JAVA IO AND NETWORKING

Parsa Hejabi, Fateme Sadat Mahmoudi

TABLE OF CONTENTS

FILE

- What is a File?
- Why do we use Files?
- How we use Files in our programs?
- Different kinds of Files?
  - Binary Files & Text Files

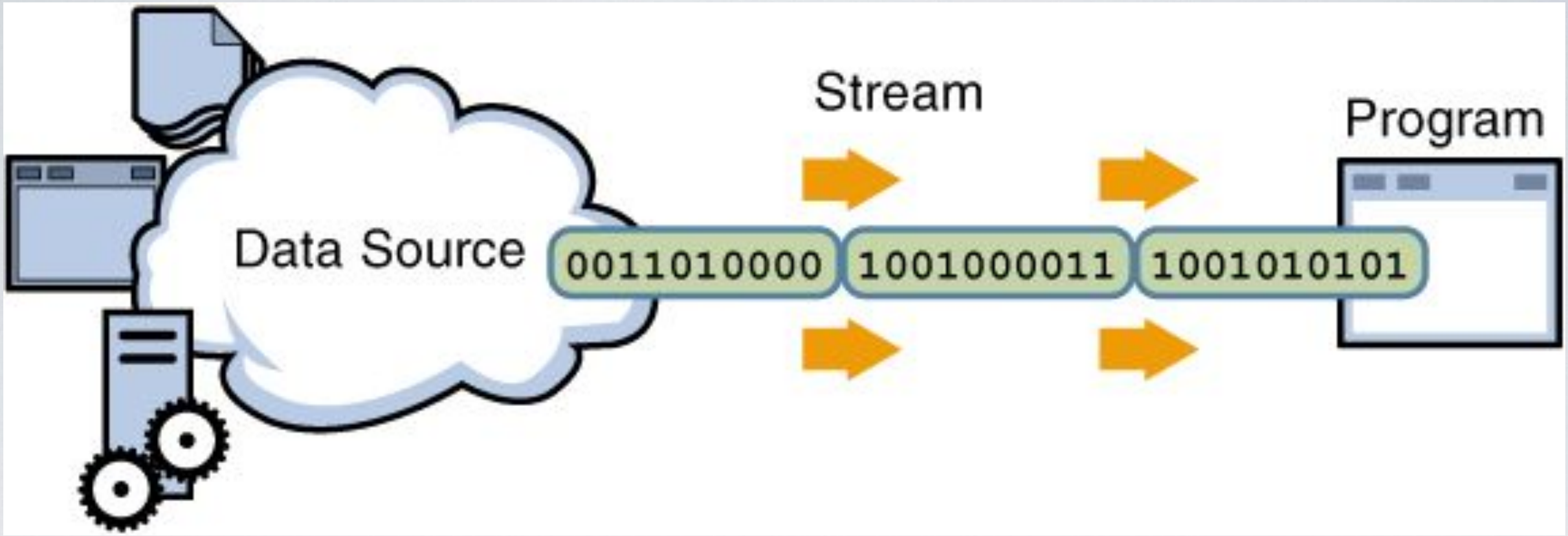
KINDS OF FILES

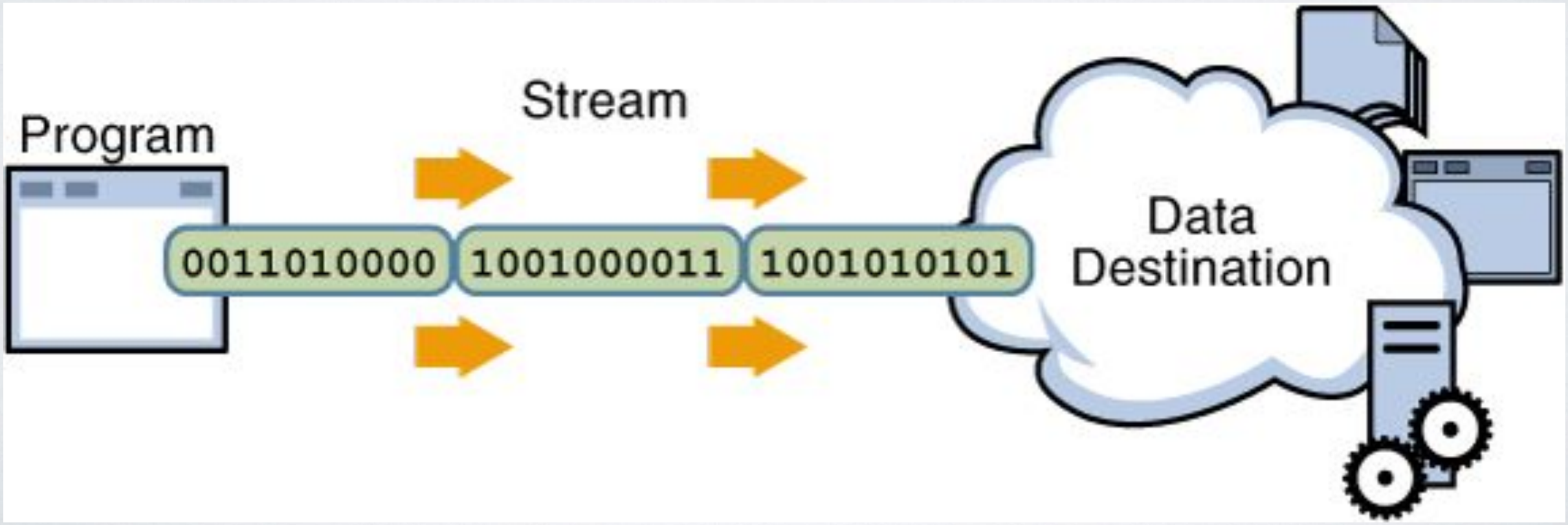
- Text Files:
  - *Characters*
  - *txt, fxml, html…*

- Binary Files:
  - *Bytes*
  - *zip, exe, pdf…*

STREAMS

- Stream of data in or out of a program:
  - *Save or restore from a file*
  - *Send data by* network
    - *Communicate with devices like monitors, printers…*
- Input Streams and Output Streams.


Stream
Program
Data Source
0011010000
1001000011
1001010101


Program
Stream
0011010000
1001000011
1001010101
Data
Destination

JAVA IO CLASS

- Textual input and outputs
  - *Stream of* characters
  - *Reader and Writer class*
    - *Read and Write texts: Telegram*
- Binary input and outputs
  - *Stream of* Bytes
  - *InputStream and OutputStream class*
  - *Read and Write Files: pdf*

ONE CLOSE() METHOD FIRST!

TEXTUAL FILES

FILEREADER CLASS

- *Reader* is an *abstract* Class in Java
- *FileReader* is a sub class of *Reader* class
- For reading text

QUIZ 🤔

```
FileReader inf = new FileReader("readme.txt");
int chCode;
while(-1 != (chCode=inf.read()))
System.out.println("Next:"+(char)chCode);
inf.close();
```

FILEWRITER CLASS

