COSC 121
Computer Programming II

Binary I/O

Part 1/2

Dr. Mostafa Mohamed

Acknowledgement: The slides mainly rely on the textbook of Y. D. Liang titled "Introduction to Java Programming, 10th Ed.", Pearson Edu. Inc.

Previous Lecture

- Text I/O
  - `Scanner, PrintWriter`
  - `BufferedReader, BufferedWriter`
- The `File` Class
- Try-with-resources
- Reading from the Web

Outline

Today:

- Text I/O vs Binary I/O
- Binary I/O classes
  - `InputStream` / `OutputStream`
  - `FileInputStream` / `FileOutputStream`
- ***Midterm 1***

Next lecture:

- More Binary I/O classes:
  - `DataInputStream` / `DataOutputStream`
  - `ObjectInputStream` / `ObjectOutputStream`
    - `Serializable, transient.`
- Improving I/O Performance
  - `BufferedInputStream / BufferedOutputStream`

Text I/O vs. Binary I/O

Storing Data on a Computer

Computers do not differentiate between binary files and text files. **All files/data are stored in binary format**, and thus all files are essentially binary files.

Here are a few examples (*conversion details are not important*):

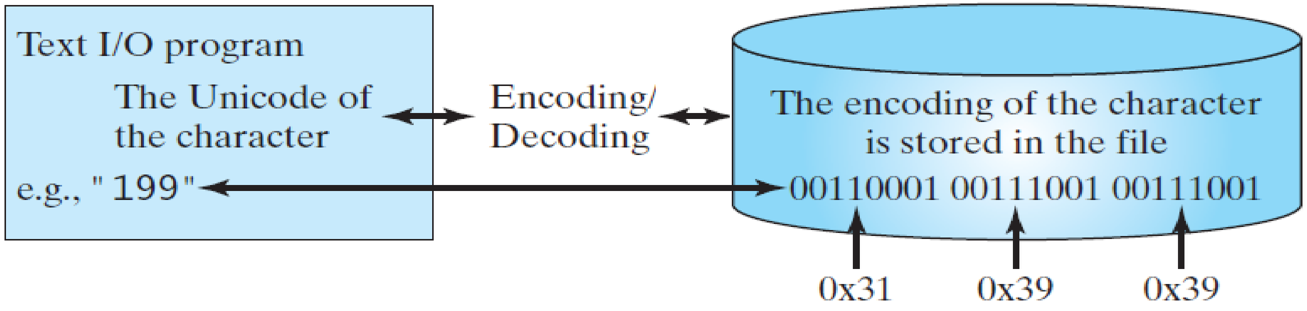
- Text is represented, e.g. using ASCII table (which is a mapping between characters/symbols and bit representations.). Examples:
  - Alphabet Characters: ‘a’ → 0110 0001 ‘A’ → 0100 0001
  - Digits represented as characters: ‘0’ → 0011 0000 ‘5’ → 0011 0101
  - Symbols represented as characters ‘$’ → 0010 0100 ‘+’ → 0010 1011
- Decimal values can be transformed to binary equivalent. Examples:
  - Number 5 as Byte type 1 byte 0000 0101
  - Number 5 as Short 2 bytes 0000 0000 0000 0101
  - Number 5 as Integer 4 bytes 0000 0000 0000 0000 0000 0000 0000 0101
- Image pixels are stored in binary by representing the color of the pixels. Example:
  - 8-bit Grayscale pixel Black: 0000 0000 White: 1111 1111 Grey: 0111 1111

