REVISION HISTORY

Date	Version	Description	Author
24.11.2021	0.1	Software Architecture overview is created	Uygar KAYA
25.11.2021	0.2	InputOutput Package is created	Uygar KAYA
26.11.2021	0.3	Algorithm Package is created	Uygar KAYA
27.11.2021	0.4	The whole diagram is created	Uygar KAYA
28.11.2021	0.5	COTS Identification is created	Uygar KAYA
16.12.2021	0.6	Report was updated	Uygar KAYA

TABLE OF CONTENTS

1 Introduction

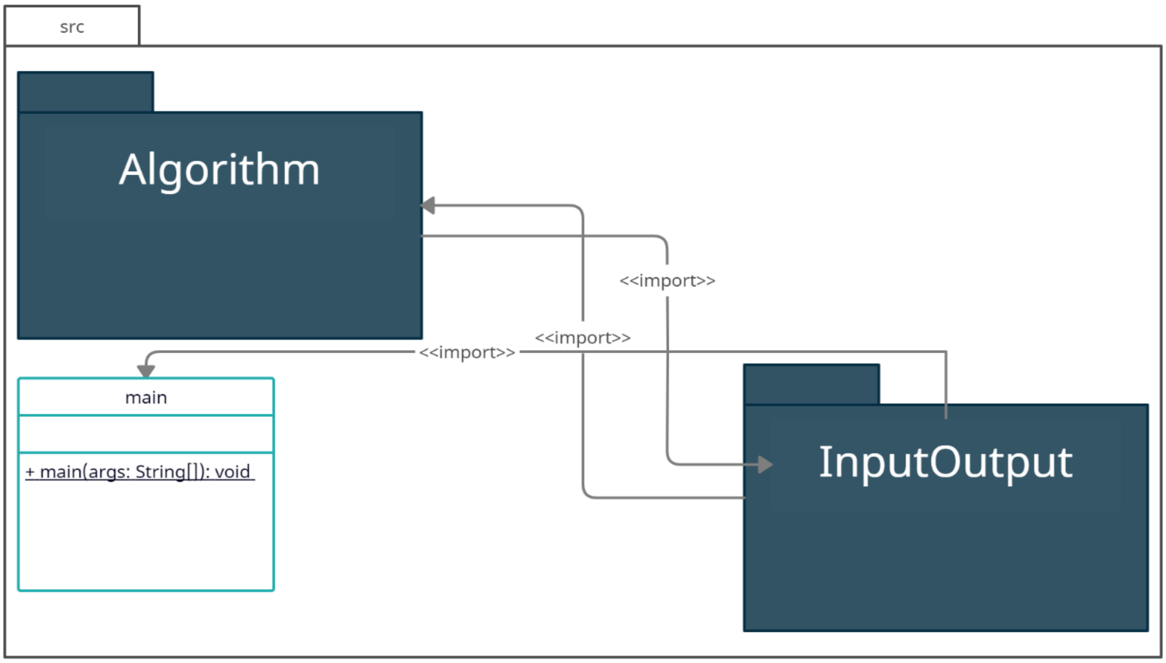
This document describes the design of the PDA (Push Down Automata) software system. In this project, I programmed the PDA Algorithm, which is given a string the simulated PDA should be able to tell if the string is accepted or rejected. In this project, I will implement the program with Java Programming Language.

