



CMSC 150

INTRODUCTION TO COMPUTING

LAB – WEEK 5

- FILE IO

The background features a light blue, concentric circular pattern. In the four corners, there are decorative circuit-like lines in a slightly darker blue, consisting of straight lines and small circles, resembling a printed circuit board or a network diagram.

**LETS GO OVER ANSWERS TO PREVIOUS
PROGRAMMING ASSIGNMENT**

INPUT AND OUTPUT

- Input devices



Keyboard



Mouse



Hard drive



Network



Digital camera



Microphone

- Output devices.



Display



Speakers



Hard drive



Network



Printer



MP3 Player

- Goal. Java programs that interact with the outside world.

- Java Libraries support these interactions
- We use the Operating System (OS) to connect our program to them

WHAT HAVE WE SEEN SO FAR?

- **Command-line input.**

- Example: read an integer N as command-line argument.

- **Standard output.**

- The OS output stream for text
- By default, standard output is sent to Terminal.
- Example: `System.out.println()` goes to standard output.

- **Standard input.**

- The OS input stream for text
- By default, standard input is received from the Terminal.
- Example: Scanner

- **“Standard Draw.”**

- Really a wrapper for Java’s GUI libraries
- Output to a window instead of a terminal
- Example: Draw a circle on the screen

FILE INPUT AND OUTPUT



FILE INPUT

- We can reuse Scanner!
- Instead of “scanning” System.in, we scan a File.

1. `Scanner in = new Scanner(
 new File(“myfile.txt”));`
2. `in.nextInt();`
3. `in.nextDouble();`
4. `in.hasNext();`

- However we must:

- Import Scanner
- Import File and FileNotFoundException
- Modify our main function to “throw” FileNotFoundException

1. **import** java.io.File;
2. **import** java.io.FileNotFoundException;
3. **import** java.util.Scanner;
4. **public class** MyProgram {
5. **public static void** main(String[] args)
 throws FileNotFoundException {
6. //Do something!
7. }
8. }

FILE OUTPUT

- We can use `PrintWriter`
- Offers `print`, `println`, `printf` just like `System.out`

1. `PrintWriter out =`

```
    new PrintWriter("MyFile.txt");
```

2. `out.println("Hello FileIO World!");`

- Similarly we need to:


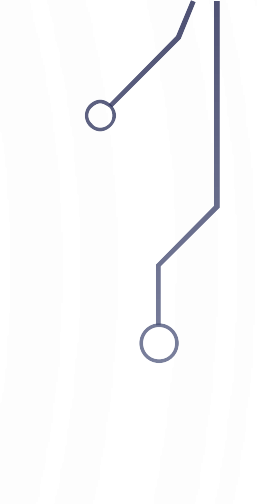
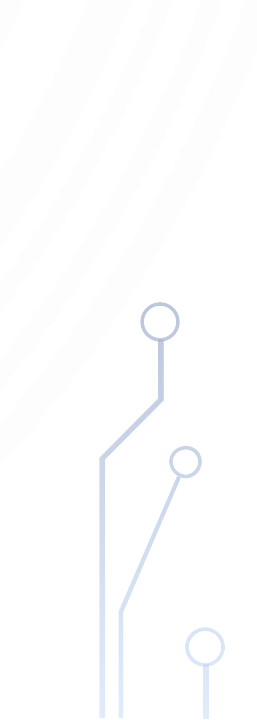
- Import `PrintWriter` and `FileNotFoundException`
 1. `import java.io.PrintWriter;`
- Modify `main` to throw `FileNotFoundException`s

FILE INPUT/OUTPUT CAVEATS

- Always call close after you are done using Scanner or PrintWriter
 1. Scanner in = **new** Scanner(
 new File("MyFile.txt"));
 2. //Use the Scanner as much
 //as you want
 3. in.close();
- Call flush often on PrintWriter to ensure all output gets into the file.
 1. PrintWriter out =
 new PrintWriter("MyFile.txt");
 2. //Use the PrintWriter as much
 //as you want
 3. out.flush(); *//Always flush after use!*
 4. out.close();



FOR MORE INFORMATION

- Google
 - API
 - Tutorials
 - StackOverflow
 - Practice, Practice, Practice!
- 
- 
- 

The background features a light blue, concentric circular pattern. In the four corners, there are decorative circuit-like lines in a slightly darker blue, consisting of straight lines and small circles, resembling a stylized PCB or network diagram.

