



CMPINF0401

Recitation

TUESDAYS 11:00-12:50

MICHAEL BARTLETT

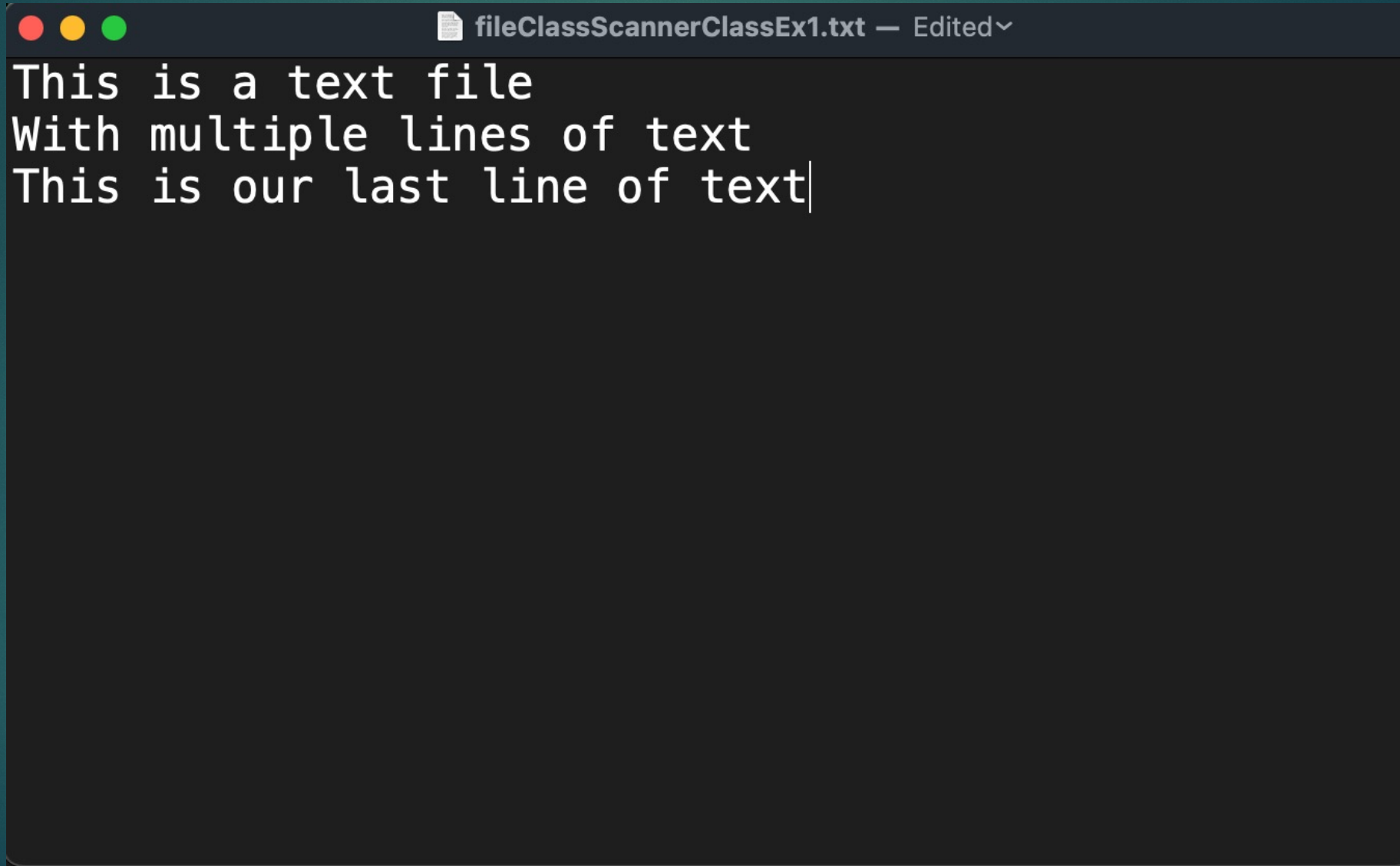
Overview

- ▶ File I/O
- ▶ Arrays
- ▶ Lab 5
- ▶ Assignment 2

File I/O: Using the Scanner

- ▶ First, we need to use the File Class to get a reference to the file that we want to read.
- ▶ Then we use a Scanner to read the file and do something with the data.
- ▶ Scanner has multiple methods that we can utilize to read the file.
 - ▶ [Check out the Java API docs for Scanner](#)