1.1 References

1.1.1 Project References

#	Document Identifier	Document Title
[R1]	DFA-SDD	Software Detailed Design Document of DFA

2 Software Architecture overview



3 Software design description

3.1 InputOutput

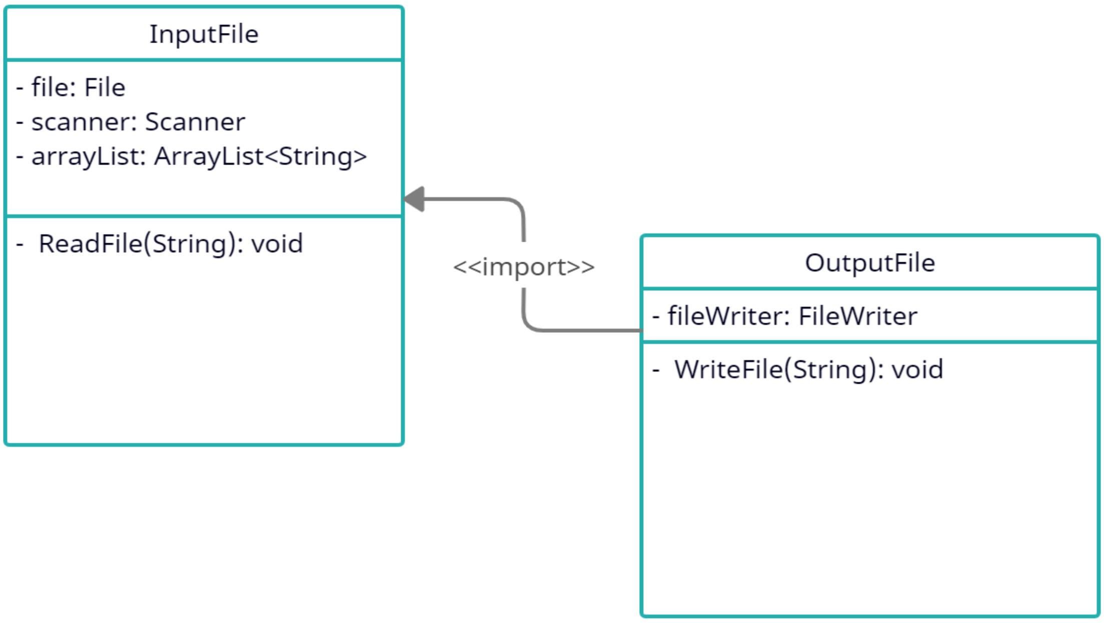
3.1.1 Component interfaces

In the InputOutput Package, there are two classes which are InputFile & OutputFile

Methods of InputFile which are available from other components are:
public void ReadFile(String filePath) – To read the input .txt file

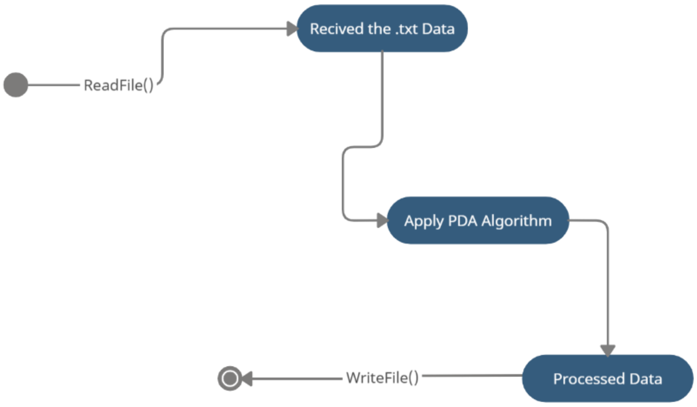
Methods of OutputFile which are available from other components are:
public void WriteFile(String routeValidation) – To write the output .txt file

3.1.2 Component design description



3.1.3 Workflows and algorithms

The Activity Diagram is given below:



3.2 Algorithm

3.2.1 Component interfaces

In the Algorithm Package, there are four classes which are Configuration, PDAlgorithm, SeparateData & Stack

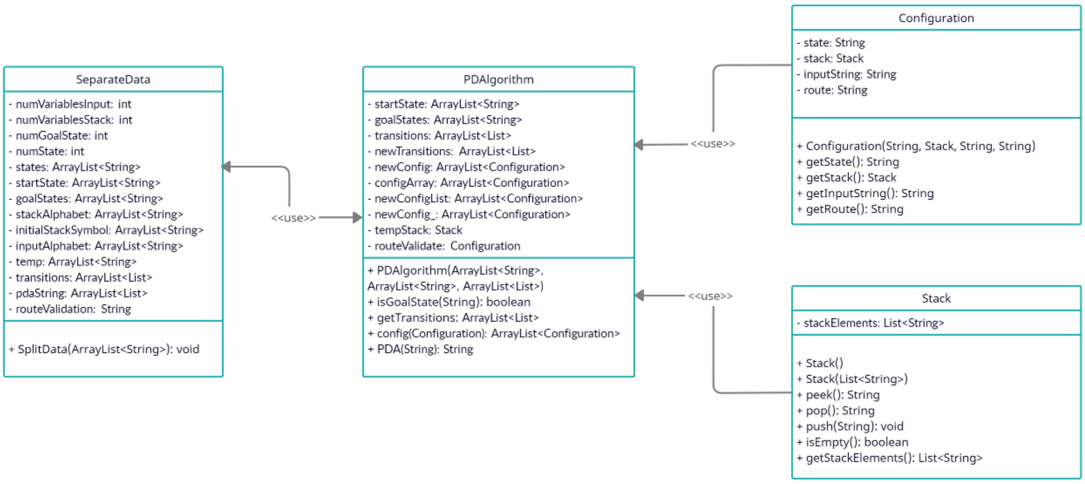
Methods of Configuration which are available from other components are:
public String getState() – To get the state
public Stack getStack() – To get the stack
public String getInputString() – To get the input
public String getRoute() – To get the route

Methods of PDAlgorithm which are available from other components are:
public void isGoalState(String state) – To check the state is goal state or not
public ArrayList<List> getTransitions(String state) – To create the new transitions
public String PDA(String input) – To apply the config method with routeValidate
public ArrayList<Configuration> config(Configuration configuration)– To apply the PDA Algorithm

