27/12/2022

# Project 3 Design Report

## Turing Machine Simulator Atakan Bodur S018406

### Introduction

This document defines the design architecture of the ‘Turing Machine Simulator’ software and was created in order to explain how the software will implement required functionalities simulate a Turing Machine implementation.

### System Overview

Software uses Java language as the technology and be as object oriented as possible. The goal designing it as is to create an easy to edit and readable code, in order words, a sustainable software. The program will take a single format text document as the Turing Machine (provided its Alphabet, States, Start State and End State, and states’ Transitions), will read the file and assign appropriate values to their respected variables in the software, run the tape value given in the input and print out whether it is accepted or not to the console.

Graphical user interface, text

Description automatically generated

Every class and function are implemented on the Main.java class and works very simple; createTuringMachine() reads the input file with a Scanner in a switch statement. This switch statement is controlled by a List of Strings that projects the main consideration points in the input file. These Strings are

* "(number of variables in input alphabet)",
* "(the input alphabet)",
* "(number of variables in tape alphabet)",
* "(the tape alphabet)",
* "(number of states)",
* "(states)",
* "(start state)",
* "(accept state)",
* "(reject state)",
* "(tape input)",
* "(transitions)"

When these lines are read, switch statement fills up the Turing Machine’s fields. If the line is equal to “(end)”, it will call the TuringMachine.readTape() method.

This method will traverse start with the start state and progresses from there according to the given transitions by the input. If the Turing Machine wants to go right when there are no inputs left, simulator will add a blank input value. If the resulting state at any given time is reject state or the accept state, it will stop the loop and print out the result.

### Design

There are three main classes which represents the Turing Machine States and their transitions.

#### Turing Machine

Turing Machine class represents the Turing Machine and it hold the inputAlphabet, tapeAlphabet and the tape as CopyOnWriteArrayList<String> and the states as CopyOnWriteArrayList<State>. It also holds several other parameters which are the startState, acceptState, rejectState etc.

It holds methods that are;

* finalizeStates() which assigns startState, acceptState and rejectState to the TuringMachine.
* readTape() which reads the tape and simulates how a Turing Machine would behave.
* printTuring() which prints out the result in the desired outut format.

#### State

State class represent the states in the Turing Machine, and it has a symbol, and a Map<String, Skey> that holds the input value as the String and the Skey for the rule statements. For example:

0 b R q2 -> this line should be read as:

input,<what to write, which way to go, and to which state should be in>

#### Skey

Skey represents the rule statements for the given input value, it holds.

* what to write
* which way to go
* which state should be in as Strings.

### Design Considerations

Dependencies

There won’t be any external libraries used in this project.

#### Constraints

Software will abide to one format of input.