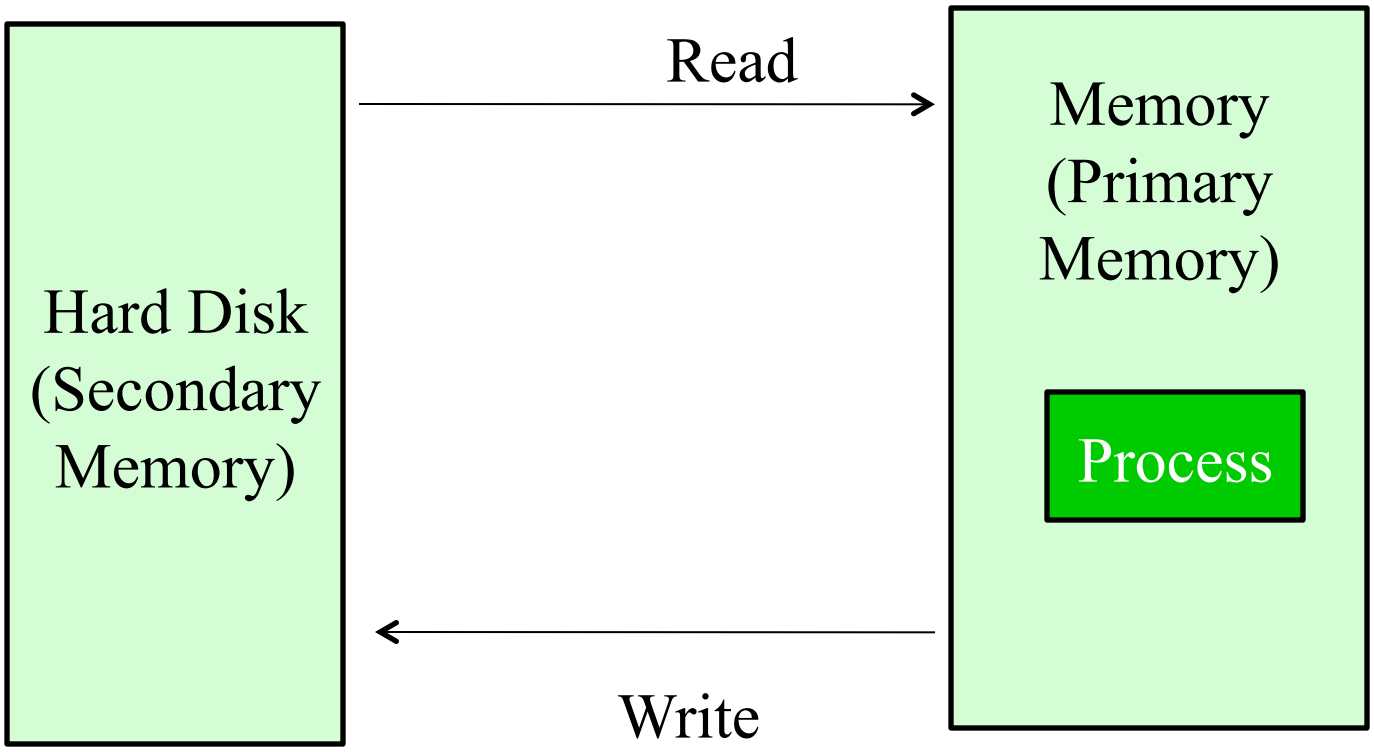
File I/O, Streams, Scanner

IST 256

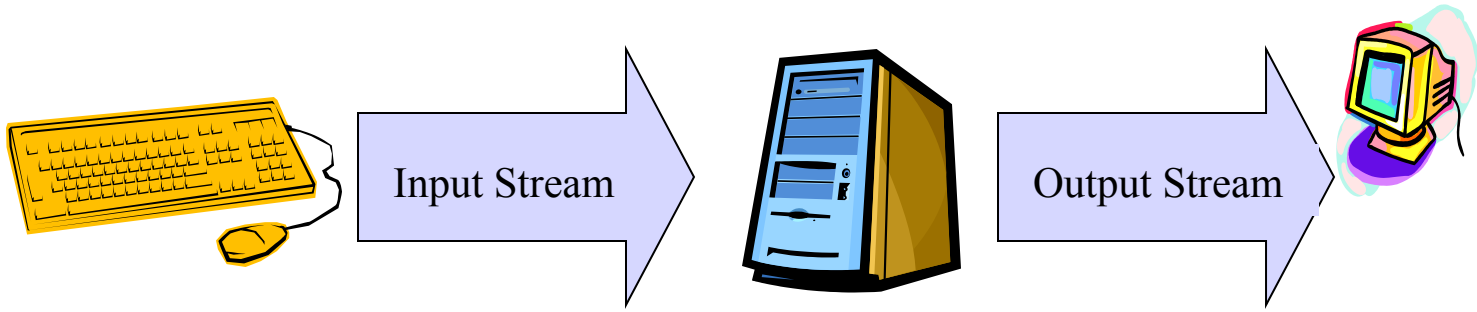
Application Programming for Information Systems

