

LEC 02

CSE 122

File I/O

Questions during Class?
Raise hand or send here

sli.do #cse122



BEFORE WE START

*Talk to your neighbors:**What is your current TV/Youtube/video
obsession?*

Respond on sli.do!

Instructor: Brett Wortzman

TAs:	Archit	Daniel	Naomi	Sreshta
	Blake P.	David	Neha	Suyash
	Blake R.	Hanna	Nicole	Trisha
	Cady	Isis	Rio	Wesley
	Carson	Jun	Rohan	Yash
	Cole	Landon	Shreyank	
	Connor	Michelle	Somya	

Music:  [CSE 122 26au Lecture Tunes](#) 

Lecture Outline

- **Announcements/Reminders** 
- PCM Review
- Code Tracing
- Programming Example
- File output with PrintStream

Announcements

- The IPL is open!
 - Schedule will be on the course website
- Materials and recordings from Java Review posted
- Programming Project 0 out Friday, Oct. 9, due next Thursday, Oct. 15
- Computer Quiz 0 window begins Friday, Oct. 16
 - See assignment and quiz dates on [course calendar](#)
- Just joined CSE 122? Welcome!
 - All materials on [course website](#)
 - Canvas not published– this is expected
- Reminder: Complete PCM before class
 - Due by 11:59pm for participation credit

Lecture Outline

- Announcements/Reminders
- **PCM Review** ◀
- Code Tracing
- Programming Example
- File output with `PrintStream`

(Review) Scanner for User Input

Scanner is defined in the
java.util package

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

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

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) Scanner for File I/O

Scanner is defined in the `java.util` package

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

File is defined in the `java.io` package

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

```
File file = new File("Example.txt");  
Scanner fileScan = new Scanner(file);
```

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

(PCM) Checked Exceptions

- New possible problem: the input file doesn't exist
- This is such a big deal, Java makes you acknowledge it:

error: unreported exception FileNotFoundException; must be caught or declared to be thrown

- Fix this error by declaring the exception:

```
public static void fileMethod() throws FileNotFoundException {  
    ...  
}
```

- This does not *prevent* the error, just makes the compiler happy!

(PCM) Token-Based Processing Pattern

```
while (scan.hasNext____()) {  
    _____ nextToken = scan.next____();  
    // do something with nextToken  
}
```