Text I/O vs. Binary I/O

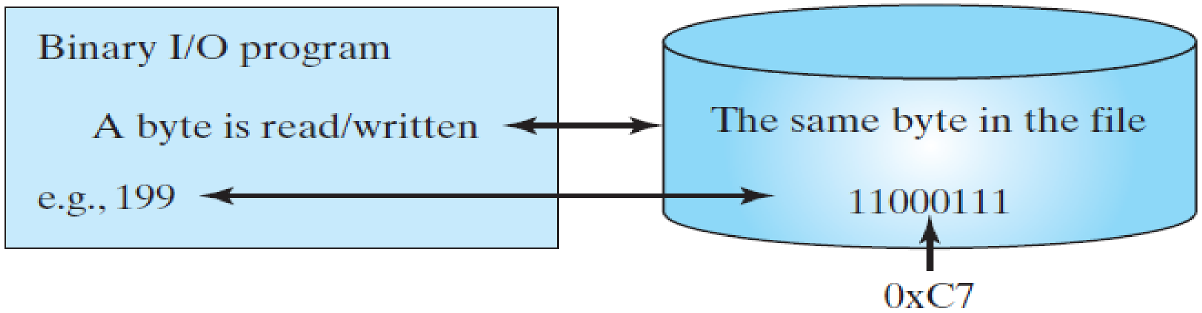
Text I/O is **built upon binary I/O** in order to **provide a level of abstraction** for character encoding and decoding.

Example: suppose you write **number 199** to a file…

- **Using text I/O**: Each **character** is written to the file using the **file's encoding scheme**. E.g., for character '1', it can be written in 8 bits if ASCII is used (0x31) or 16 bits for Unicode (0x0031).



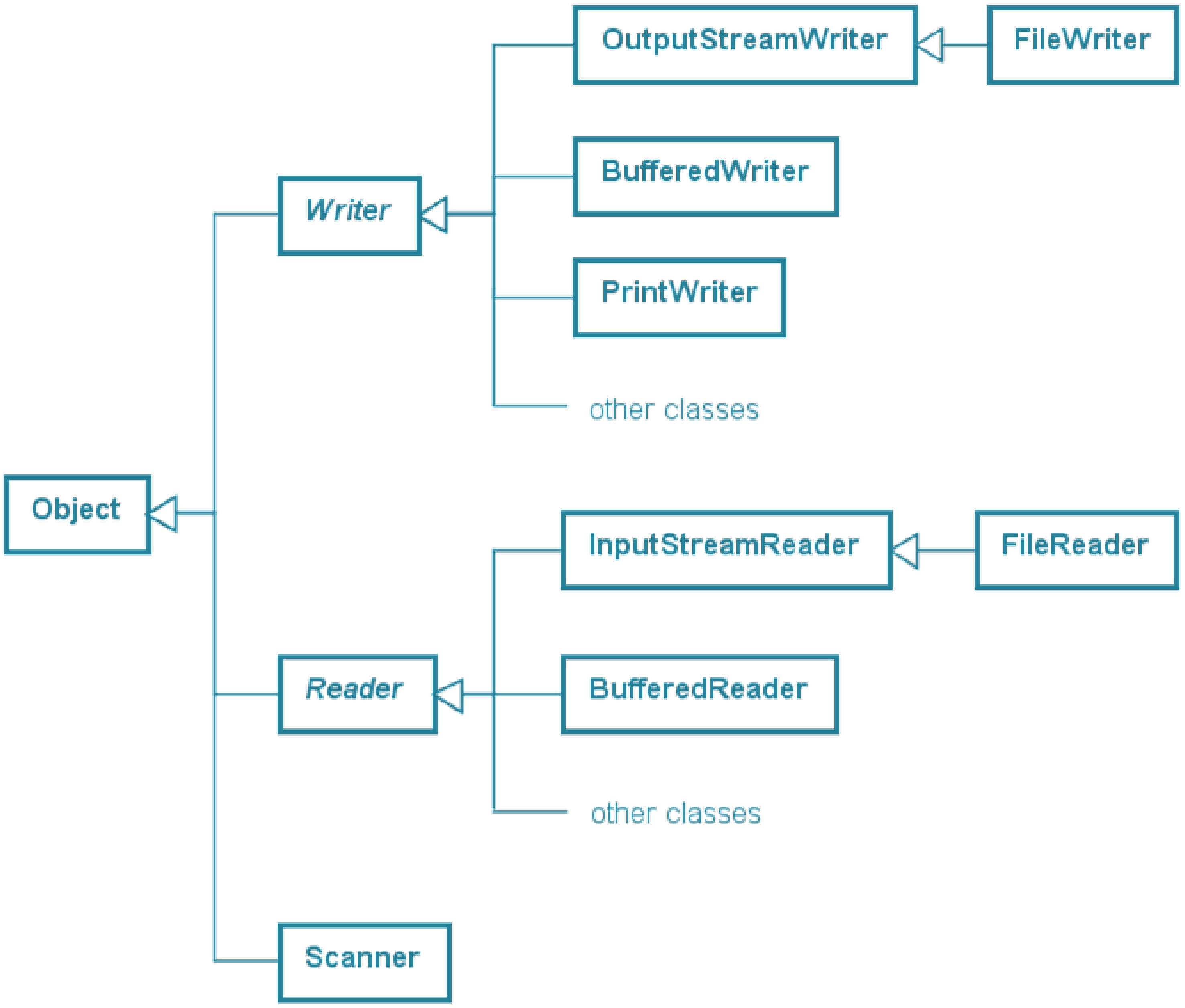
- **Using binary I/O**: the numeric binary equivalence of 199 is written.