Reading Text Files Using Scanner



```
fileClassScannerClassEx1.txt — Edited✓  
This is a text file  
With multiple lines of text  
This is our last line of text|
```


Reading Text Files Using Scanner

```
File file = new File("fileClassScannerClassEx1.txt");
Scanner sc = new Scanner(file);

while (sc.hasNextLine()) // Checks if the file has another line
{
    System.out.println(sc.nextLine()); // Prints the next line while the condition is true
}
```

→ This is a text file
With multiple lines of text
This is our last line of text

```
File file1 = new File("fileClassScannerClassEx1.txt");
Scanner sc1 = new Scanner(file1);

while(sc1.hasNext()) // Checks if file has another "token"
{
    // In this case a "token" is a string of characters that is terminated by a space
    System.out.println(sc1.next()); // Prints the next word
}
```

→ This
is
a
text
file
With
multiple
lines
of
text
This
is
our
last
line
of
text

Reading Text Files Using BufferedReader

- ▶ First, we need to use the File Class to get a reference to the file that we want to read.
- ▶ Then we use a BufferedReader to read the file.
- ▶ BufferedReader has multiple methods that we can utilize to read the file.
 - ▶ [Check out the Java API docs for BufferedReader](#)

Reading Text Files Using BufferedReader

```
public static void main(String[] args) throws Exception {
    File file = new File("fileClassScannerClassEx1.txt");
    BufferedReader br = new BufferedReader(new FileReader(file)); // Declare new buffered reader while using FileReader to read the file in
    String str; // String to hold values to print

    System.out.println();
    System.out.println();

    while ((str = br.readLine()) != null) // Checks if the file has another line
    {
        System.out.println(str); // Prints the next line while the condition is true
    }

    System.out.println();
    System.out.println();
}
```



This is a text file
With multiple lines of text
This is our last line of text

Scanner Vs. BufferedReader

- ▶ Scanner will *parse* every “token” in a file that way you can interpret all the “tokens” in a file
 - ▶ Parsing: Interpreting the given input as tokens (parts). It's able to give back you specific parts directly as int, string, decimal, etc. See also all those nextXx() methods in Scanner class.
- ▶ BufferedReader will simply read the stream without doing any special parsing
 - ▶ Reading = Dumb streaming. It keeps giving back you all characters, which you in turn have to manually inspect if you'd like to match or compose something useful. But if you don't need to do that anyway, then reading is sufficient.

Writing to files Using PrintWriter

- ▶ First, make a new PrintWriter object and set the file name.
- ▶ Then, write your data.
- ▶ Make sure to close it using `writer.close()`;
 - ▶ Your data won't write if you don't do this.

Writing Text Files Using PrintWriter

```
public class PrintWriterEx {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        PrintWriter outputFile = new PrintWriter("output.txt");  
  
        for (int i = 0; i < 25; i++)  
        {  
            outputFile.println("This is line " + i);  
        }  
        outputFile.close();  
    }  
}
```

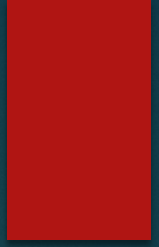


```
output.txt ~  
This is line 0  
This is line 1  
This is line 2  
This is line 3  
This is line 4  
This is line 5  
This is line 6  
This is line 7  
This is line 8  
This is line 9  
This is line 10  
This is line 11  
This is line 12  
This is line 13  
This is line 14  
This is line 15  
This is line 16  
This is line 17  
This is line 18  
This is line 19  
This is line 20  
This is line 21  
This is line 22  
This is line 23  
This is line 24
```


Appending Data To Existing File Using FileWriter

- ▶ First, make a new File object
- ▶ Then, make a new FileWriter object and set the file name.
- ▶ Then, make a BufferedWriter object out of your FileWriter object
- ▶ Write your data and then close the BufferedWriter and FileWriter objects.

