Seminar 1: Software Architecture - Presentation

DVA422

Matko Butković

Task:

- Describe the decomposition structure of the classes in your solution
- Include a complete list of classes along with a description of the design decisions that each class encapsulates (i.e., hides)
- Discuss whether there are any examples of the is-a-submodule-of relation in your system.
- What other criteria than the hiding of design decisions (if any) did you use for decomposing your system into classes?
- Propose a division of the work between the two developers, which should be able to make their own design decisions as independently as possible (and do as much parallel work as possible).

Alternative A: Decomposition structure

- Modules are assigned specific computational responsibilities
- The units are modules that are related to each other by the is-a-submodel-of relation, showing how modules are decomposed into smaller modules recursively until the modules are small enough to be easily understood.
- Useful for resource allocation and project structuring and planning; information hiding, encapsulation; configuration control
- Quality Attribute Affected: Modifiability

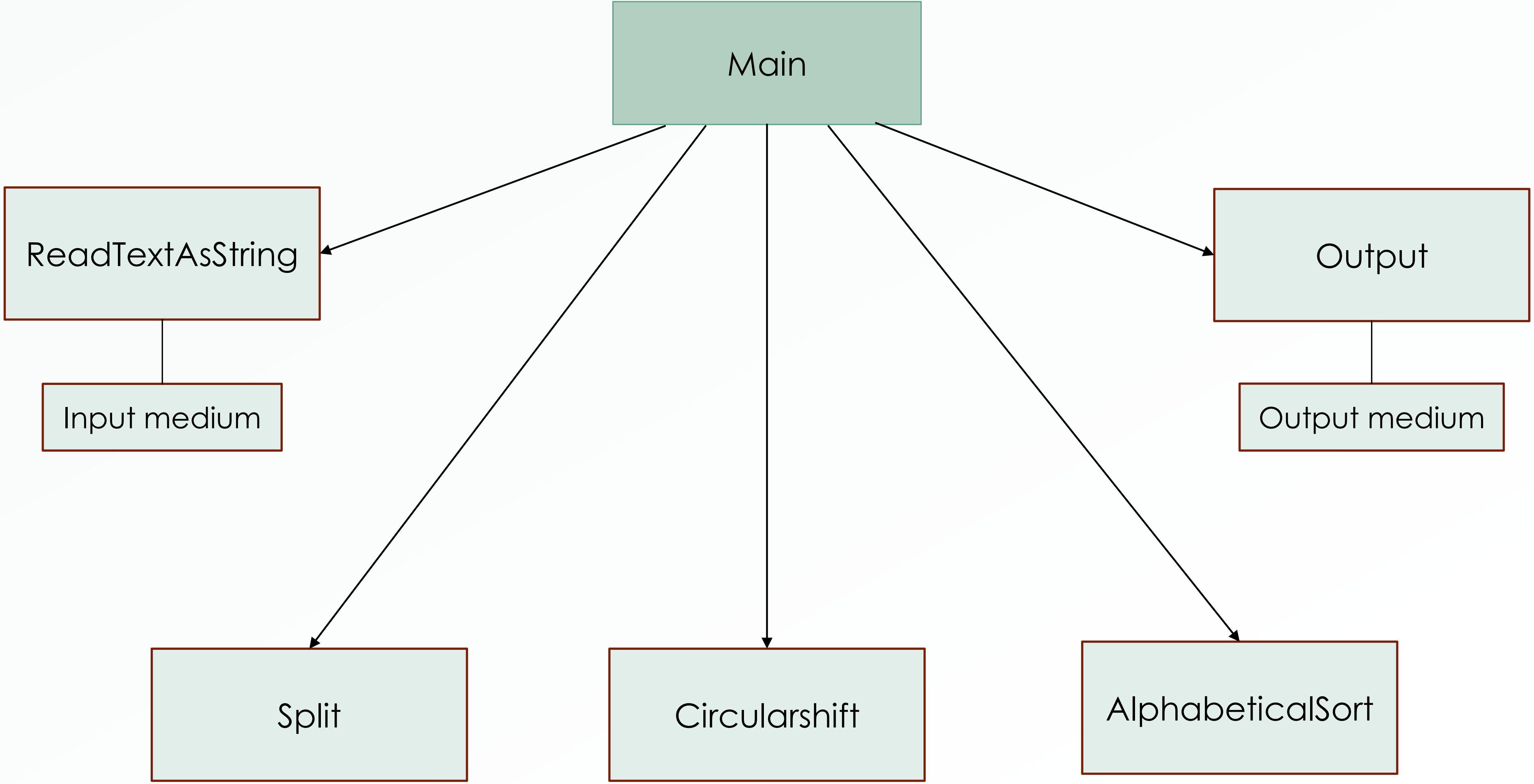
Decomposition of KWIC system:

- System which we implemented: The key word in context (KWIC)
- One module for each of the five function (design decision)
- Decomposition of system to modules:
  1. The input media and data format
  2. The format for storing the lines in the input
  3. The algorithm for computing the circularly shifted lines
  4. The algorithm for sorting the circularly shifted lines
  5. The output media and data format
  - Main program calls each of the subroutines in turn
- Each module can be tested as separate unit of the system