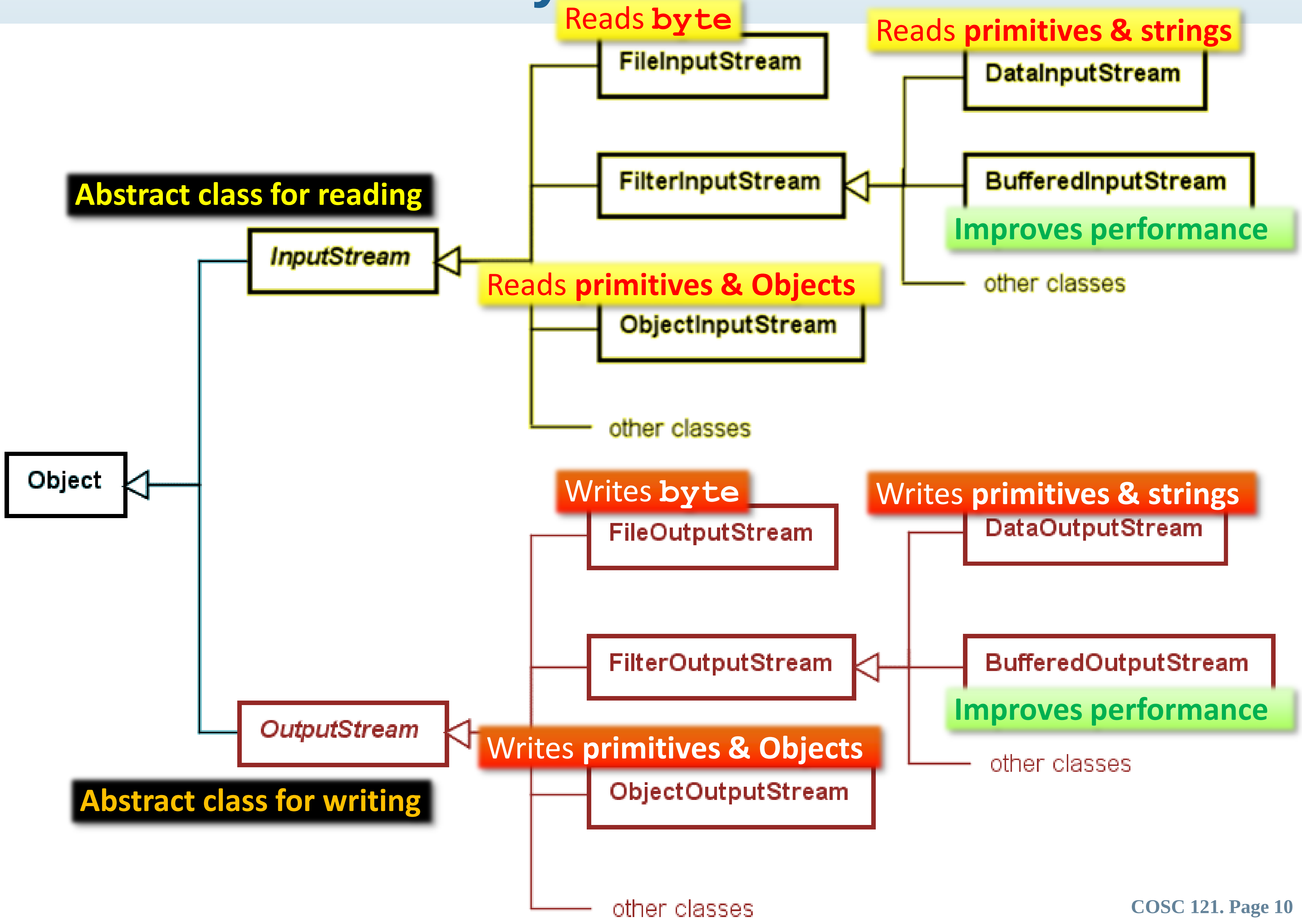
Text I/O vs. Binary I/O, cont.