- *Writer* is also an *abstract* Class in Java
  - *FileWriter* is a sub class of *Writer* class
- For writing text
- If the file doesn't exist it will be created.
- If the file already exist the content will be deleted.

```
FileWriter wr = new
FileWriter("text.txt", true);
```

EXCEPTIONS!

- there is gonna be a IOException! like:
  - FileNotFoundException: when you want to work with a file that doesn't exist!
  - EOFException: Signals that an end of file or end of stream has been reached unexpectedly during input.
  - Trying to access a system file.
  - …

BINARY STREAMS

- InputStream: Reading from data Stream.
- OutputStream: writing to data Stream.
- Every instances of these classes will connect to a data Stream.
- Data Stream samples:
  - *File, send and receive data over network, communicate with devices…*
  - *Standard input and output:*

```
InputStream is = system.in;
OutputStream os = System.out;
```

FILEINPUTSTREAM CLASS

- For reading from file.
- Extends *InputStream* class
  - .read() method.
  - .read(byte[]) method.

```
List<Byte> list = new ArrayList<>();
FileInputStream inf = null;
try{
   inf = new FileInputStream(“file”);
   int bCode;
   while(-1 != (bCode=inf.read()))
   list.add((byte)bCode);
}
finally{
   if(inf!=null)
      inf.close();
}
```

FILEOUTPUTSTREAM CLASS

- For writing to file.
- Extends *OutputStream* class
  - .flush() method.
  - .write() method.