Methods of SeparateData which are available from other components are:
public void SplitData(ArrayList<String> arrayList) – To separate the given ArrayList

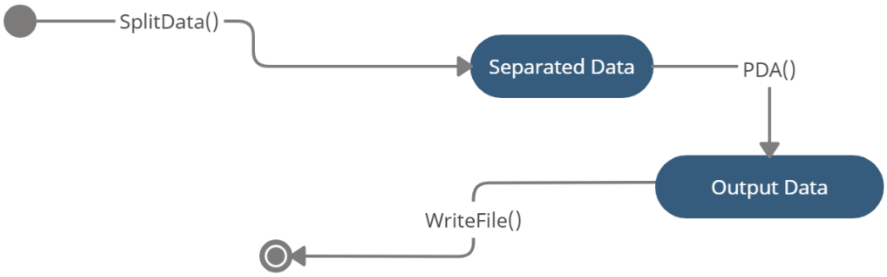
Methods of Stack which are available from other components are:
public String peek () – To get the top elements of stack
public String pop() – To pop the top elements of stack
public void push(String element) – To push the element to stack
public boolean isEmpty() – To get stack Is empty or not
public List<String> getStackElements() – To get the stack elements

3.2.2 Component design description


SeparateData
- numVariablesInput: int
- numVariablesStack: int
- numGoalState: int
- numState: int
- states: ArrayList<String>
- startState: ArrayList<String>
- goalStates: ArrayList<String>
- stackAlphabet: ArrayList<String>
- initialStackSymbol: ArrayList<String>
- inputAlphabet: ArrayList<String>
- temp: ArrayList<String>
- transitions: ArrayList<List>
- pdaString: ArrayList<List>
- routeValidation: String
+ SplitData(ArrayList<String>): void
<<use>>
PDAlgorithm
- startState: ArrayList<String>
- goalStates: ArrayList<String>
- transitions: ArrayList<List>
- newTransitions: ArrayList<List>
- newConfig: ArrayList<Configuration>
- configArray: ArrayList<Configuration>
- newConfigList: ArrayList<Configuration>
- newConfig_: ArrayList<Configuration>
- tempStack: Stack
- routeValidate: Configuration
+ PDAlgorithm(ArrayList<String>, ArrayList<String>, ArrayList<List>)
+ isGoalState(String): boolean
+ getTransitions: ArrayList<List>
+ config(Configuration): ArrayList<Configuration>
+ PDA(String): String
<<use>>
Configuration
- state: String
- stack: Stack
- inputString: String
- route: String
+ Configuration(String, Stack, String, String)
+ getState(): String
+ getStack(): Stack
+ getInputString(): String
+ getRoute(): String
<<use>>
Stack
- stackElements: List<String>
+ Stack()
+ Stack(List<String>)
+ peek(): String
+ pop(): String
+ push(String): void
+ isEmpty(): boolean
+ getStackElements(): List<String>

3.2.3 Workflows and algorithms

The Activity Diagram is given below:



3.3 The Whole Diagrams

3.3.1 Component interfaces

Methods of main which are available from other components are:
public static void main(String[] args) – To run the code

Methods of InputFile which are available from other components are:
public void ReadFile(String filePath) – To read the input .txt file

Methods of OutputFile which are available from other components are:
public void WriteFile(String routeValidation) – To write the output .txt file

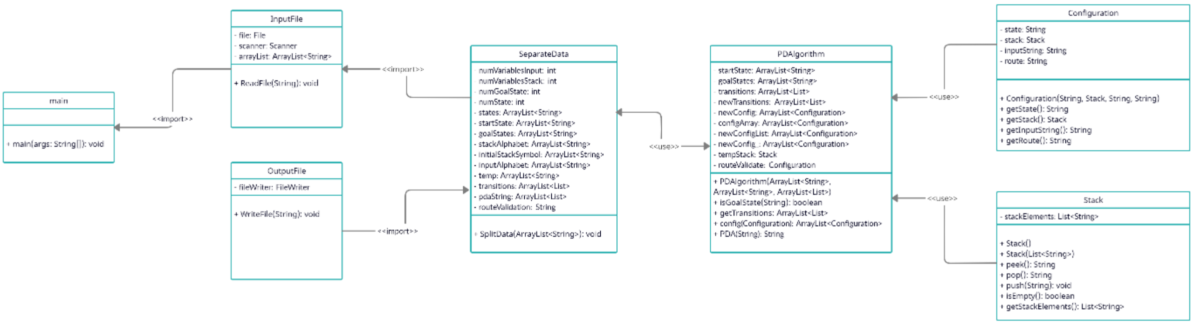
Methods of Configuration which are available from other components are:
public String getState() – To get the state
public Stack getStack() – To get the stack
public String getInputString() – To get the input
public String getRoute() – To get the route

