

Reading Files



Colorado State University

Department of Computer Science

Slides Originally Created by Albert Lionelle (Albert.Lionelle@colostate.edu),
updated by Marcia Moraes (marcia.moraes@colostate.edu)

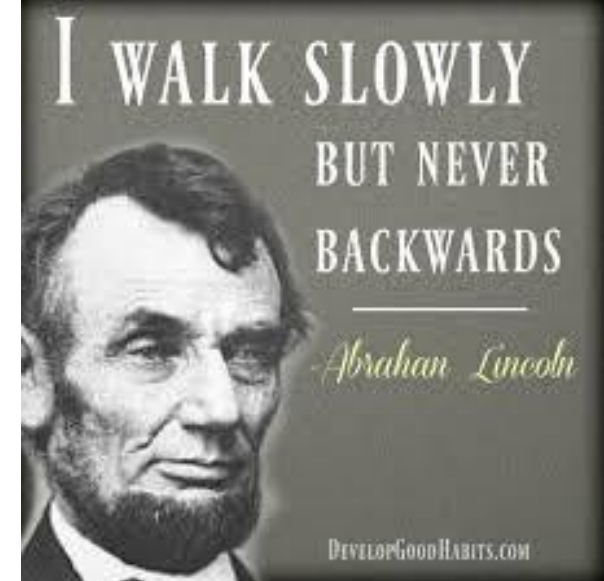
Announcements

TODO Reminders:

Readings are due **before** lecture

- Reading 14 (zybooks) – you should have already done that 😊
- Lab 09
- Reading 15 (zyBooks)
- Lab 10
- Reading 16 (zybooks)
- RPA 7

Keep practicing your RPAs in a spaced and mixed manner 😊



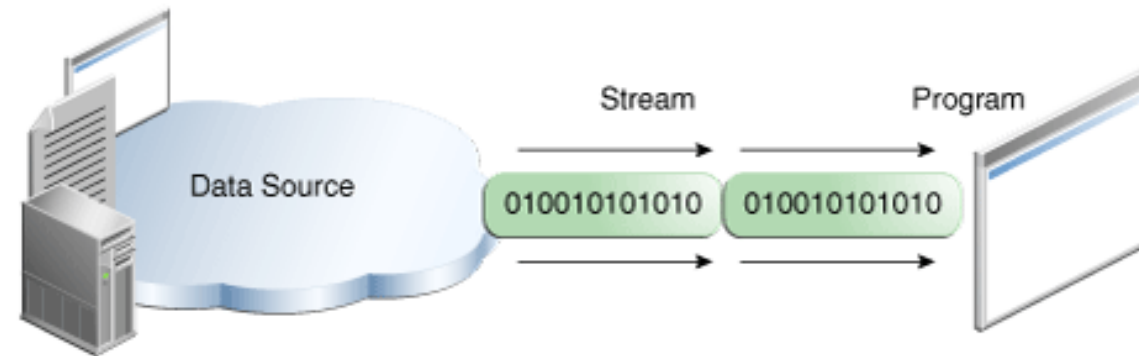
NEXT WEEK
Exam 2

Don't procrastinate
and catch up if you
need!

