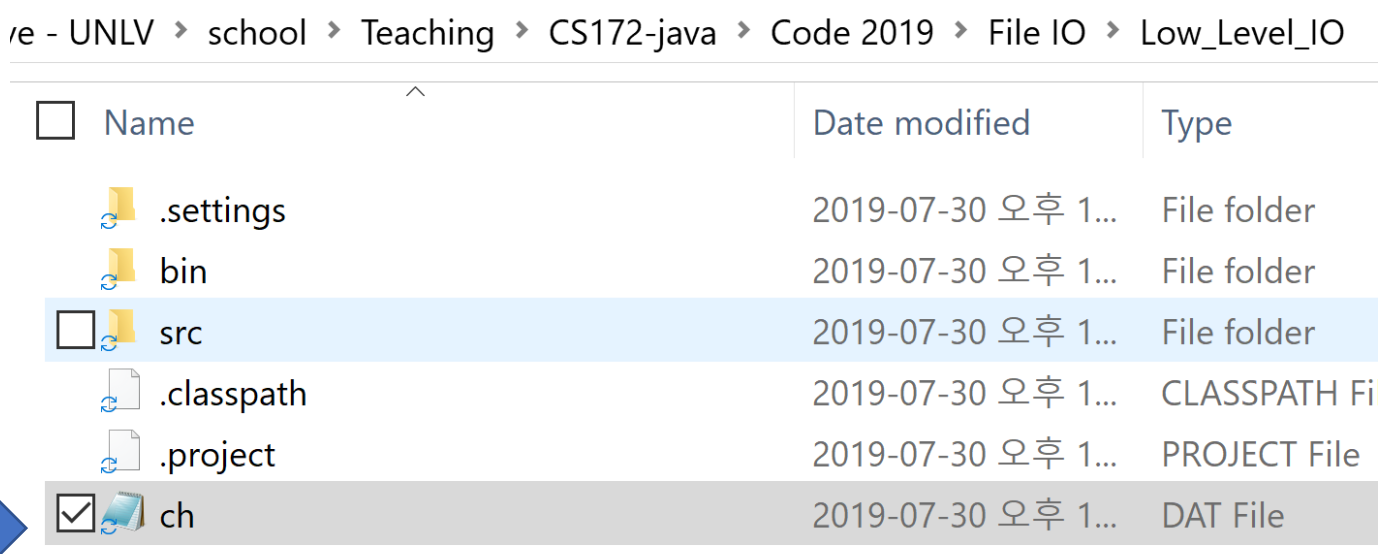
```
    //Generate Input Stream Object
    FileInputStream inStream = new FileInputStream(file);
    //set up an array to read data in
    int    fileSize  = (int)file.length();
    byte[] byteArray2 = new byte[fileSize];

    //read data in and display them
    inStream.read(byteArray2);
    for ( int i = 0; i < fileSize; i++) {
    System.out.println(byteArray2[i]);
    }
    inStream.close();
}  }
```

Practice – Code

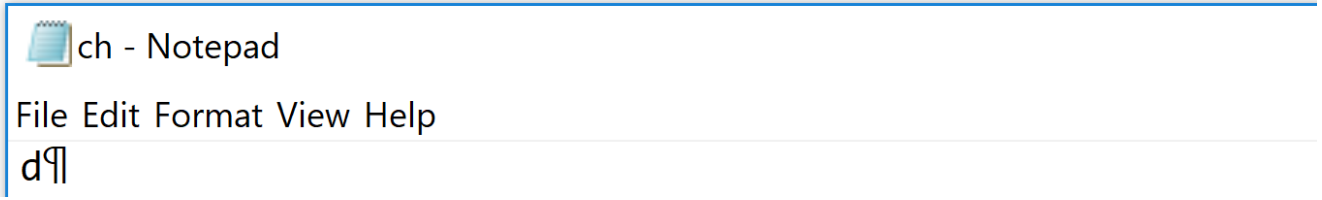
Main.java

```java
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

public class Main {
    public static void main(String[] args) throws IOException {
        // TODO Auto-generated method stub

        //set up file and stream
        File file = new File("ch.dat");
        //"C:/Users/lee/Downloads/test.dat"

        // Generate Output Stream object
        FileOutputStream outStream = new FileOutputStream( file );

        //Prepare data to save
        byte[] byteArray = {100,20};
        //write data to the stream
        outStream.write( byteArray );
        outStream.flush();
        //output done, so close the stream
        outStream.close();

        //Generate Input Stream Object
        FileInputStream inStream = new FileInputStream(file);
        //set up an array to read data in
        int    fileSize  = (int)file.length();
        byte[] byteArray2 = new byte[fileSize];

        //read data in and display them
        inStream.read(byteArray2);

        for ( int i = 0; i < fileSize; i++) {
            System.out.println(byteArray2[i]);
        }
        inStream.close();
    }
}
```

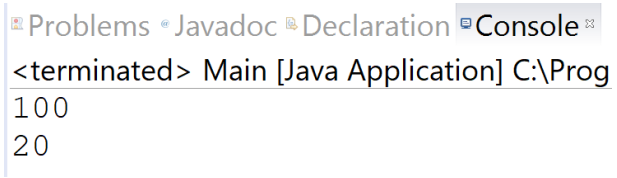
Practice – Result



Generated File



Byte Type Data



Result

Summary

- Java.io Package
  - File Class
    - File file = new File(“filename");
  - FileOutputStream
    - FileOutputStream outStream = new FileOutputStream(file);
    - outStream.write(byteArray);
  - FileInputStream
    - FileInputStream inStream = new FileInputStream(file);
    - inStream.read(byteArray2);
- Type of File I/O
  - Low Level Input and Output