QUIZ 🤔

```
int[] numbers = {1234567890, 1234567891,
1234567892};
byte[] array = intToByteArray(numbers);//length=12
FileOutputStream out = null;
try{
   out = new FileOutputStream(“file”);
   out.write(array);
}
finally{
   if(out!=null)
      out.close();
}
```

ADDRESSING

- Most of OS's use "/" for dividing folders.
  - /home/AP/TA
- We use "\" in windows for dividing folders.
  - c:\program files\ap.txt
- But "\" is an *escape character*!
- Solutions:
  - "c:\\textfiles\\newfile.txt"
  - "c:/textfiles/newfile.txt"

SCANNER CLASS

- For reading text.
- Is not in java.io (in java.util).
- system.in is an InputStream.
- Scanner can be used by *InputStreams* and *Readers*.

```
s = new Scanner("1.txt");
```

```
s = new Scanner(new File("1.txt"));
```

```
s = new Scanner(new FileInputStream("1.txt")); s = new Scanner(new FileReader("1.txt"));
```

FORMATTER CLASS

- For writing text.
- Is not in java.io (in java.util).
- Like printf in c.
- Formatter can be used by *OutputStreams* and *Writers*.

```
Formatter f = new Formatter(new
FileWriter("l.txt"));
f = new Formatter(new
FileOutputStream("l.txt"));
f = new Formatter(new File("l.txt"));
f = new Formatter("l.txt");
f = new Formatter(System.out);
f.format("age=%d,name=%s,grade=%
.2f", 20, "Ali", 18.453);
```

OTHER IMPORTANT CLASSES

- *DataInputStream & Data OutputStream:*
  - For Reading and writing binary data.
  - for reading primitive types:
    - readBoolean, readChar, readDouble, readInt…

- *BufferedWriter & BufferedReader*
- *BufferedInputStream & BufferedOutputStream*

- Buffer input and outputs: increase speed.
- writing is not constantly done! use .flush()!

- *ByteArrayInputStream, ByteArrayOutputStream*

- *StringReader, StringWriter*

- *PrintStream*

DECORATOR DESIGN PATTERN

```
FileOutputStream file = new FileOutputStream(“c:/f.txt”);
BufferedOutputStream buffer = new
BufferedOutputStream(file);
PrintStream print = new PrintStream(buffer);
;("print.println(“salam
```

QUIZ 🤔

```
ByteArrayOutputStream baos = new
ByteArrayOutputStream();
DataOutputStream dos = new DataOutputStream(baos);
byte[] bytes;
dos.writeInt(2147483647);
bytes = baos.toByteArray();
System.out.println(bytes.length);
baos.reset();
dos.writeDouble(1);
bytes = baos.toByteArray();
System.out.println(bytes.length);
```

CLOSABLE INTERFACE

- Most of IO classes Implement *Closable Interface*.
- This Interface has .close() method.
- From Java 1.7 *Closable* implements *AutoClosable Interface*.
- try-with-resources: in the end of try block resources will be automatically closed!

```
try (FileReader fr = new FileReader("1.txt")){
    int read = fr.read();
    ...
}
```

REAL QUIZ! 😅

1. Which one is used for file size? a.File b.FileReader

2. Which class is used for writing AND reading? a.File b.RandomAccessFile c.Formatter d.InputStream

3. If we read a file created by *FileWriter* with *FileInputStream*; is there gonna be an exception?

IO IS FINISHED, LET'S DIVE INTO NETWORK…