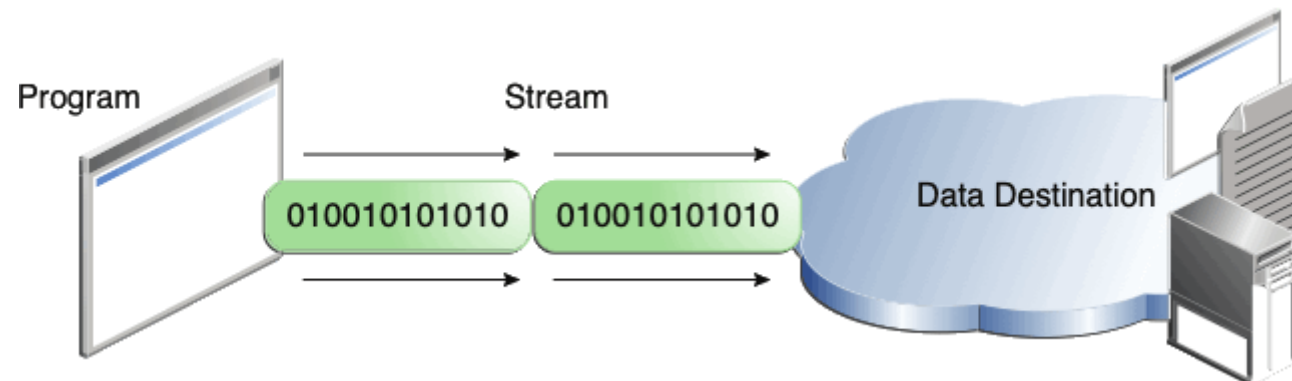
I/O Streams

<https://docs.oracle.com/javase/tutorial/essential/io/streams.html>

- A stream is a sequence of data.
- A program uses an input stream to read data from a source, one item at a time:



- A program uses an output stream to write data to a destination, one item at a time:



OutputStream

- `System.out` is a predefined `OutputStream` object reference that is associated with a system's standard output, usually a computer screen.
- The `print()` and `println()` methods are overloaded in order to support the various standard data types, such as `int`, `boolean`, `float`, etc., each method converting that data type to a sequence of characters.

InputStream

- `System.in` is an input byte stream
- When using an `InputStream`, a programmer must append the clause `throws IOException` when using the method `read()`
 - A `throws` clause tells the Java virtual machine that the corresponding method may exit unexpectedly due to an exception, which is an event that disrupts a program's execution.
- Instead of reading a byte stream, dealing with `IOException`, and after it converting the data to a `String` or other data types, we have been using the Wrapper class named `Scanner` 😊
 - `Scanner scnr = new Scanner(System.in);`
 - automatically scanning a sequence of bytes and converting those bytes to the desired data type depending on the type of method we use to read the data (`nextLine()`, `nextInt()`, etc.)

Scanner – reading from a String

- When we want to read something from the terminal we use:

```
Scanner scnr = new Scanner(System.in);
```

 - The parameter `System.in` indicates that we are reading from the terminal.
- Sometimes we may want to read something from a String, so instead of using `System.in` as a parameter we have a String as a parameter

```
String line = "Let me be that I am and seek not to alter me.\n-Thank you!";  
Scanner scanner = new Scanner(line);
```

Scanner – reading from a String

```
public class ScannerString {  
    public static void main(String args[]){  
        String line = "Let me be that I am and seek not to alter me.\n-Thank you!";  
        Scanner scanner = new Scanner(line);  
        int tokenCounter = 0;  
        while(scanner.hasNext()) {  
            tokenCounter++;  
            System.out.println(scanner.next());  
        }  
        System.out.println(tokenCounter);  
    }  
}
```

.hasNext() – returns true if the scanner has another token in its input

.next() – finds and returns the next complete token from this scanner.

Each token is separated by a delimiter, default delimiter is whitespace.

Scanner – What is printed?

```
public class ScannerString {  
    public static void main(String args[]){  
        String line = "Let me be that I am and seek not to alter me.\n-Thank you!";  
        Scanner scanner = new Scanner(line);  
        int tokenCounter = 0;  
        while(scanner.hasNext()) {  
            tokenCounter++;  
            System.out.println(scanner.next());  
        }  
        System.out.println(tokenCounter);  
    }  
}
```

Let
me
be
that
I
am
and
seek
not
to
alter
me.
-Thank
you!
14

Scanner – reading from a String

```
import java.util.Scanner;
```

```
public class ScannerString {  
    public static void main(String args[]){  
        String line = "Let me be that I am and seek not to alter me.\n-Thank you!";  
        Scanner scanner = new Scanner(line);  
        scanner = new Scanner(line);  
        int lineCounter = 0;  
        while(scanner.hasNext()) {  
            lineCounter++;  
            System.out.println(scanner.nextLine());  
        }  
        System.out.println(lineCounter);  
    }  
}
```

.hasNext() – returns true if the scanner has another token in its input

.nextLine() – advances this scanner past the current line and returns the input that was skipped

Scanner – What is printed?

```
import java.util.Scanner;

public class ScannerString {
    public static void main(String args[]){
        String line = "Let me be that I am and seek not to alter me.\n-Thank you!";
        Scanner scanner = new Scanner(line);
        scanner = new Scanner(line);
        int lineCounter = 0;
        while(scanner.hasNext()) {
            lineCounter++;
            System.out.println(scanner.nextLine());
        }
        System.out.println(lineCounter);
    }
}
```

```
Let me be that I am and seek not to alter me.
-Thank you!
2
```

