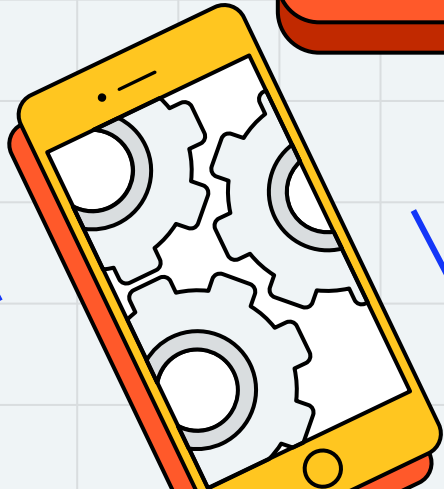


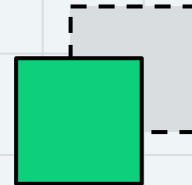


Application

Programming



Hend Alkittawi





File I/O

Working with File Input and Output

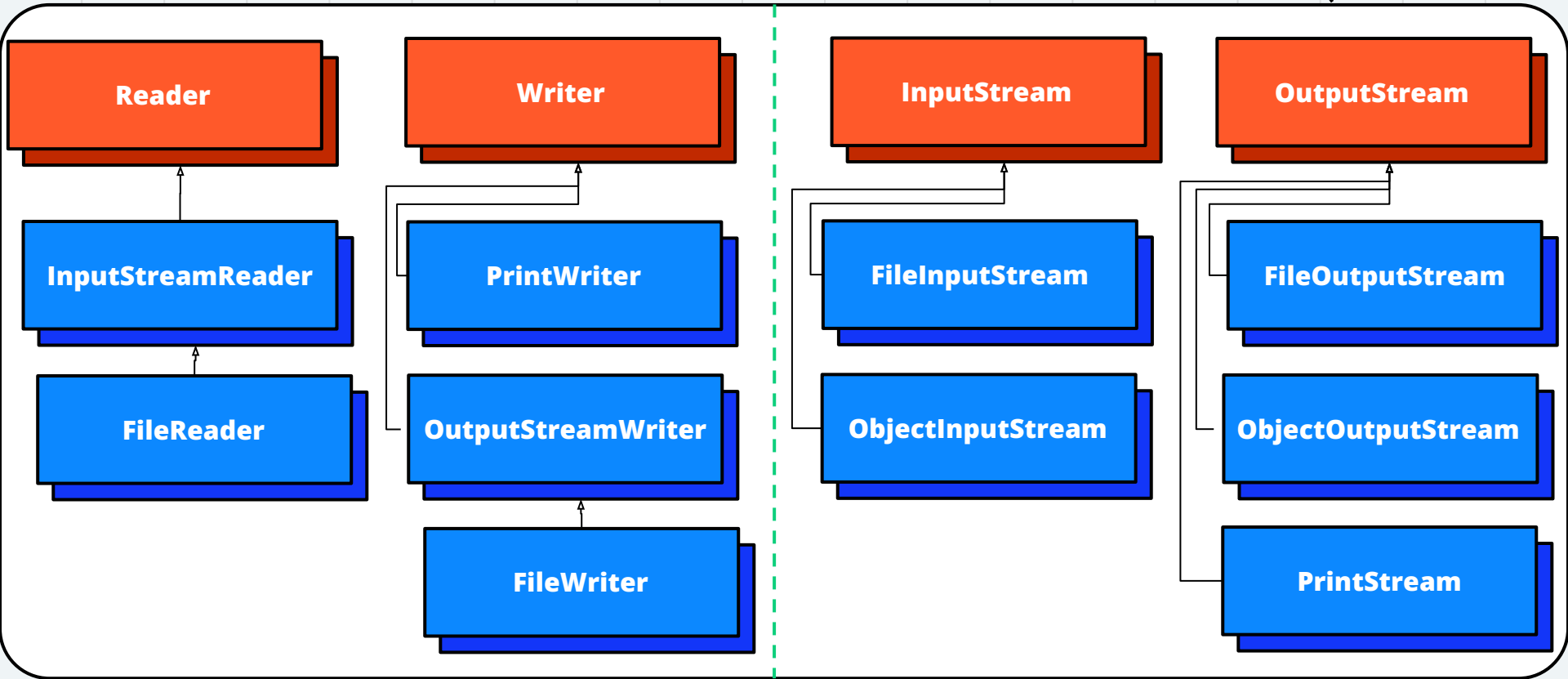
WORKING WITH DATA

- Data can be stored on a computer in a file with a particular name!
- There are two fundamentally different ways to store data in a file:
 - **Text files** store data in **text format**: data items are represented in human readable form, and can be edited using simple text editors.
 - *.txt, *.java, *.html, *.csv
 - **Binary files** store data in **binary format**: data items are represented in bytes, and stored using an internal format that requires special software to process.
 - *.jpeg, *.pdf

