

LEC 03

CSE 122

File I/O – Hybrid processing and Printing

Questions during Class?

Raise hand or send here

sli.do #cse122



BEFORE WE START


Talk to your neighbors:
Any weekend plans?

Music: [122 24sp Lecture Tunes](#) 

Instructors: Miya Natsuhara and Kasey Champion

TAs:	Ayush	Kyle	Colin	Chaafen
	Poojitha	Jacob	Ronald	Smriti
	Chloe	Atharva	Saivi	Ambika
	Ailsa	Rucha	Shivani	Elizabeth
	Jasmine	Megana	Kavya	Aishah
	Lucas	Eesha	Steven	Minh
	Logan	Zane	Ken	Katharine


Lecture Outline

- **Announcements/Reminders** 
- Refresh Last Time
- Scanners with Strings
 - Hybrid Approach & Files
- Using Printstream for File Output
- Example

Announcements

- Programming Assignment 0 (P0) out later today!
 - Due next Thursday, April 11th!
 - Focused on File I/O
- Creative Project 0 (C0) was due last night. How'd it go?
 - Expect grades back about a week after the assignment was due
 - Joined class late? Use Resubmission Cycle 0 to submit it!

Lecture Outline

- Announcements/Reminders
- **Refresh Last Time** 
- Scanners with Strings
 - Hybrid Approach & Files
- Using Printstream for File Output
- Example

(Last Time) Scanner/File for input

Scanner is defined in the
java.util package

```
import java.util.*;
```

File is defined in the
java.io package

```
import java.io.*;
```

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

```
File newFile = new File("example.txt");
```

```
Scanner fileScan = new Scanner(newFile);
```

Scanner Methods	Description
nextInt()	Reads the next token from the user as an <code>int</code> and returns it
nextDouble()	Reads the next token from the user as a <code>double</code> and returns it
next()	Reads the next token from the user as a <code>String</code> and returns it
nextLine()	Reads an <i>entire line</i> from the user as a <code>String</code> and returns it
hasNextInt()	Returns <code>true</code> if the next token can be read as an <code>int</code> , <code>false</code> otherwise
hasNextDouble()	Returns <code>true</code> if the next token can be read as a <code>double</code> , <code>false</code> otherwise
hasNext()	Returns <code>true</code> if there is another token of input to be read in, <code>false</code> otherwise
hasNextLine()	Returns <code>true</code> if there is another line of input to be read in, <code>false</code> otherwise


(PCM) Typical Line-Processing Pattern

```
while (scan.hasNextLine()) {  
    String nextLine = scan.nextLine();  
    // do something with nextLine  
}
```

(PCM) Typical Token-Processing Pattern

```
while (scan.hasNext__()) {  
    __ nextToken = scan.next__();  
    // do something with nextToken  
}
```

Lecture Outline

- Announcements/Reminders
- Refresh Last Time
- **Scanners with Strings** 
 - Hybrid Approach & Files
- Using Printstream for File Output
- Example

(PCM) Scanners with Strings

```
String str = "A quick, brown fox";
```

```
Scanner stringScan = new Scanner(str);  
while (stringScan.hasNext__()) {  
    __ nextToken = stringScan.next__();  
    // do something with nextToken  
}
```

(PCM) Scanners with Strings

```
String str = "A quick, brown fox";
```

```
Scanner stringScan = new Scanner(str);  
while (stringScan.hasNext()) {  
    String nextToken = stringScan.next();  
    System.out.println(nextToken);  
}
```

(PCM) Scanners with Strings

```
String str = "A quick, brown fox";
```

```
Scanner stringScan = new Scanner(str);  
while (stringScan.hasNext()) {  
    String nextToken = stringScan.next();  
    System.out.println(nextToken);  
}
```

A

(PCM) Scanners with Strings

```
String str = "A quick, brown fox";
```

```
Scanner stringScan = new Scanner(str);  
while (stringScan.hasNext()) {  
    String nextToken = stringScan.next();  
    System.out.println(nextToken);  
}
```

quick,

(PCM) Scanners with Strings

```
String str = "A quick, brown fox";
```

```
Scanner stringScan = new Scanner(str);  
while (stringScan.hasNext()) {  
    String nextToken = stringScan.next();  
    System.out.println(nextToken);  
}
```

brown


(PCM) Scanners with Strings

```
String str = "A quick, brown fox";
```

```
Scanner stringScan = new Scanner(str);  
while (stringScan.hasNext()) {  
    String nextToken = stringScan.next();  
    System.out.println(nextToken);  
}
```

fox

Lecture Outline

- Announcements/Reminders
- Refresh Last Time
- Scanners with Strings
 - **Hybrid Approach & Files** 
- Using Printstream for File Output
- Example