Methods of PDAlgorithm which are available from other components are:
public void isGoalState(String state) – To check the state is goal state or not
public ArrayList<List> getTransitions(String state) – To create the new transitions
public String PDA(String input) – To apply the config method with routeValidate
public ArrayList<Configuration> config(Configuration configuration)– To apply the PDA Algorithm

Methods of SeparateData which are available from other components are:
public void SplitData(ArrayList<String> arrayList) – To separate the given ArrayList

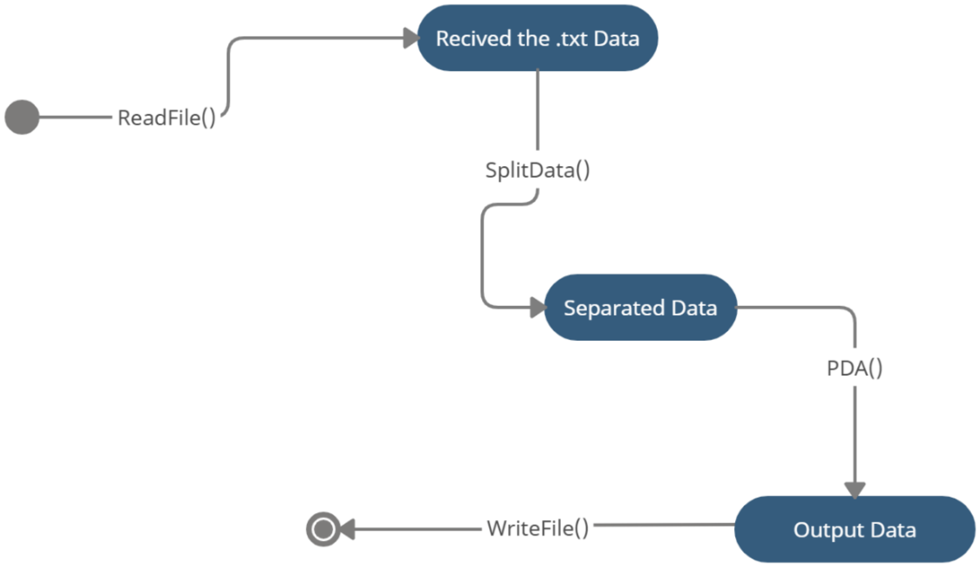
Methods of Stack which are available from other components are:
public String peek () – To get the top elements of stack
public String pop() – To pop the top elements of stack
public void push(String element) – To push the element to stack
public boolean isEmpty() – To get stack Is empty or not
public List<String> getStackElements() – To get the stack elements

3.3.2 Component design description


main
+ main(args: String[]): void
<<import>>
InputFile
- file: File
- scanner: Scanner
- arrayList: ArrayList<String>
+ ReadFile(String): void
<<import>>
OutputFile
- fileWriter: FileWriter
+ WriteFile(String): void
<<import>>
SeparateData
- numVariablesInput: int
- numVariablesStack: int
- numGoalState: int
- numState: int
- states: ArrayList<String>
- startState: ArrayList<String>
- goalStates: ArrayList<String>
- stackAlphabet: ArrayList<String>
- initialStackSymbol: ArrayList<String>
- inputAlphabet: ArrayList<String>
- temp: ArrayList<String>
- transitions: ArrayList<List>
- pdaString: ArrayList<List>
- routeValidation: String
+ SplitData(ArrayList<String>): void
<<use>>
PDAlgorithm
- startState: ArrayList<String>
- goalStates: ArrayList<String>
- transitions: ArrayList<List>
- newTransitions: ArrayList<List>
- newConfig: ArrayList<Configuration>
- configArray: ArrayList<Configuration>
- newConfigList: ArrayList<Configuration>
- newConfig_: ArrayList<Configuration>
- tempStack: Stack
- routeValidate: Configuration
+ PDAlgorithm(ArrayList<String>, ArrayList<String>, ArrayList<List>)
+ isGoalState(String): boolean
+ getTransitions: ArrayList<List>
+ config(Configuration): ArrayList<Configuration>
+ PDA(String): String
<<use>>
Configuration
- state: String
- stack: Stack
- inputString: String
- route: String
+ Configuration(String, Stack, String, String)
+ getState(): String
+ getStack(): Stack
+ getInputString(): String
+ getRoute(): String
<<use>>
Stack
- stackElements: List<String>
+ Stack()
+ Stack(List<String>)
+ peek(): String
+ pop(): String
+ push(String): void
+ isEmpty(): boolean
+ getStackElements(): List<String>

3.3.3 Workflows and algorithms

The Activity Diagram is given below:



4 COTS Identification

COTS (commercial of the shelf) libraries used in PDA are the following:

- Java, JDK 8, https://www.oracle.com/java/technologies/downloads/