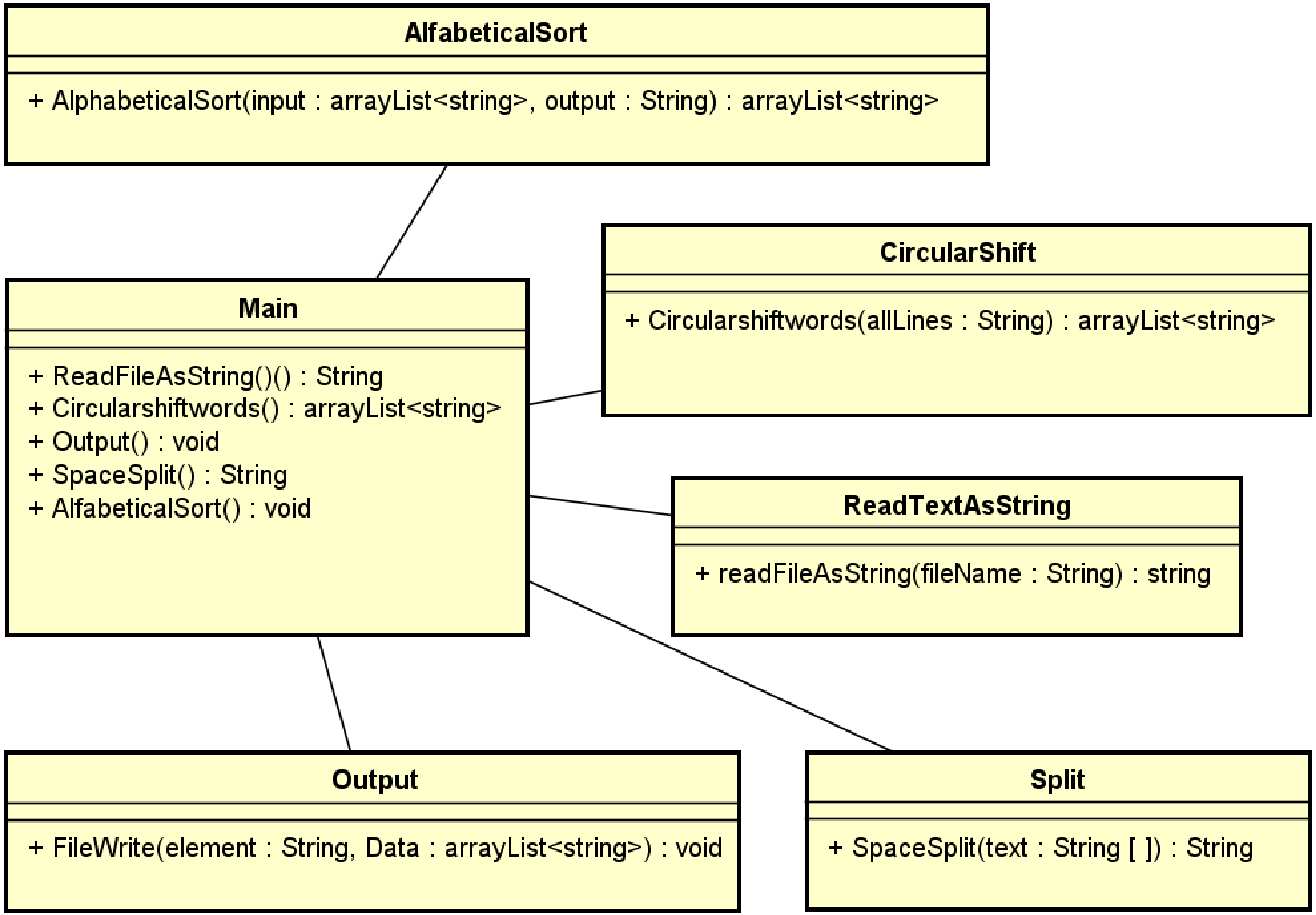
List of classes (modules):

	Class name:	Function:	Input:	Output:
1.	**ReadTextAsString**	Reads the .txt file	Path Argument to .txt file	String
2.	**Split**	Splits words by space	String	Array of Strings
3.	**Circularshift**	Perform circularshift	String	Array List
4.	**AlphabeticalSort**	Sorts the lines	Array List	Array List
5.	**Output**	Write to .txt document	Array List; Path Argument to .txt file	.txt file

Decomposition structure:



Class diagram:



AlfabeticalSort

+ AlphabeticalSort(input : arrayList<string>, output : String) : arrayList<string>

Main

+ ReadFileAsString()() : String
+ Circularshiftwords() : arrayList<string>
+ Output() : void
+ SpaceSplit() : String
+ AlfabeticalSort() : void

CircularShift

+ Circularshiftwords(allLines : String) : arrayList<string>

ReadTextAsString

+ readFileAsString(fileName : String) : string

Output

+ FileWrite(element : String, Data : arrayList<string>) : void

Split

+ SpaceSplit(text : String [ ]) : String

Decomposition structure

- Module decomposition:
  - Main class has main executable method that is private (its not used by other classes and it uses functionality from other classes)
  - Other classes (ReadTextAsString, Split, CircularShift, AlfabeticalSort, Output) provide operational functionality for the master class
  - Operational module:
  - Input/Output data handler submodules: ReadTextAsString, Output
  - Operational submodule: AlphabeticalSort, CircularShift, Split

Modifiability:

- Data format of input and output file
- Split function based on spaces
- Different type of shift
- Different type of sorting

System implementation

- How to implement the system by two developers if the decomposition structure is decided?
- Input: Low-level design document
- Divide the work to each developer (Gantt diagram):
  - Developer A works on module 1,2;
  - Developer B works on module 3,4,5.
- Merge:
  - Developer A can create Main class to call the methods from implemented classes;
  - Developer B can create test cases

Thank you!

- Questions?