File I/O Summary

by Michael Gerhardt, Krystina Kamjorn, Matthew Mercer

- Simplest way to read in text: Scanner class
- Scanner uses the File class to read input from disk files

  -
    ```
    File fileName = new File("silly.txt");
    Scanner input = new Scanner(fileName);
    ```
- Use the Scanner methods to read data from input file
  - next(), nextLine(), nextInt(), nextDouble()
- Use PrintWriter to write to write output to a file
  - PrintWriter output = new PrintWriter("foo.txt");
- Data from an existing output file will be emptied before new data is written into it, otherwise a new file is created
- print(), println(), and printf() can be used with any PrintWriter object
- You **must** close PrintWriter after you're done writing to a file, otherwise all of the output might not be in the file
  - output.close();
- FileNotFoundException will occur if the input or output file doesn't exist, so throw a FileNotFountException in the main method
- Can also use a JFileChooser to open or save a file

  -
    ```
    JFileChooser picky = new JFileChooser();
    Scanner input = null;
    if (picky.showOpenDialog(null) == JFileChooser.APPROVE_OPTION){
        File fileName = picky.getSelectedFile():
        input = new Scanner(fileName);
    }
    ```
- A word in Java is any non-white space character (e.g., computers. , 1234, A+)
- White space is removed from the input when using hasNext();
- To read only letters, call the useDelimiter() method on the Scanner object
- next() will take an entire line with the white space and display the line without the space
- When using a string literal for a file name, you **must** use two backslashes
  - File inFile = new File("\\src\\input.txt");
  - Using one backslash is valid for user inputs
- nextLine() will remove white space from after the last character in a line, whereas nextInt() and nextDouble() will not