READERS, WRITERS, AND STREAMS

- Java APIs provide **two set of classes** for handling input and output.
- If data is stored in **text form** (as a sequence of characters), the **Reader** and **Writer** classes and their subclasses are used to produce **input** and **output**.
- If data is stored in **binary form** (as a sequence of bytes), the **InputStream** and **OutputStream** classes and their subclasses are used to produce **input** and **output**.

READERS, WRITERS, AND STREAMS

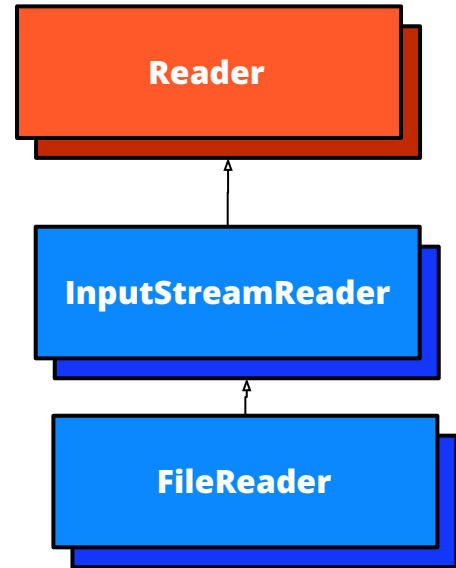


READERS, WRITERS, AND STREAMS

- To **read** text data from a disk file, create a **FileReader** object
- To **read** binary data from a disk file create a **FileInputStream** object
- To **write** text data to a disk file, create a **FileWriter** object
- To **write** binary data to a disk file create a **FileOutputStream** object

THE Scanner CLASS

- The Scanner class is flexible in that Scanner objects can be attached to many different kinds of input.
 - input from the console
 - input from a file
- When constructing a **Scanner** from a file object, the **Scanner** automatically constructs a **FileReader**.
- The **Scanner** class is part of the java.util package.
- If the program is unable to locate the specified input file, it will generate an exception which will prevent the the program from continuing normal execution.
- You can throw the exception or handle it with a try/catch.



THE Scanner CLASS

- Some of the useful methods for reading input in the Scanner class:
 - token-based processing
 - hasNext(): checks if the input has a token to read
 - next(): reads and returns the next token as a String
 - nextInt(): reads and returns an int value
 - nextDouble(): reads and returns a double value
 - line-based processing
 - hasNextLine(): checks if the input has a line to read
 - nextLine(): reads and returns the next line of input as a String

FILES AND File OBJECTS

- To access a file from inside a Java program, you need to construct an internal object that will represent the file.
- Once you have constructed the object, you can call a number of methods to manipulate the file.

```
File myFile = new File ("myFile.txt");  
if (file.exists() && file.canRead()) {  
    // read the file!  
}
```

- The **File** class is part of java.io package.