Appending Data To Existing File Using FileWriter



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

public class FileWriterEx {
    Run | Debug
    public static void main(String[] args) throws IOException {
        File file = new File("output.txt");
        FileWriter fw = new FileWriter(file, true); // take in the file name and set to true (looking at the Java API the boolean value is to append or to overwrite the data.)
        BufferedWriter bw = new BufferedWriter(fw);
        bw.write("line 25");

        bw.close();
        fw.close();
    }
}
```



```
output.txt - Edited
This is line 0
This is line 1
This is line 2
This is line 3
This is line 4
This is line 5
This is line 6
This is line 7
This is line 8
This is line 9
This is line 10
This is line 11
This is line 12
This is line 13
This is line 14
This is line 15
This is line 16
This is line 17
This is line 18
This is line 19
This is line 20
This is line 21
This is line 22
This is line 23
This is line 24
line 25
```


What's an Array?

- ▶ Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.
 - ▶ This variable is a “reference variable.” This means it “points” to the first object in the array and then all the objects in the array are stored next to it in memory.
 - ▶ Therefore, we use indices when accessing an array. Let's say we have an array named `arr` and we want the first item, `arr[0]` will give us that item, `arr[1]` gives us the second item and so on...
- ▶ When making an array, declare the type that's going to be stored in it and name it. Then set the values that you want in the array.
- ▶ Or you can declare an array that's not equal to anything and just has its size in the square brackets.

Creating/Reading Arrays

```
public class arrayEx1 {  
    Run | Debug  
    public static void main(String[] args) {  
        String[] cars = new String[] {"Volvo", "BMW", "Ford", "Mazda"}; // Make a new array with its values right away  
  
        for (int i = 0; i < cars.length; i++) // Loop through the array and print every element  
            System.out.println(cars[i]);  
  
        int[] nums = new int [10]; // Allocate room for 10 elements of an array in memory  
  
        for (int i = 0; i < nums.length; i++) // Fill the array with the number that their space in the array represents  
            nums[i] = i;  
  
        for (int i : nums) // "For each (int) i in nums print i"  
            System.out.println(i);  
    }  
}
```



```
michaelbartlett@Michaels-MacBook-Pro-2 Examples % java arrayEx1  
Volvo  
BMW  
Ford  
Mazda  
0  
1  
2  
3  
4  
5  
6  
7  
8  
9
```


Multidimensional Arrays

```
public static void main(String[] args) {  
    // declaring and initializing 2D array  
    int arr[][] = { {2,7,9},{3,6,1},{7,4,2} };  
  
    // printing 2D array  
    for (int i=0; i< 3 ; i++)  
    {  
        for (int j=0; j < 3 ; j++)  
            System.out.print(arr[i][j] + " ");  
        System.out.println();  
    }  
}
```



```
michaelbartlett@Michaels-MacBook-Pro-2 Examples % java multiDimensionalArrayEx  
2 7 9  
3 6 1  
7 4 2
```

The Array Discipline

- ▶ When you declare an array you should declare an int named count or such to track how many values you have put into the array
- ▶ Initialize count to 0 and then use count to represent two things:
 - ▶ The number of values you have put into the array so far
 - ▶ The index position of where the next value should be stored

Let's look at the example again using Array Discipline