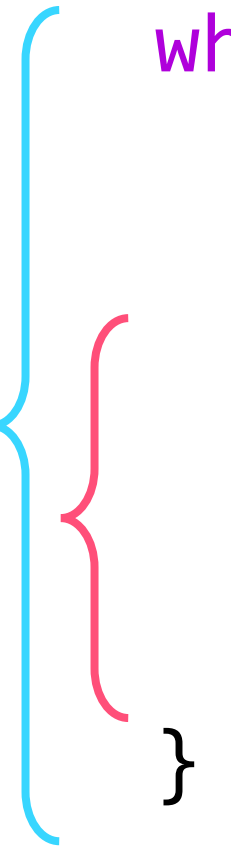
(PCM) Line-Based Processing Pattern

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

(PCM) Hybrid Processing Pattern

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

(PCM) Hybrid Processing Pattern



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

Line-by-line and then token-by-token to do something

(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);  
}
```



(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

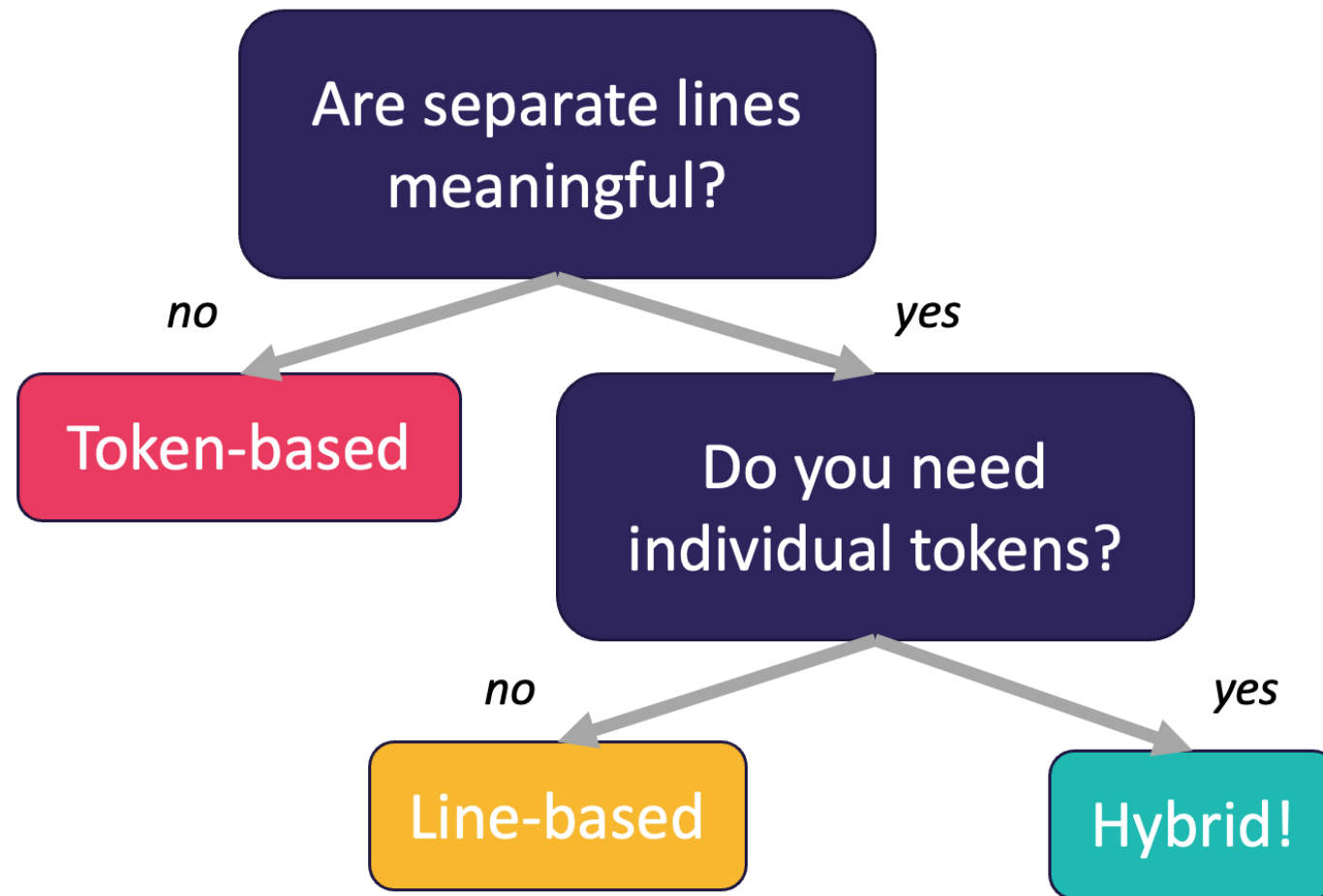
```
String str = "A quick, brown fox";
```

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



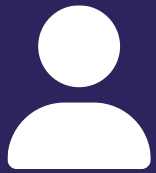
fox

Token-based vs. Line-based vs. Hybrid



Lecture Outline

- Announcements/Reminders
- PCM Review
- **Code Tracing Practice** 
- Programming Practice
- File output with PrintStream



Practice : Think

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

What is the output of this Java code?

```
File f = new File("Example.txt");
Scanner fileScan = new Scanner(f);
while (fileScan.hasNext()) {
    String word = fileScan.next();
    System.out.println("W: " + word + ", ");
}
```

A) W: One, W: Two,
W: Three,

B) W: One,
W: Two,
W: Three,

C) W: One, Two,
W: Three,

D) Exception

Example.txt:

One Two
Three



Practice : Pair



sli.do

#cse122

What is the output of this Java code?