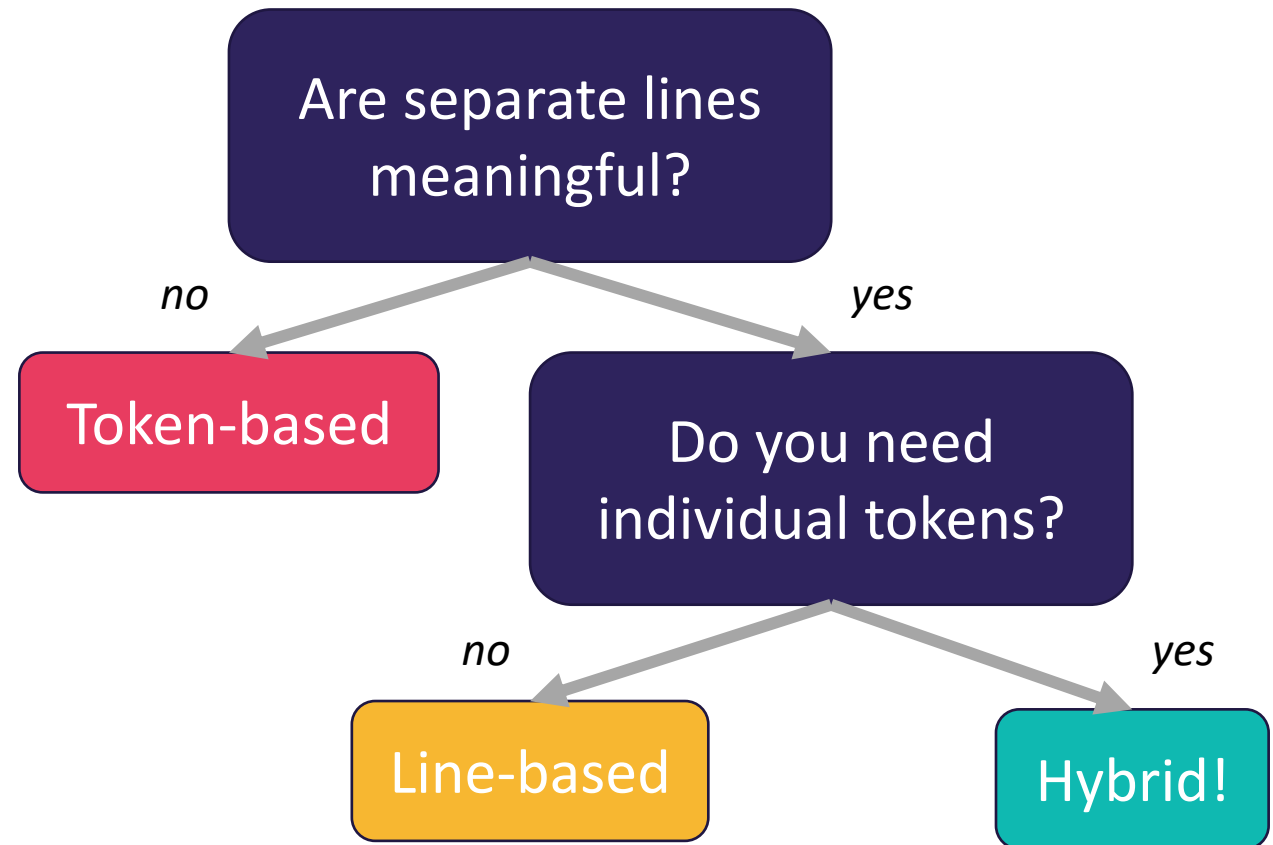
(PCM) Typical Hybrid Pattern

```
File newFile = new File("in.txt");
Scanner fileScan = new Scanner(newFile);
while (fileScan.hasNextLine()) {
    String line = fileScan.nextLine();

    Scanner lineScan = new Scanner(line);
    while (lineScan.hasNext__()) {
        __ nextToken = lineScan.next__();
        // do something with nextToken
    }
}
```


(PCM) Token vs. Line vs. Hybrid?

- We now know 3 different ways to use Files!
 - Although this gives us flexibility – it can sometimes get confusing
- Feel free to use the following diagram to help!

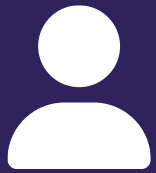


(PCM) Scanning Numeric Data

On Wednesday, we primarily used `String`-based Scanner methods to read input from a file. Let's work with some numeric data now!

We're going to make more use of

- `hasNextInt()`
- `hasNextDouble()`
- `nextInt()`
- `nextDouble()`
- Assumptions about our file's format!