EXERCISE – EVERYONE CODE ALONG

EXERCISE



Banker of the Month!

- We work for JLDiablo's National Bank of Tristram.
- We manage savings, checking, and loan accounts for the citizens of Tristram, like good ole Wirt with his peg leg.
- We have all of the accounts stored in a file (`accounts.txt`). Each account has an id, a type, and an amount.
- We are gonna write a program to compute monthly changes based on the following:
 - Savings earn interest of 0.01%
 - Loans accrue interest of 0.4%
 - Checking accounts have a monthly fee of 10 gold.
- After we are done, we are going to save to a new file



EXERCISE – START THE PROGRAM

```
1. import java.util.Scanner;
2. import java.io.File;
3. import java.io.FileNotFoundException;
4. Import java.io.PrintWriter;
5. public class Banking {
6.     public static void main(String[] args) throws FileNotFoundException {
7.         String inFileName = args[0];
8.         String outFileName = args[1];
9.         //Read each of the accounts into arrays
10.        //Process the account type
11.        //Output each of the accounts into the new file
12.    }
13. }
```

EXERCISE – START FILLING OUT THE COMMENTS INTO CODE

1. *//Read each account into an array*
2. Scanner in = **new** Scanner(**new** File(inFileName));
3. **int** numAccounts = in.nextInt();
4. String[] accountTypes = **new** String[numAccounts];
5. **double**[] accountValues = **new double**[accountValues];
6. **for**(**int** i = 0; i < numAccounts; ++i) {
7. accountTypes[i] = in.next();
8. accountValues[i] = in.nextDouble();
9. }
10. in.close();

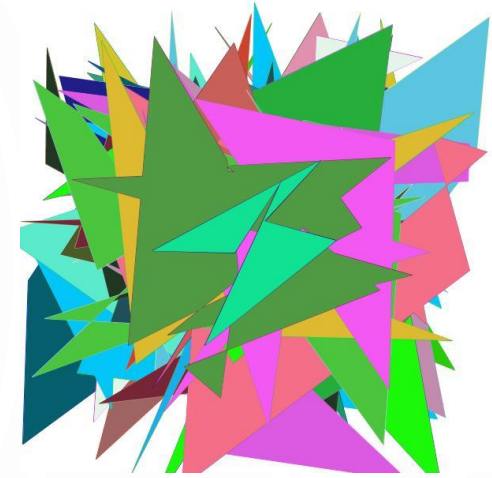
EXERCISE – COMPILE, TEST, AND CONTINUE

1. //Output each of the accounts into the new file
2. `PrintWriter out = new PrintWriter(outFileName);`
3. `out.println(numAccounts);`
4. `for(int i = 0; i < numAccounts; ++i) {`
5. `out.printf("%4d %8s %6.2f\n", i, accountTypes[i], accountValues[i]);`
6. `}`
7. `out.flush();`
8. `out.close();`

EXERCISE – COMPILE, TEST, AND CONTINUE

```
1. //Process the account type
2. for(int i = 0; i < numAccounts; ++i) {
3.     if(accountTypes[i].equals("Savings"))
4.         accountValues[i] *= 1.0001; //0.01% interest
5.     else if(accountTypes[i].equals("Checking"))
6.         accountValues[i] -= 10;    //10 gold monthly fee
7.     else if(accountTypes[i].equals("Loan"))
8.         accountValues[i] *= 1.0004; //0.4% interest
9. }
```

EXERCISE



1. Write a program that will generate a random polygon of $N = [3, 20]$ sides at random (x, y) points between $(-10, 10)$ – i.e., an array. Compute the center of mass $(com_x = \frac{\sum x_i}{N}, com_y = \frac{\sum y_i}{N})$. Shift all points of the polygon by $(-com_x, -com_y)$. Write the polygon (array) to a file – first line is N , each line after is the points (output $x\ y$, not (x, y))
2. Write a program that reads your polygon file and shows it to the user using StdDraw. Use a random color to show the outline and a different random color to fill the polygon.
3. Augment your programs to generate and show $M = [25, 50]$ random polygons, i.e., use multiarrays.