```
File f = new File("Example.txt");
Scanner fileScan = new Scanner(f);
while (fileScan.hasNext()) {
    String word = fileScan.next();
    System.out.println("W: " + word + ", ");
}
```

A) W: One, W: Two,
W: Three,

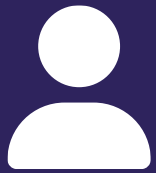
B) W: One,
W: Two,
W: Three,

C) W: One, Two,
W: Three,

D) Exception

Example.txt:

One Two
Three



Practice : Think



sli.do

#cse122

What is the output of this Java code?

```
File f = new File("Example.txt");
Scanner fileScan = new Scanner(f);
while (fileScan.hasNextLine()) {
    String line = fileScan.nextLine();
    Scanner lineScan = new Scanner(line);
    while (lineScan.hasNext()) {
        String word = lineScan.next();
        System.out.print("W: " + word + ", ");
    }
    System.out.println();
}
```

A) W: One, W: Two,
W: Three,

B) W: One,
W: Two,
W: Three,

C) W: One, Two,
W: Three,

D) Exception

Example.txt:

One Two
Three



Practice : Pair



sli.do

#cse122

What is the output of this Java code?

```
File f = new File("Example.txt");
Scanner fileScan = new Scanner(f);
while (fileScan.hasNextLine()) {
    String line = fileScan.nextLine();
    Scanner lineScan = new Scanner(line);
    while (lineScan.hasNext()) {
        String word = lineScan.next();
        System.out.print("W: " + word + ", ");
    }
    System.out.println();
}
```

A) W: One, W: Two,
W: Three,

B) W: One,
W: Two,
W: Three,

C) W: One, Two,
W: Three,

D) Exception

Example.txt:

One Two
Three

Lecture Outline

- Announcements/Reminders
- PCM Review
- Code Tracing Practice
- **Programming Practice** 
- File output with PrintStream

Initial Letter Counts

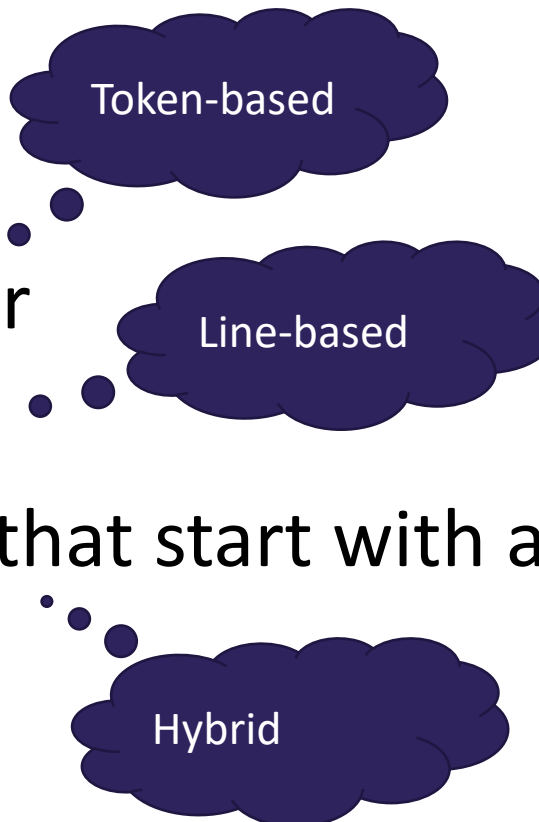
Three tasks:

- Count words that start with a letter
- Count lines that start with a letter
- Find the line with the most words that start with a letter

Initial Letter Counts

Three tasks:

- Count words that start with a letter
- Count lines that start with a letter
- Find the line with the most words that start with a letter



Token-based

Line-based

Hybrid

Lecture Outline

- Announcements/Reminders
- PCM Review
- Code Tracing Practice
- Programming Practice
- **File output with `PrintStream`** ◀

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!