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 24wi Lecture Tunes

Instructors Miya Natsuhara and Joe Spaniac

TAs

Ailsa	Chaafen	Helena	Megana	Sahej
Alexander	Chloe	Jessie	Mia	Shivani
Ambika	Claire	Katharine	Minh	Smriti
Andy	Colin	Kavya	Nicolas	Steven
Arkita	Colton	Ken	Poojitha	Vinay
Atharva	Connor	Kyle	Rohini	Zane
Autumn	Elizabeth	Logan	Ronald	
Ayush	Hannah	Marcus	Rucha	

Lecture Outline

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

Announcements

- Quiz 0 is next Thursday, January 18th!
  - Topics: Java Review, Functional Decomposition, File I/O Part 1 (Scanners, Files, token/line-based processing)
  - Taken in your *registered* quiz section
  - More details on quiz logistics in Ed announcement posted soon!
- Programming Assignment 0 (P0) out later today!
  - Due next Thursday, January 18th! (Note: same day as quiz, plan accordingly)
- Creative Project 0 (C0) was due last night. How'd it go?
  - Expect grades back a week after the assignment was due
  - Joined class late? Use Resubmission Cycle 0 to submit it!
- Monday Holiday means IPL closed January 15th!

Lecture Outline

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

(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 `int` and returns it
`nextDouble()`	Reads the next token from the user as a `double` and returns it
`next()`	Reads the next token from the user as a `String` and returns it
`nextLine()`	Reads an *entire line* from the user as a `String` and returns it
`hasNextInt()`	Returns `true` if the next token can be read as an `int`, `false` otherwise
`hasNextDouble()`	Returns `true` if the next token can be read as a `double`, `false` otherwise
`hasNext()`	Returns `true` if there is another token of input to be read in, `false` otherwise
`hasNextLine()`	Returns `true` if there is another line of input to be read in, `false` otherwise

(Last Time) Typical Token-Processing Patterns

Console Input:

```
Scanner console = new Scanner(System.in);
while (console.hasNext__()) {
    __ nextToken = console.next__();
    // do something with nextToken
}
```

File Input:

```
File newFile = new File("newFile.txt");
Scanner fileScan = new Scanner(newFile);
while (fileScan.hasNext__()) {
    __ nextToken = fileScan.next__();
    // do something with nextToken
}
```

Notice any similarities between the two?

(Last Time) Typical Line-Processing Patterns

Console Input:

```
Scanner console = new Scanner(System.in);
while (console.hasNextLine()) {
    String line = console.nextLine();
    // do something with line
}
```

File Input:

```
File newFile = new File("newFile.txt");
Scanner fileScan = new Scanner(newFile);
while (fileScan.hasNextLine()) {
    String line = fileScan.nextLine();
    // do something with line
}
```

Notice any similarities between the two?