Practice : Think

[sli.do](#)[#cse122](#)

What would be the result of executing the following code?

```
Scanner fileScan = new Scanner(new File("data.txt"));
while (fileScan.hasNextLine()) {
    String line = fileScan.nextLine();
    Scanner lineScan = new Scanner(line);
    double min = lineScan.nextDouble();
    double max = min;
    while (lineScan.hasNextDouble()) {
        double nextNum = lineScan.nextDouble();
        min = Math.min(min, nextNum);
        max = Math.max(max, nextNum);
    }
    System.out.println("Max: " + max + ", Min: " + min);
}
```

data.txt

```
2.3 9.2
    17   0.73
-1.5000
```

- A) Max: 9.2, Min: 2.3
Max: 17.0, Min: 0.73
Max: -1.5, Min: -1.5
- B) Max: 9.2, Min: -1.5
- C) Max: 9.2, Min: 2.3
Max: 17, Min: 0.73
Max: 0.0, Min: -1.5
- D) Error



Practice : Pair



sli.do #cse122

What would be the result of executing the following code?


```
Scanner fileScan = new Scanner(new File("data.txt"));
while (fileScan.hasNextLine()) {
    String line = fileScan.nextLine();
    Scanner lineScan = new Scanner(line);
    double min = lineScan.nextDouble();
    double max = min;
    while (lineScan.hasNextDouble()) {
        double nextNum = lineScan.nextDouble();
        min = Math.min(min, nextNum);
        max = Math.max(max, nextNum);
    }
    System.out.println("Max: " + max + ", Min: " + min);
}
```

data.txt

```
2.3 9.2
    17   0.73
-1.5000
```

- A) Max: 9.2, Min: 2.3
Max: 17.0, Min: 0.73
Max: -1.5, Min: -1.5
- B) Max: 9.2, Min: -1.5
- C) Max: 9.2, Min: 2.3
Max: 17, Min: 0.73
Max: 0.0, Min: -1.5
- D) Error

Lecture Outline

- Announcements/Reminders
- Refresh Last Time
- Scanners with Strings
 - Hybrid Approach & Files
- **Using Printstream for File Output** 
- Example

(PCM) PrintStreams for output

PrintStream is defined
in the `java.io` package

```
import java.io.*;
```

```
File outputFile = new File("out.txt");  
PrintStream output = new PrintStream(outputFile);
```


PrintStream Methods	Description
<code>print(...)</code>	Prints the given value to the set output location.
<code>println(...)</code>	Prints the given value to the set output location, and then terminates the line.

```
System.out.print("Hello, world! ");  
System.out.println("#1 Bee Movie fan!");
```

```
output.print("Hello, world! ");  
output.println("#1 Bee Movie fan!");
```

Hello, world! #1 Bee Movie fan!

Lecture Outline

- Announcements/Reminders
- Refresh Last Time
- Scanners with Strings
 - Hybrid Approach & Files
- Using Printstream for File Output
- **Example** 

Movie Ratings

In this program, we'll be examining and altering data from a file of IMDB ratings for popular U.S. movies. This will happen through 3 major user-entered commands:

(F)ind movie, **(A)dd** a rating, and **(S)ave**.

```
small.tsv
```

```
5
Title    Average Total
Bee_Movie 6.1 176805
Barbie   6.9 455488
Oppenheimer 8.4      588723
Poor_Things 8.5      20542
Spider-Man:_Across_the_Spider-Verse 8.6      329200
```


Movie Ratings

```
Welcome to the CSE 122 Movie Rating Program!  
Loaded 5 movies from small.tsv!
```

```
Menu: (F)ind movie, (A)dd rating, (S)ave, (Q)uit  
Enter your choice: F  
What's the name of the movie? Bee Movie  
Movie Bee_Movie found!  
Average Rating: 6.1  
Total Ratings: 176805
```

```
Menu: (F)ind movie, (A)dd rating, (S)ave, (Q)uit  
Enter your choice: A  
What movie would you like to add your rating to? Bee Movie  
And what rating would you like to give? 100000
```

```
Menu: (F)ind movie, (A)dd rating, (S)ave, (Q)uit  
Enter your choice: f  
What's the name of the movie? Bee Movie  
Movie Bee_Movie found!  
Average Rating: 6.7  
Total Ratings: 176806
```

```
Menu: (F)ind movie, (A)dd rating, (S)ave, (Q)uit  
Enter your choice: S  
What's the name of the file you'd like to save to? out.txt
```

```
Menu: (F)ind movie, (A)dd rating, (S)ave, (Q)uit  
Enter your choice: q  
Thank you for using this program! Bye!
```

small.tsv

```
5  
Title Average Total  
Bee_Movie 6.1 176805  
Barbie 6.9 455488  
Oppenheimer 8.4 588723  
Poor_Things 8.5 20542  
Spider-Man:_Across_the_Spider-Verse 8.6 329200
```



```
[Bee_Movie, Barbie, Oppenheimer, Poor_Things, Spider-Man...]  
[6.1, 6.9, 8.4, 8.5, 8.6]  
[176805, 455488, 588723, 20542, 329200]
```

Movie Ratings: Development Strategy

1. Fill in the main method with a loop that calls a method to read the data in from the .tsv file and allows the user to select between the different options (find, add, save, quit)
2. Implement a method to load the movie rating data from the file and populate the appropriate arrays
3. Implement a method that allows users to find the rating for a movie
4. Implement a method that allows users to add a rating for a movie
5. Implement a method that allows users to save the movie ratings information to a file