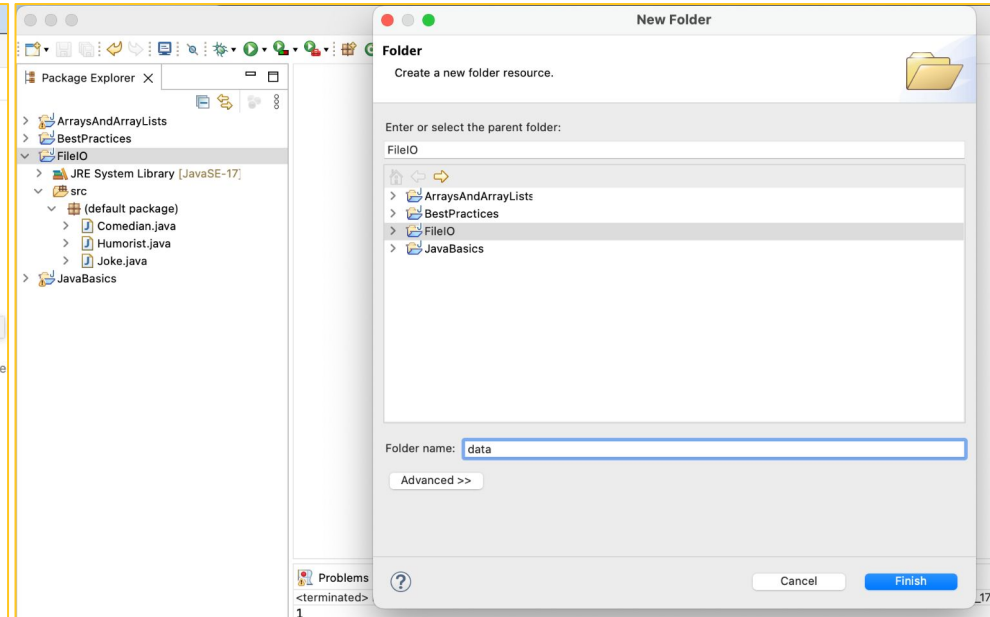
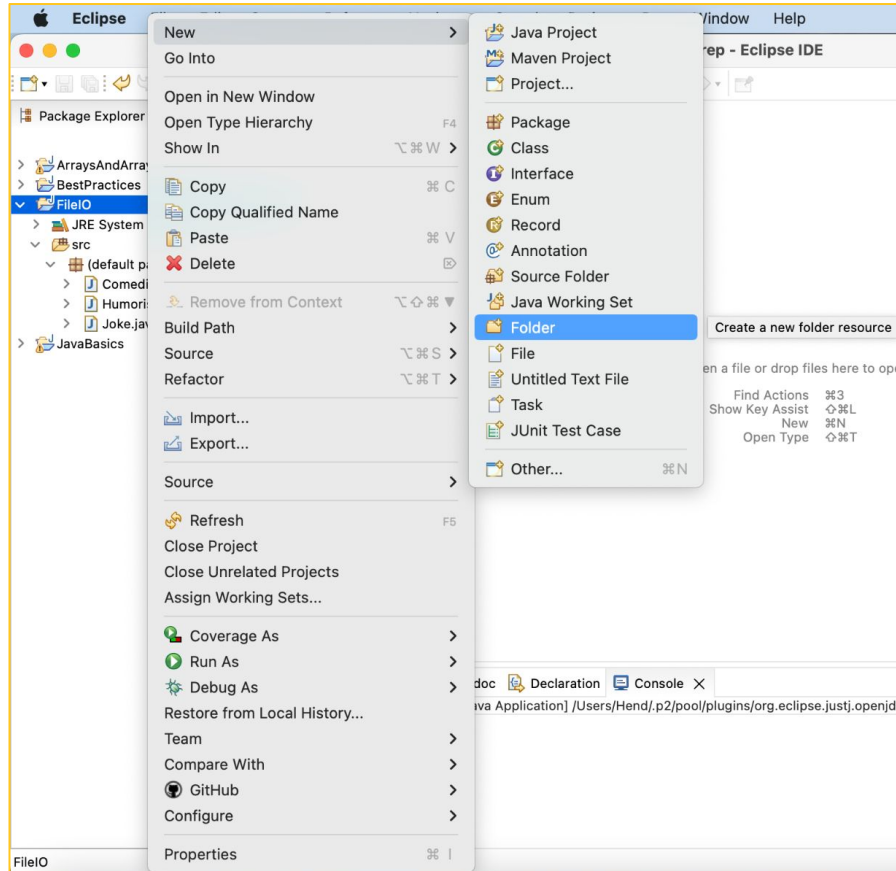
READING A FILE WITH THE Scanner CLASS

- Instead of instantiating objects with fixed values coded into the application, we can instantiate objects by reading data from a file.
 - create a **data directory** in your Eclipse project, and place the file in it. Then, add code to read the data from a file

