# Task 0

I tried to make task 0 implementation testable. So, I started with a failing unit test case, then worked on the actual implementation. Both unit test case and implementation gradually improved and I ended up having below test case:

/\*\*

\* Homework 6. Junit test case for Task 0 - Deterministic Finite State Automata Machine Implementation.

\* **@author** bsanchin

\*/

**public** **class** TestDFSASimulator **extends** TestCase {

**private** **final** String USER\_INPUT = "0010010101001100111111101111";

**public** **void** testSplittingForDefs() {

DFSASimulator sim = **new** DFSASimulator(USER\_INPUT);

*assertEquals*(USER\_INPUT, sim.getUserInput());

*assertEquals*("001001010100", sim.getStateDef());

*assertEquals*("00", sim.getAcceptingStateDef());

*assertEquals*("1111101111", sim.getCommand());

}

**public** **void** testStateCreationAndDefinition() {

DFSASimulator sim = **new** DFSASimulator(USER\_INPUT);

*assertEquals*("It should have 2 states.", 2, sim.getStatesCount());

*assertEquals*("Initial state should be the state one.",

1, sim.getCurrentState().getNumber());

sim.doOne();

*assertEquals*("On 1, first state should transition to first state",

1, sim.getCurrentState().getNumber());

sim.doZero();

*assertEquals*("On 0, first state should transition to second state",

2, sim.getCurrentState().getNumber());

sim.doOne();

*assertEquals*("On 1, second state should transition to second state",

2, sim.getCurrentState().getNumber());

sim.doZero();

*assertEquals*("On 0, second state should transition to first state",

1, sim.getCurrentState().getNumber());

}

**public** **void** testAcceptingState() {

DFSASimulator sim = **new** DFSASimulator(USER\_INPUT);

List<State> acceptingStates = sim.getAcceptingStates();

*assertEquals*("Only one accepting state should have been defined.",

1, acceptingStates.size());

State acceptingState = acceptingStates.get(0);

*assertEquals*("The only accepting state should be state 2.",

2, acceptingState.getNumber());

}

**public** **void** testStateTransition() {

// Accept scenario

// 0010010101001100111111101111

DFSASimulator sim = **new** DFSASimulator(USER\_INPUT);

sim.drive(); // Let the DFSA process user input

*assertEquals*("User input should have been accepted", **true**, sim.acceptedUserInput());

*assertEquals*("Should be in second state", 2, sim.getCurrentState().getNumber());

// Reject scenario

// 00100101010011001111111011110

sim = **new** DFSASimulator(USER\_INPUT + "0");

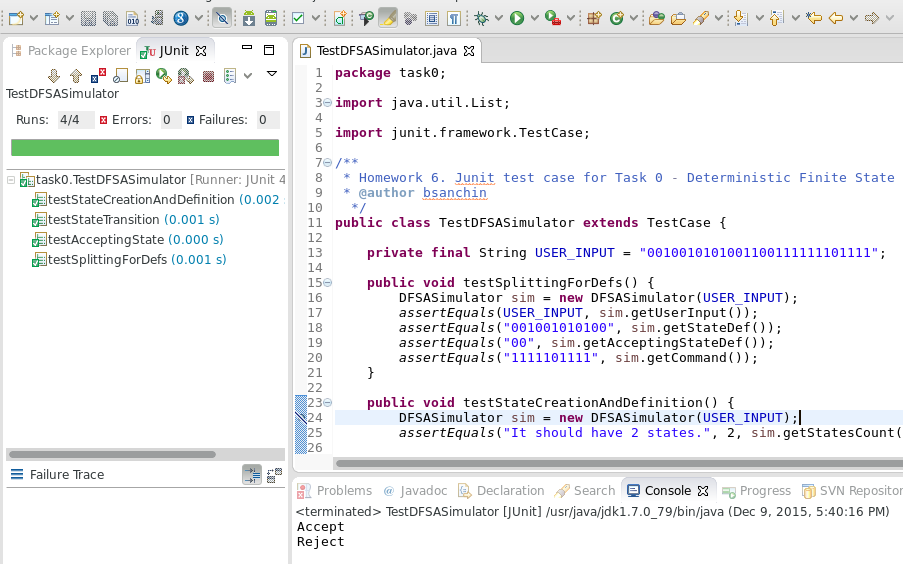
sim.drive(); // Let the DFSA process user input

*assertEquals*("User input should have been rejected", **false**, sim.acceptedUserInput());

*assertEquals*("Should be in first state", 1, sim.getCurrentState().getNumber());

}

}



*Picture 1. Junit test case for DFSASimulator in action.*

## Task 1

Source code/main method for task 1 is below.