	Text I/O	Binary I/O
Can handle..	Text files (readable by human) Example: .java, .txt	Binary files (not readable by human) Example: .class, .dat
Efficiency	Less efficient Involves encoding or decoding	**More efficient.** Doesn't involve encoding or decoding
Classes	Descendent of • **`Reader`** • **`Writer`**	Descendent of • **`InputStream`** • **`OutputStream`**

Remember: Text I/O Classes

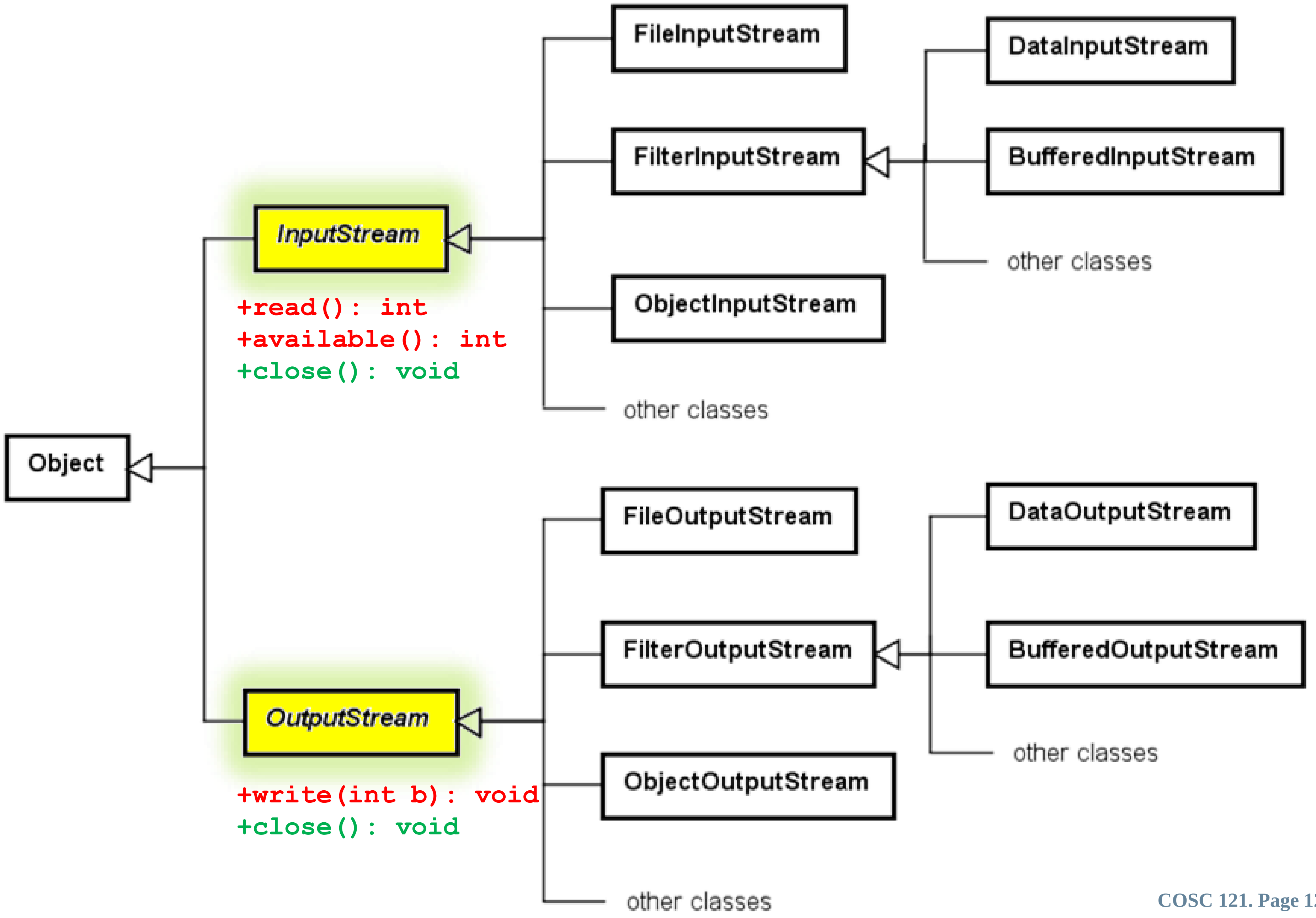


Binary I/O Classes

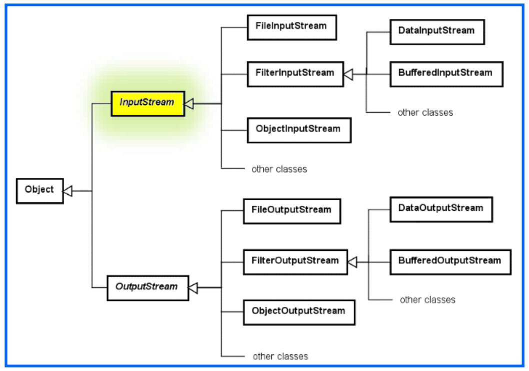


InputStream / OutputStream

Binary I/O Classes



InputStream



`+read(): int`

- reads **next byte** of data **as int** (0 to 255).
- **–1 is returned at end of stream**.