Scanner – reading from a String

```
import java.util.Scanner;
```

```
public class ScannerString {  
    public static void main(String args[]){  
        String line = "Let me be that I am and seek not to alter me.\n-Thank you!";  
        Scanner scanner = new Scanner(line);  
        scanner.useDelimiter("-");  
        int otherCounter = 0;  
        while(scanner.hasNext()) {  
            otherCounter++;  
            System.out.println(scanner.next());  
        }  
    }  
}
```

.useDelimiter(String) - Sets this scanner's delimiting pattern to a pattern constructed from the specified String

What is printed now?

```
Let me be that I am and seek not to alter me.
```

```
Thank you!
```

Scanner – what is printed?

```
import java.util.Scanner;
```

```
public class StringLocation {  
    public static void main(String args[]){  
        String location = "Fort Collins,40°35'6.9288\"N,105°5'3.9084\"W";  
        Scanner locScan = new Scanner(location);  
        locScan.useDelimiter(",");  
        String city = locScan.next();  
        String lat = locScan.next();  
        String lon = locScan.next();  
        System.out.println(city);  
        System.out.println(lat);  
        System.out.println(lon);  
    }  
}
```

Fort Collins

40°35'6.9288"N

105°5'3.9084"W

Scanner – different uses

- When we want to read something from the terminal we use:

```
Scanner scnr = new Scanner(System.in);
```

- The parameter `System.in` indicates that we are reading from the terminal.

- When we want to read something from a String we use:

```
String line = "Let me be that I am and seek not to alter me.\n-Thank you!";
```

```
Scanner scanner = new Scanner(line);
```

- What happens if instead of reading from the terminal or a String, we want to read from a file?
 - We need to use the `File` class!
 - And pass an object of `File` as a parameter instead of `System.in` or a String when we construct a Scanner object

Input Streams

- `System.in`
- `Strings`
- `File`
- `FileInputStream`

Scanner – reading from a File

Basic Structure

```
import java.io.File;
import java.io.IOException;
import java.util.Scanner;
public class FileReadingBasicStructure {
    public static void readFile(String fileName){
        try {
            Scanner fileIn = new Scanner(new File(fileName));
            // now the file is in a scanner, and looping matters!
            while(fileIn.hasNext()) { // could also use hasNextLine()
                System.out.println(fileIn.nextLine());
            }
        } catch(IOException ex) {
            // there was an error finding the file or reading the file!
            // so how do you hand that? For now, we just say that and end the
            //program
            System.err.print("Error reading file!");
            ex.printStackTrace();
        }
    }

    public static void main(String args[]){
        readFile("file.txt");
    }
}
```

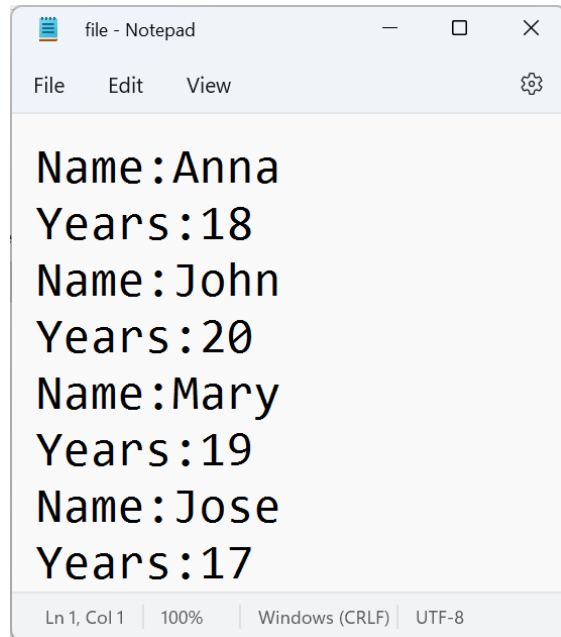
Creates a Scanner from a file

While still has elements

Reads the line

Test for the exception

What is printed?

