

LEC 02

CSE 122

File I/O – Token and line-based processing

Questions during Class?

Raise hand or send here

sli.do #cse122



BEFORE WE START

Talk to your neighbors:


Slido Poll: Are you a cat person or dog person?

Music: [122 25wi Lecture Tunes](#) 

Instructor: Elba Garza

TAs:	Anya	Daniel Ryan	Ken	Nicole
	Ashley	Diya	Kuhu	Nicole
	Cady	Elizabeth	Kyle	Niyati
	Caleb	Hannah	Leo	Sai
	Carson	Harshitha	Logan	Steven
	Chaafen	Ivory	Maggie	Yang
	Colin	Izak	Mahima	Zach
	Connor	Jack	Marcus	
	Dalton	Jacob	Minh	

Lecture Outline

- **Announcements/Reminders** 
- Review Java
- Scanners for User Input and Files
 - Token-based & Line-based processing
- File I/O Examples

Announcements

- The IPL is open!
 - MGH 334, Schedule is on the course website; staffed by our awesome TAs!
 - Open 12:30 to 9:30PM most days, but check the schedule...
- Creative Project 0 due Thursday, January 16th at 11:59pm
 - Make sure to complete the “Final Submission” slide and **submit!**
 - Submit as many times as you’d like—we will only grade the latest submission made before the deadline
- Just joined CSE 122? That’s okay; look at Ed & [course website](#) and catch up!
 - Freaking out that C0 is due tomorrow? It’s ok! [Resubmission cycles](#) allow you to submit it later.
- Go to your designated quiz sections!
- Quiz dates:

Quiz 0: January 28th

Quiz 1: February 18th

Quiz 2: March 4th


Lecture Outline

- Announcements/Reminders
- **Review Java** ◀
- Scanners for user input and Files
 - Token-based & Line-based Processing
- File I/O Examples


Reminders: Review Java Syntax

[Java Tutorial](#) reviews all the relevant programming features you should familiar with (even if you don't know them in Java).

- Printing and comments
- Variables, types, expressions
- Conditionals (if/else if/ else)
- Loops (for and while)
- Strings
- Methods
- Arrays & 2D arrays

There were some technical difficulties with the recordings of the Java Review Sessions from Monday (January 13th), but we uploaded a couple from 24Au 

Lecture Outline

- Announcements/Reminders
- Review Java
- **Scanner for User Input and Files**
 - **Token-based & Line-based Processing** 
- File I/O Examples

(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) Token vs. Line-based Scanning

[The quick, brown fox
Jumped over the
Lazy dog.]

Token are units of input (as defined by the Scanner) that are separated by *whitespace* (spaces, tabs, new lines)

(PCM) Token vs. Line-based Scanning

The quick, brown fox
Jumped over the
Lazy dog.

The

(PCM) Token vs. Line-based Scanning

The quick, brown fox
Jumped over the
Lazy dog.

quick,

(PCM) Token vs. Line-based Scanning

The quick, brown fox
Jumped over the
Lazy dog.

brown

(PCM) Token vs. Line-based Scanning

The quick, brown fox
Jumped over the
Lazy dog.

fox

(PCM) Token vs. Line-based Scanning

[The quick, brown fox
Jumped over the
Lazy dog.]

(PCM) Token vs. Line-based Scanning

The quick, brown fox
[Jumped over the
Lazy dog.

The quick, brown fox



Practice : Think



sli.do

#cse122

How many tokens are in the following line?

“Hello world !” my-name is Elba

- A) Four B) Five C) Six D) Seven**



Practice : Pair

sli.do

#cse122

How many tokens are in the following line?

↓ ↓ ↓ ↓ ↓ ↓

“Hello world !” my-name is Elba

A) Four

B) Five

C) Six

D) Seven

(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 from the user as an <code>int</code> and returns it
<code>nextDouble()</code>	Reads the next token from the user as a <code>double</code> and returns it
<code>next()</code>	Reads the next token from the user as a <code>String</code> and returns it
<code>nextLine()</code>	Reads an <i>entire line</i> from the user 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

If you try to compile a program working with file scanners, you may encounter this error message:

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

To resolve this, you need to be `throws FileNotFoundException` at the end of the header of any method containing file Scanner creation code, or any method that calls that method!

This is like signing a waiver and telling Java – "Hey, I hereby promise to not get mad at you when you bug out and crash my program if I give you a file that doesn't actually exist."

(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  
}
```



Practice : Think

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

What is the output of this Java program?

```
import java.util.*;
import java.io.*;
public class Demo {
    public static void main(String[] args) throws
        FileNotFoundException {
        File f = new File("Example.txt");
        Scanner fileScan = new Scanner(f);
        while (fileScan.hasNextLine()) {
            System.out.print(fileScan.nextLine() + ", ");
        }
    }
}
```

Example.txt:

```
One Two
Three
```

- A) One, Two, Three,
- B) One, Two, Three,
- C) One Two, Three,
- D) One Two, Three,
- E) Error / Exception



Practice : Pair



sli.do #cse122

What is the output of this Java program?


```
import java.util.*;
import java.io.*;
public class Demo {
    public static void main(String[] args) throws
        FileNotFoundException {
        File f = new File("Example.txt");
        Scanner fileScan = new Scanner(f);
        while (fileScan.hasNextLine()) {
            System.out.print(fileScan.nextLine() + ", ");
        }
    }
}
```

Example.txt:

One Two
Three

- A) One, Two, Three,
- B) One, Two, Three,
- C) One Two, Three,
- D) One Two, Three,
- E) Error / Exception

Lecture Outline

- Announcements/Reminders
- Review Java
- Scanner for User Input and Files
 - Token-based & Line-based Processing
- **File I/O Examples** 

(Friday's PCM) Typical Hybrid Pattern

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