`+read(b: byte[]): int`

- Reads up to `b.length` bytes into **array `b`**.

`+read(b: byte[], off: int, len: int): int`

- stores read bytes into `b[off]`, `b[off+1]`,..., `b[off+len-1]`.

`+skip(n: long): long`

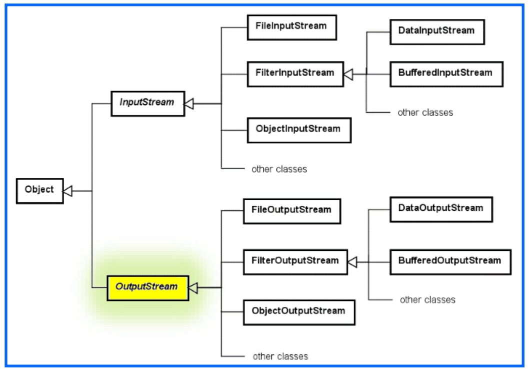
- Skips **`n` bytes** of data from this stream. actual # of bytes skipped returned.

`+available(): int`

- Returns the **number of bytes remaining in the input stream**.
- available()==0 indicates the **end of file (EOF)**

`+close(): void`

OutputStream



`+write(int b): void`

- writes **(byte) b** to this output stream.

`+write(b: byte[]): void`

- Writes all the **bytes** in array `b` to the output stream.