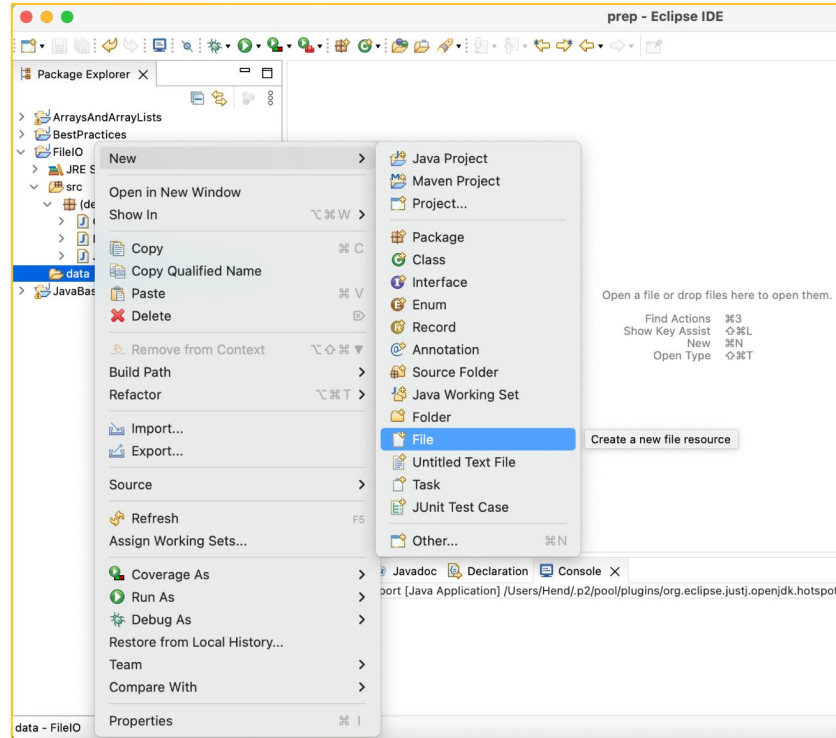
```
public void loadData(String fileName) throws FileNotFoundException {
    File file = new File( "data/" + filename );
    Scanner scan = new Scanner( file );
    while( scan.hasNextLine() ) {
        String line = scan.nextLine();
        String[] tokens = line.split(",");
        // do sth with the string and/or tokens, we will create an object based on the tokens!
        MyClass obj = new MyClass(tokens[0], Double.parseDouble(tokens[1]), ...);
    }
    scan.close();
}
```

To create a **data directory** in your Eclipse project

right-click on project > New > Folder



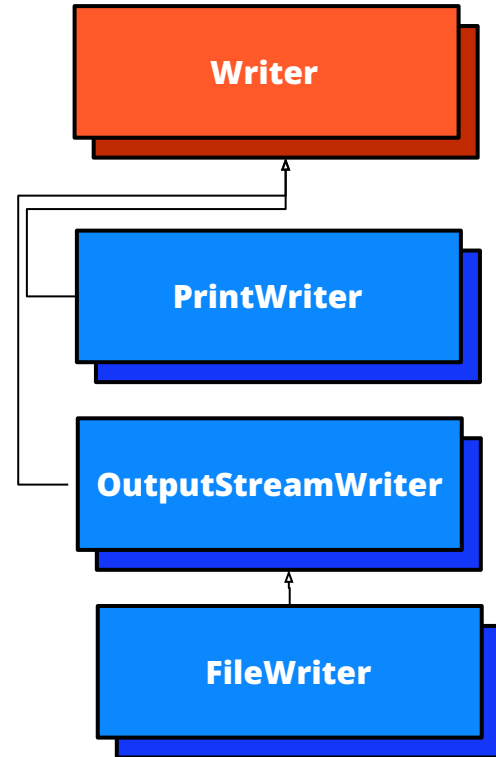
Create a new file in the data directory, or copy and paste an existing file
right-click on data folder > New > File then enter data
OR
copy an existing file > right-click on data folder > paste



THE FileWriter CLASS

- The `FileWriter` class can be utilized to write text data to a file.
- Use the `write()` method to write a `String` to the output file.
- If the program is unable to locate the specified input file, it will generate an exception which will prevent the the program from continuing normal execution. You can throw the exception or handle it with a `try/catch`.

```
public void writeFile(String filename) throws IOException{  
    File outFile = new File("data/" + filename);  
    FileWriter out = new FileWriter(outFile);  
    out.write("Success! \n");  
    out.write("Keep it up ...");  
    out.close();  
}
```





CODE DEMO

- Modify the classes created in previous sessions to demo file I/O concepts.



THANK

YOU!



DO YOU HAVE ANY QUESTIONS?



hend.alkittawi@utsa.edu



By Appointment



Online