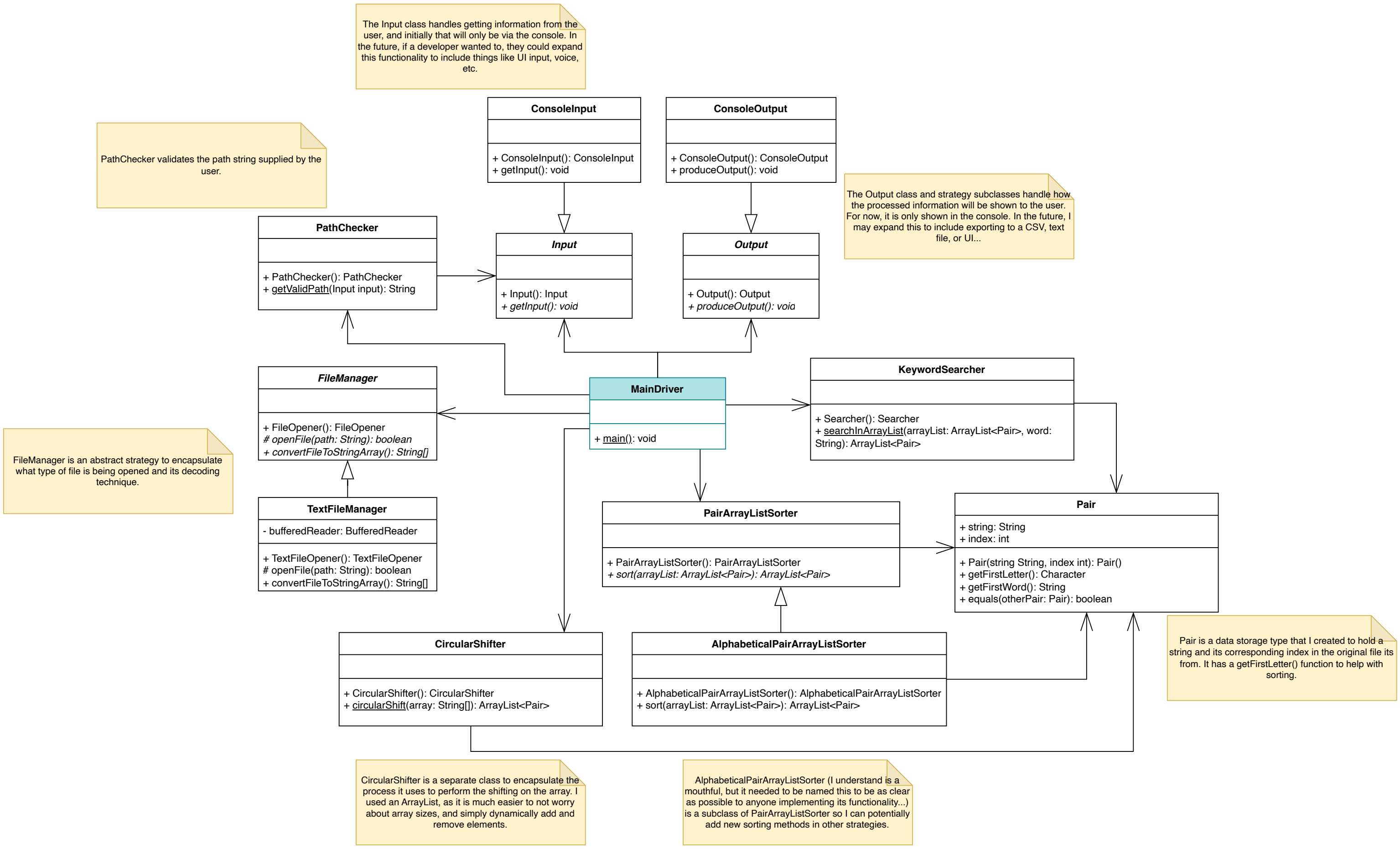
ConsoleInput
+ ConsoleInput(): ConsoleInput
+ getInput(): void
ConsoleOutput
+ ConsoleOutput(): ConsoleOutput
+ produceOutput(): void
PathChecker
+ PathChecker(): PathChecker
+ getValidPath(Input input): String
Input
+ Input(): Input
+ getInput(): void
Output
+ Output(): Output
+ produceOutput(): void
FileManager
+ FileOpener(): FileOpener
# openFile(path: String): boolean
+ convertFileToStringArray(): String[]
MainDriver
+ main(): void
KeywordSearcher
+ Searcher(): Searcher
+ searchInArrayList(arrayList: ArrayList<Pair>, word: String): ArrayList<Pair>
TextFileManager
- bufferedReader: BufferedReader
+ TextFileOpener(): TextFileOpener
# openFile(path: String): boolean
+ convertFileToStringArray(): String[]
PairArrayListSorter
+ PairArrayListSorter(): PairArrayListSorter
+ sort(arrayList: ArrayList<Pair>): ArrayList<Pair>
Pair
+ string: String
+ index: int
+ Pair(string String, index int): Pair()
+ getFirstLetter(): Character
+ getFirstWord(): String
+ equals(otherPair: Pair): boolean
CircularShifter
+ CircularShifter(): CircularShifter
+ circularShift(array: String[]): ArrayList<Pair>
AlphabeticalPairArrayListSorter
+ AlphabeticalPairArrayListSorter(): AlphabeticalPairArrayListSorter
+ sort(arrayList: ArrayList<Pair>): ArrayList<Pair>

The Input class handles getting information from the user, and initially that will only be via the console. In the future, if a developer wanted to, they could expand this functionality to include things like UI input, voice, etc.

PathChecker validates the path string supplied by the user.

The Output class and strategy subclasses handle how the processed information will be shown to the user. For now, it is only shown in the console. In the future, I may expand this to include exporting to a CSV, text file, or UI...

FileManager is an abstract strategy to encapsulate what type of file is being opened and its decoding technique.

Pair is a data storage type that I created to hold a string and its corresponding index in the original file its from. It has a getFirstLetter() function to help with sorting.

CircularShifter is a separate class to encapsulate the process it uses to perform the shifting on the array. I used an ArrayList, as it is much easier to not worry about array sizes, and simply dynamically add and remove elements.

AlphabeticalPairArrayListSorter (I understand is a mouthful, but it needed to be named this to be as clear as possible to anyone implementing its functionality...) is a subclass of PairArrayListSorter so I can potentially add new sorting methods in other strategies.