Lecture Outline

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

(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

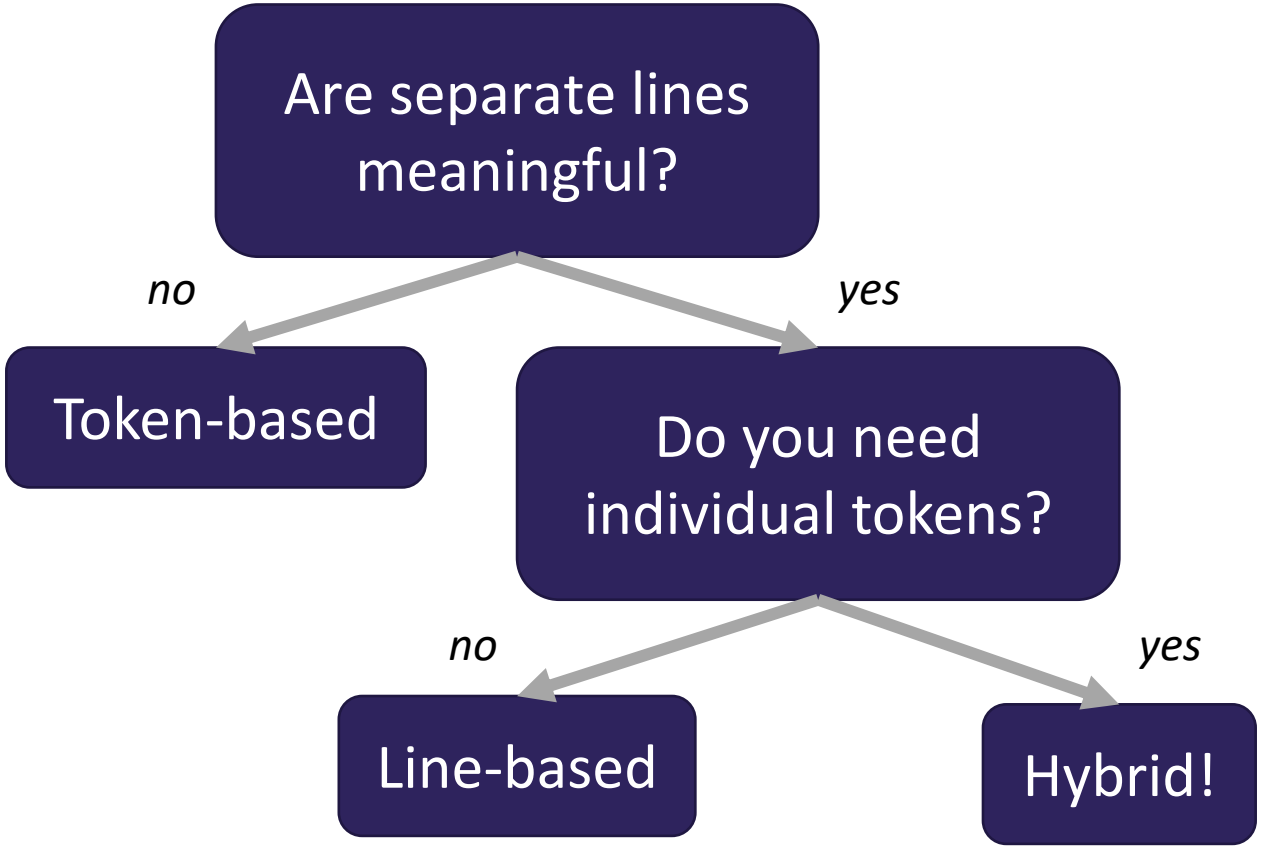
(PCM) Typical Hybrid Pattern

```
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 the result of running FindMaxAndMin with data2.txt as input?

data2.txt

```
2.3  9.2
     17          0.73
   3.14   4.83   -1.5000
```

A) Max: 9.2, Min: -1.5

B) Max: 9.2, Min: 2.3
Max: 17, Min: 0.73
Max: 4.83, Min: -1.5

C) Max: 9.2, Min: 2.3
Max: 17, Min: 0.0
Max: 4.83, Min: -1.5

D) Error

Practice : Pair

sli.do #cse122

What would the result of running FindMaxAndMin with data2.txt as input?

data2.txt

```
2.3  9.2
     17          0.73
  3.14   4.83   -1.5000
```

A) Max: 9.2, Min: -1.5

B) Max: 9.2, Min: 2.3
Max: 17, Min: 0.73
Max: 4.83, Min: -1.5

C) Max: 9.2, Min: 2.3
Max: 17, Min: 0.0
Max: 4.83, Min: -1.5

D) Error

Practice : Think

sli.do #cse122

What would the result of running FindMaxAndMin with data3.txt as input?

data3.txt

```
2.3  9.2
     17          0.73
```

A) Max: 9.2, Min: 0.73

B) Max: 9.2, Min: 2.3
Max: 17, Min: 0.73

C) Max: 9.2, Min: 2.3
Max: 17, Min: 0.0

D) Error

Practice : Pair

sli.do #cse122

What would the result of running FindMaxAndMin with data3.txt as input?

data3.txt

```
2.3  9.2
     17          0.73
```

A) Max: 9.2, Min: 0.73

B) Max: 9.2, Min: 2.3
Max: 17, Min: 0.73

C) Max: 9.2, Min: 2.3
Max: 17, Min: 0.0

D) Error

Lecture Outline

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

(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
print(…)	Prints the given value to the set output location.
println(…)	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!