`+write(b: byte[], off: int, len: int): void`

- Writes `b[off]`, `b[off+1]`,…, `b[off+len-1]` into the output stream

`+flush(): void`

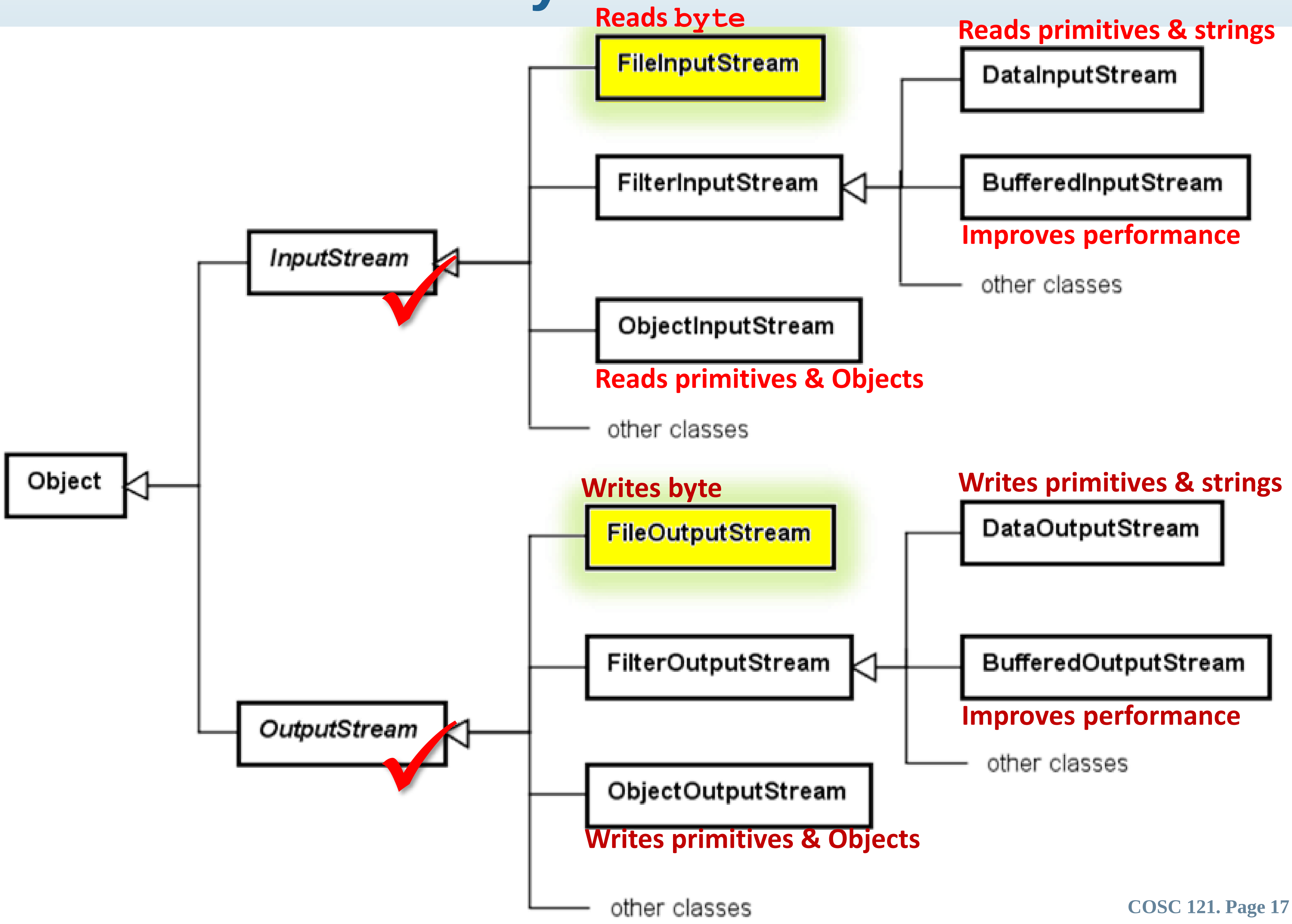
- Flushes this output stream and forces any buffered output **bytes** to be written out.