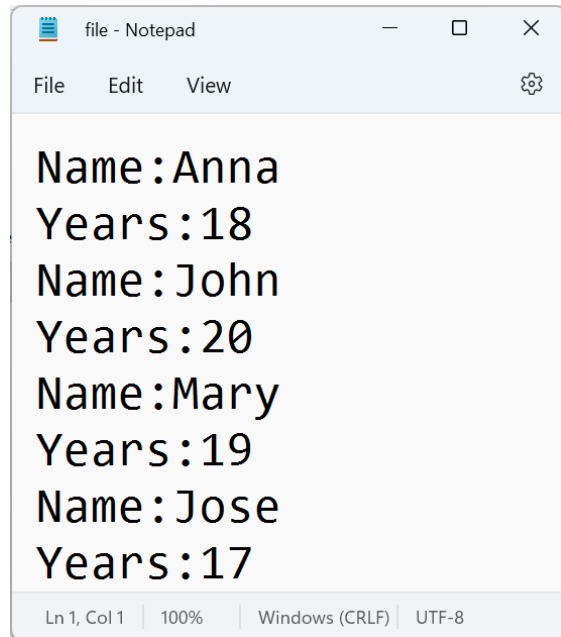


```
file - Notepad
File Edit View
Name:Anna
Years:18
Name:John
Years:20
Name:Mary
Years:19
Name:Jose
Years:17
Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8
```

```
import java.io.File;
import java.io.IOException;
import java.util.Scanner;
public class FileReadingBasicStructure {
    public static void readFile(String fileName){
        try {
            Scanner fileIn = new Scanner(new File(fileName));
            // now the file is in a scanner, and looping matters!
            while(fileIn.hasNext()) { // could also use hasNextLine()
                System.out.println(fileIn.nextLine());
            }
        } catch(IOException ex) {
            // there was an error finding the file or reading the file!
            // so how do you hand that? For now, we just say that and end the
            //program
            System.err.print("Error reading file!");
            ex.printStackTrace();
        }
    }
    public static void main(String args[]){
        readFile("file.txt");
    }
}
```

Name:Anna
Years:18
Name:John
Years:20
Name:Mary
Years:19
Name:Jose
Years:17

Reading from a File and Creating Objects to add into an ArrayList



A screenshot of a Notepad window titled "file - Notepad". The window contains a text file with the following content:

```
Name:Anna
Years:18
Name:John
Years:20
Name:Mary
Years:19
Name:Jose
Years:17
```

The status bar at the bottom indicates "Ln 1, Col 1", "100%", "Windows (CRLF)", and "UTF-8".

```
public class Person {
    private String name;
    private int yearsOld;
    public Person(String name){
        setName(name);
        setYearsOld(0);
    }
    public Person(String name, int years){
        setName(name);
        setYearsOld(years);
    }
    public void setName(String name){
        this.name = name;
    }
    public void setYearsOld(int years){
        yearsOld = years;
    }
    public String getName(){
        return name;
    }
    public int getYearsOld(){
        return yearsOld;
    }
    public String toString(){
        return "Name: " + name + " Years Old: " + yearsOld;
    }
}
```


Reading from a File and Creating Objects to add into an ArrayList

```
file - Notepad
File Edit View
Name:Anna
Years:18
Name:John
Years:20
Name:Mary
Years:19
Name:Jose
Years:17
Ln 1, Col 1 100% Windows (CR LF) UTF-8
```

```
public static ArrayList<Person> readFilePerson(String fileName){
    ArrayList<Person> list = new ArrayList<>();
    try {
        Scanner fileIn = new Scanner(new File(fileName));
        String name = "";
        int years = 0;
        int index = 0;
        int lineNumber = 0;
        while(fileIn.hasNext()) {
            String line = fileIn.nextLine();
            if(line.contains("Name")){
                index = line.indexOf(":");
                name = line.substring(index+1, line.length());
                lineNumber++;
            }
            else if (line.contains("Years")){
                index = line.indexOf(":");
                years = Integer.parseInt(line.substring(index+1, line.length()));
                lineNumber++;
            }
        }
    }
}
```

```
        if(lineNumber == 2){
            Person p = new Person(name, years);
            list.add(p);
            lineNumber = 0;
        }
    }
} catch(IOException ex) {
    System.err.print("Error reading file!");
    ex.printStackTrace();
}
return list;
}

public static void main(String args[]){
    ArrayList<Person> lst = readFilePerson("file.txt");
    System.out.println(lst);
}
```