FEATURED ORIENTED PROGRAMMING (PR-1)

Akanksha Bansal (akankshabansal1827@gmail.com)
Gaurav Nanda (gaurav324@gmail.com)
Prateek Aggarwal (prat0318@gmail.com)

Objective of this assignment is to generate java source code for given the FSM.

Description of PART-1 and PART-2:

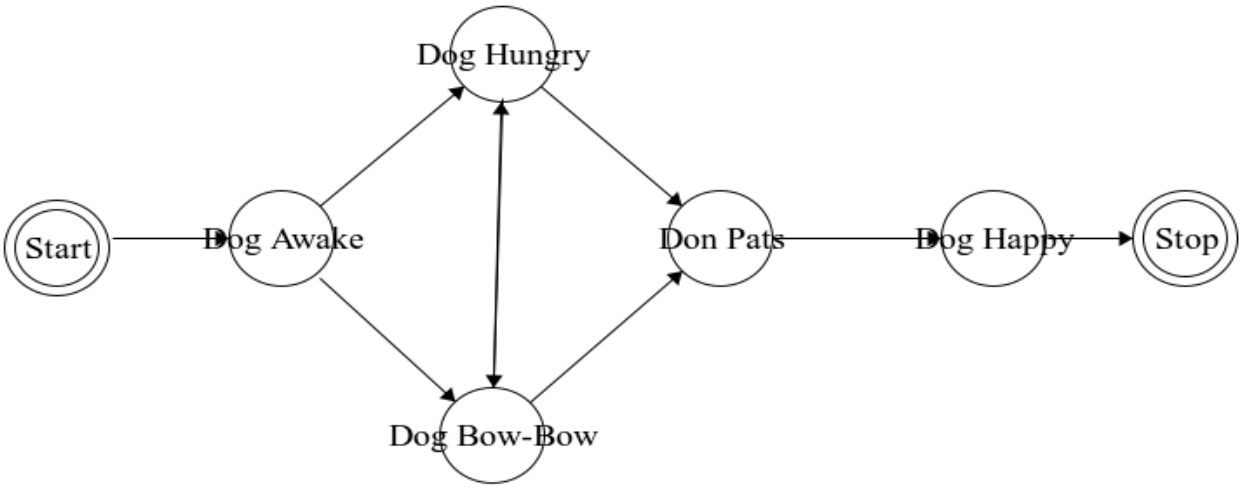
The aim of this part was to generate java source code from the given prolog file. A same FSM is implemented in two ways:
a. Generated class files for each of the states.
b. Generated functions for the various states of the FSM in a single class.

In our implementation we have Part1.vm that would generate different classes for various states in the FSM. It would also generate common.java and fsm1.java, which would be used by app.java to test the correctness of the generated code.

For the second problem, we have written Part2.vm that would generate fsm2.java and common.java. This vm file, instead of creating separate class files would create one java having different functions for various possible states.

Description of PART-3:

As solution to the part-3, we have defined DOG_FSM (dog_fsm.png) that indicates various states Professor's dog goes through the day ☺



In the process, we have written a prolog file (part3.pl) defining various states of the FSM. This file is then used in conjunction with part1.vm and part2.vm to generate java source code. We have also written app_part3.java (similar to app.java) to test our generated code. It traverses through the various states of the FSM and if successful, reaches the end point and prints the last terminal state that should be "**stop**".

This experiment would ensure that our velocity templates generated in part-1 and part-2 are general and can be used for any FSM.

Description of .zip submitted:

1. **run.script** (+x) (TO BE RUN ON UNIX/LINUX ENVIRONMENT)

Main script responsible for generating java source code, compiling it and then running it against app.java and app_part3.java to test our generated code.

2. MDELiteReduced/ app.java data.pl

Files provided.

3. part1/ part2/ part3/

part1/ and part2/ contain .vm files for generating fsm1.java and fsm2.java respectively. All other source and class files would also be generated inside the respective folders.

part3/ contains custom fsm diagram and its corresponding prolog file.

4. app_part3.java

Similar to app.java, it verifies the correctness of fsm1.java and fsm2.java. However, these java files would now have been generated using custom desgined FSM.

Execution

```
$ cd akanksha_gaurav_prateek
$ ./run.script
```