`+close(): void`

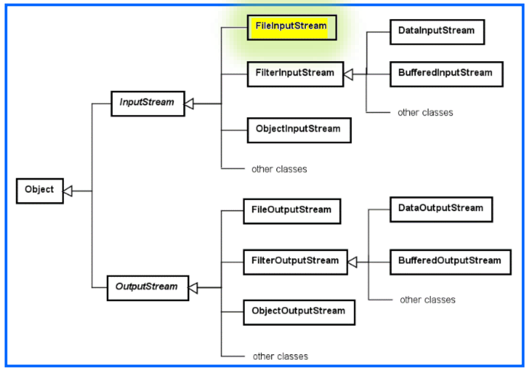
- Closes this output stream

FileInputStream / FileOutputStream

Binary I/O Classes



FileInputStream



Methods:

- All methods from *InputStream*

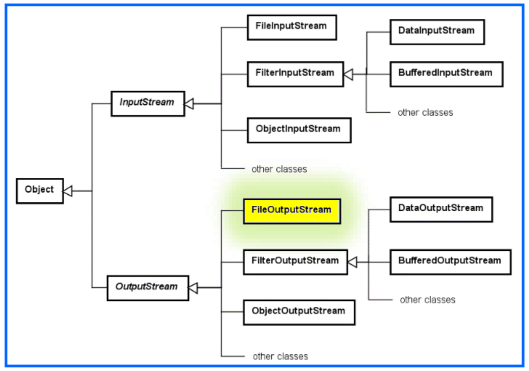
Constructors

- public FileInputStream(String filename)
- public FileInputStream(File file)

Exceptions

- Throws IOException

FileOutputStream



Methods:

- All methods from *OutputStream*

Constructors

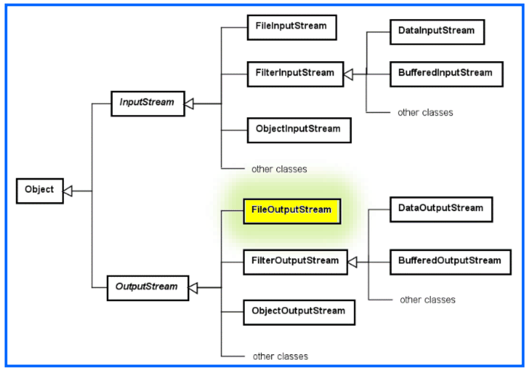
- public FileOutputStream(String filename)
- public FileOutputStream(File file)
  - If the file doesn't exist, a new file would be created.
  - If the file already exists, the current contents in the file are deleted.
- public FileOutputStream(String filename, **boolean append**)
- public FileOutputStream(File file, **boolean append**)
  - Used to retain the current content and append new data into the file.

Exceptions

- Throws IOException

Exercise1

Q1) Write Java code to write ten byte values from **1** to **10** to a file named **temp.dat**



```
FileOutputStream out = new FileOutputStream("temp.dat");
for (int i = 1; i <= 10; i++)
    out.write(i);
out.close();
```

More Exercises

Q2) Write Java code to display the values you wrote previously in Q1 into **temp.dat.**

Q3) Try Q2 again, but read values from any file other than temp.dat. For example:

- A text file: e.g. source.txt
- An image file: e.g. bird.jpg

What is the output?

Q4) Write a method that copies the contents from a **source file to a target file**, byte by byte, and displays the number of bytes copied.

Exercises Solutions (Hidden)

- Write Java code to write ten byte values from **1** to **10** to a file named **temp.dat** and reads them back from the file.

```
// WRITING
FileOutputStream out = new FileOutputStream("temp.dat");
for (int i = 1; i <= 10; i++)
    out.write(i);
out.close();
// READING
FileInputStream input = new FileInputStream("temp.dat");
int value;
while ((value = input.read()) != -1)
    System.out.print(value + " ");
```

- Write a method that copies the contents from a source file to a target file, byte by byte, and displays the number of bytes copied.