```
public class arrayEx1 {  
    Run | Debug  
    public static void main(String[] args) {  
        int nums[] = new int[10]; // Allocate room for 10 elements of an array in memory  
        int count = 0; // Set count eq. to 0  
  
        while (count < nums.length) // Increment count while setting the element at nums[count] = count  
        {  
            nums[count] = count++; // Post increment which means it will increment the variable after this line runs.  
            // If it was nums[count++] = count; or nums[count] = ++count; then it would increment count before setting the variable equal to it.  
        }  
  
        for (int i : nums)  
            System.out.println(i);  
    }  
}
```

Lab 5

- ▶ Assignment:
 - ▶ https://canvas.pitt.edu/courses/127916/files/8050395?module_item_id=2735274
 - ▶ Make sure to change the name of the class because it currently says "Project 4" but needs to be the same name as the java file which is "Lab5".
- ▶ You're finding four values in this lab.
 - ▶ Minimum value in the array
 - ▶ Maximum value in the array
 - ▶ Average of all the values in the array
 - ▶ Product of all the values in the array

Lab 5: How?

- ▶ You're going to use loops!
- ▶ You can do this in one loop, but your professor would like to see you use one loop per calculation to get you used to them.
- ▶ Let's check out some pseudocode

Lab 5: Pseudocode

Lab5.java > Lab5 > main(String[])

```
6 public static void main( String args[] )
7
8     int[] arr = { 13, 5, 7, 11, 23, 15, 6, 31, 91, 27, 14, 41, 55, 14 };
9     int arrmin, arrmax, arrprod;
10    double arrave;
11
12    /*
13    arrmin = some default value // typically you want to use the first value in the array that way you can compare back to that value and change it as needed
14
15    for (int i = 0; i < arr.length; ++i)          // Set i equal to 0 and then loop through until i is less than arr.length.
16        if (arr[i] is less than arrmin)          arr.length returns how many items are in the array, which is why we need to count to one less than it for the purpose of indexing
17            arrmin = arr[i]                      Our index starts at 0
18    // Since arr[i] is less than arrmin, we have a new min and can update our value accordingly
19
20    // We can then follow the same structure for arrmax
21
22    // The average can be found by adding the sum (need a sum variable: incremented by looping) and then dividing the sum by the number of elements
23
24    // The product can be found by setting the variable equal to a default value (in this case we'd want to start with 1) and then looping through the array
25    and setting the product variable *= arr[i]
26    */
27
28    for ( int i=0 ; i<arr.length ; ++i ) //Sample loop that will process through an entire array.
29    {
30    }
31    System.out.println( "The smallest number in the array is: " + arrmin );
32    System.out.println( "The largest number in the array is: " + arrmax );
33    System.out.println( "The average of the numbers in the array is: " + arrave );
34    System.out.println( "The product of the numbers in the array is: " + arrprod );
35
36
37    // END MAIN METHOD
38
```


Assignment 2

- ▶ Assignment:
 - ▶ https://canvas.pitt.edu/courses/127916/files/8421170?module_item_id=2846089
- ▶ Let's check out some pseudocode

Assignment 2: Pseudocode

```
while (input.hasNext())  
{  
    // pull in the values for name,gpa,level from file  
  
    // test for next level  
    //If GPA is 2.75 or more, advance the students level from its current state, if not 2.75 or more, their grade level will NOT increase  
    // Hint - A nested if test may be the easiest way to accomplish this  
    // Hint - Declare another variable of type boolean named passed and use that to set if the student passed or not  
    /*  
    if (gpa is greater than 2.75)  
        set passed eq. to true  
        nested if statements to set level to the next level (if senior, the next level is graduate)  
    */  
  
    // test for honors  
    // If gpa is 3.5 or more, honors is true, else false  
  
    // Do file output to a2output.txt  
    //Outputs needed to be shown: Student Name, Current grade level, if they are an honors student, and a statement of if they passed or not(You will need another logical test)  
  
    // The first line to set the output would look something like:  
    myWriter.write("Student Name: " + name + "\n"); // Adding \n because file writer doesn't automatically put a new line in  
    System.out.println("Student Name: " + name);  
  
    // Add a new line in myWriter and a println to match the Canvas Output (puts an empty line in between each output)  
}  
//while
```