File I/O


Hard Disk
(Secondary
Memory)
Read
Write
Memory
(Primary
Memory)
Process

I/O Streams

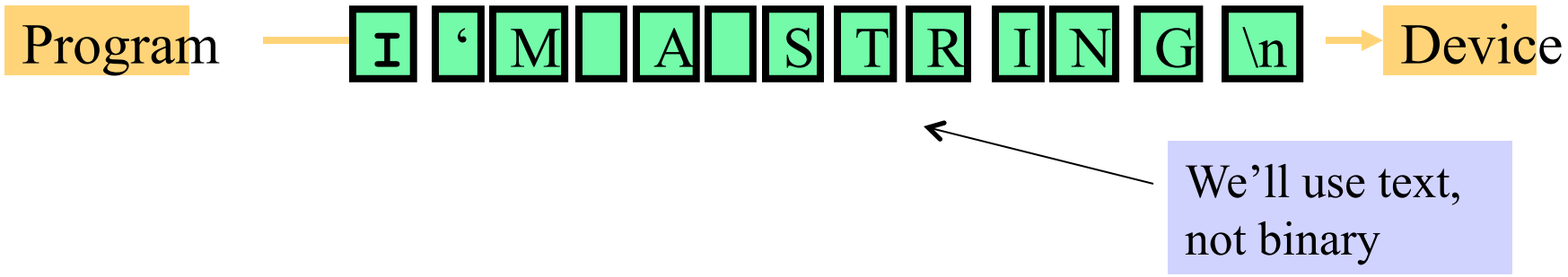
Default input/output streams



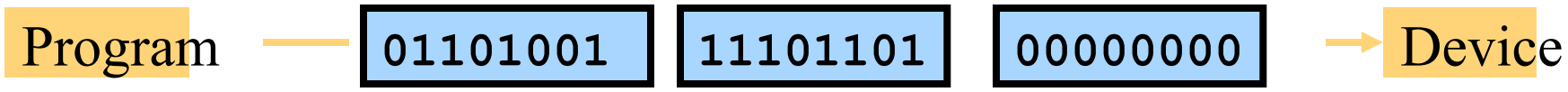
Java - Streams

JAVA provides 2 types of streams

Text streams - containing ‘characters‘



Binary Streams - containing 8 – bit information



Reading and Writing Files

Create a stream object and associate it with a disk-file

Give the stream object the desired functionality

while there is more information
read(write) next data from(to) the stream

close the stream

Reading Files

Code excerpt showing an Input Stream created as a BufferedReader and reading one line of text at a time:

```
import java.io.*;

BufferedReader ins
        = new BufferedReader(new FileReader("test.txt"));
while(ins.ready())
{
        String s = ins.readLine();
        System.out.println(s);
}
ins.close();
```

Place the code inside
try{…}
catch(IOException e){….}
wherever necessary

Writing Files

Code excerpt showing an Output Stream created as a BufferedWriter, writing a string

```
import java.io.*;

String message = “example text”;
BufferedWriter outs
        = new BufferedWriter(new FileWriter(“test_out.txt”));
outs.write(message);
outs.close();
```

Place the code inside
try{…}
catch(IOException e){….}
wherever necessary

Reading Tokens from a File

Code excerpt showing a Scanner wrapping an Input Stream, reading the next String token from the input

```
import java.io.*;
import java.util.Scanner;
```

```
BufferedReader ins
        = new BufferedReader(new FileReader("test.txt"));
Scanner scanner = new Scanner(ins);
while(scanner.hasNext())
{
        String message = scanner.next();
        System.out.println(message);
}
scanner.close();
```

Place the code inside
try{…}
catch(IOException e){….}
wherever necessary