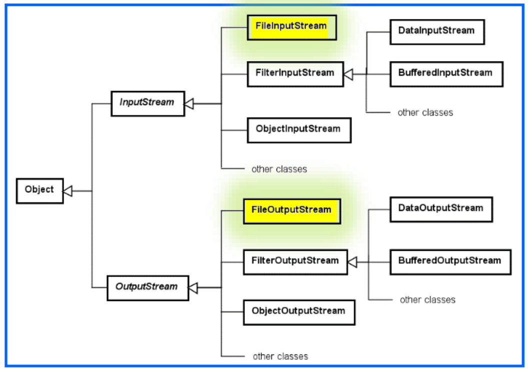
```
public void copyFile(String source, String target) throws IOException{
    FileInputStream in = new FileInputStream(source);
    FileOutputStream out = new FileOutputStream(target);
    int b, numberOfBytes = 0;
    while((b = in.read()) != -1){
        out.write(b);
        numberOfBytes++;
    }
    System.out.println(numberOfBytes + " bytes successfully copied.");
}
```

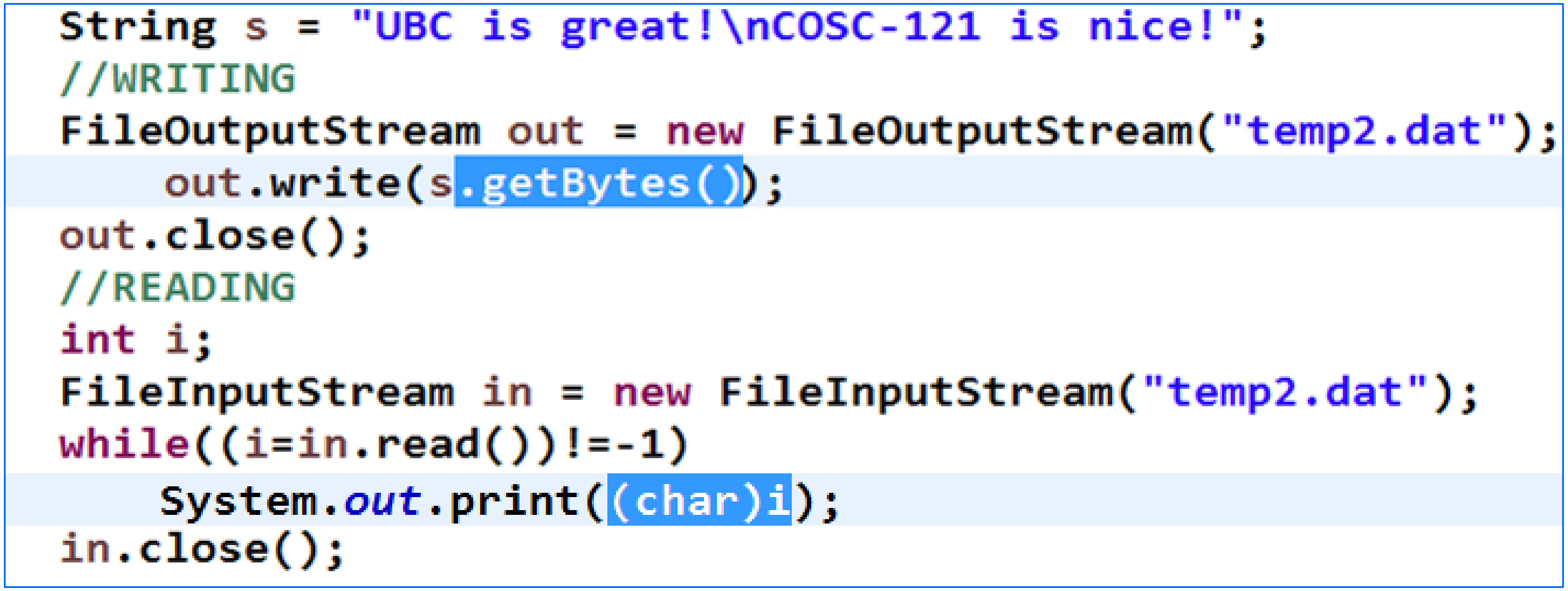
Exercise4 (hidden)

Write Java code to write the following text to a file and reads it back from the file.

UBC is great!

COSC-121 is nice!!





```java
String s = "UBC is great!\nCOSC-121 is nice!";
//WRITING
FileOutputStream out = new FileOutputStream("temp2.dat");
    out.write(s.getBytes());
out.close();
//READING
int i;
FileInputStream in = new FileInputStream("temp2.dat");
while((i=in.read())!=-1)
    System.out.print((char)i);
in.close();
```