**package** task123;

**public** **class** Turing1 {

**public** **static** **void** main(String[] args) {

// This machine will have one state.

// Note: There is an additional halt state.

// The values on the input tape are set to all B’s.

TuringMachine machine = **new** TuringMachine(1);

Transition one = **new** Transition('0', Transition.***RIGHT***, 0);

Transition two = **new** Transition('1', Transition.***RIGHT***, 0);

Transition three = **new** Transition('B', Transition.***LEFT***, 1);

machine.addTransition(0, '0', two);

machine.addTransition(0, '1', one);

machine.addTransition(0, 'B', three);

// The leftmost value of inTape will be

// placed under the read/write head.

String inTape = "11111100010101";

System.***out***.println(inTape);

String outTape = machine.execute(inTape);

System.***out***.println(outTape);

// Let's process another user input.

System.***out***.println();

// Do not forget to reset the machine!!!

machine.reset();

inTape = "0000011111";

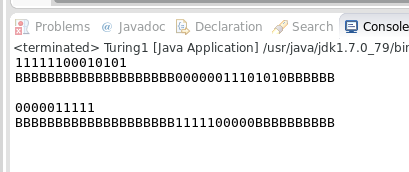
System.***out***.println(inTape);

outTape = machine.execute(inTape);

System.***out***.println(outTape);

}

}



*Picture 2. Program output from Task 1* implementation/main method/test driver*.*

## Task 2

Translation of pseudocode from Task 2 problem statement.

**package** task123;

**public** **class** Turing2 {

**public** **static** **void** main(String[] args) {

// This machine will have 7 states (numbered from 0 to 6)

// 7th state or state6 is the HALT\_STATE.

TuringMachine machine = **new** TuringMachine(6);

Transition q0\_0 = **new** Transition('B', Transition.***RIGHT***, 1);

Transition q1\_0 = **new** Transition('0', Transition.***RIGHT***, 1);

Transition q1\_1 = **new** Transition('1', Transition.***RIGHT***, 2);

Transition q2\_1 = **new** Transition('1', Transition.***RIGHT***, 2);

Transition q2\_0 = **new** Transition('1', Transition.***LEFT***, 3);

Transition q3\_0 = **new** Transition('0', Transition.***LEFT***, 3);

Transition q3\_1 = **new** Transition('1', Transition.***LEFT***, 3);

Transition q3\_B = **new** Transition('B', Transition.***RIGHT***, 0);

Transition q2\_B = **new** Transition('B', Transition.***LEFT***, 4);

Transition q4\_1 = **new** Transition('B', Transition.***LEFT***, 4);

Transition q4\_0 = **new** Transition('0', Transition.***LEFT***, 4);

Transition q4\_B = **new** Transition('0', Transition.***RIGHT***, 6);

Transition q0\_1 = **new** Transition('B', Transition.***RIGHT***, 5);

Transition q5\_0 = **new** Transition('B', Transition.***RIGHT***, 5);

Transition q5\_1 = **new** Transition('B', Transition.***RIGHT***, 5);

Transition q5\_B = **new** Transition('B', Transition.***RIGHT***, 6);

machine.addTransition(0, '0', q0\_0);

machine.addTransition(1, '0', q1\_0);

machine.addTransition(1, '1', q1\_1);

machine.addTransition(2, '1', q2\_1);

machine.addTransition(2, '0', q2\_0);

machine.addTransition(3, '0', q3\_0);

machine.addTransition(3, '1', q3\_1);

machine.addTransition(3, 'B', q3\_B);

machine.addTransition(2, 'B', q2\_B);

machine.addTransition(4, '1', q4\_1);

machine.addTransition(4, '0', q4\_0);

machine.addTransition(4, 'B', q4\_B);

machine.addTransition(0, '1', q0\_1);

machine.addTransition(5, '0', q5\_0);

machine.addTransition(5, '1', q5\_1);

machine.addTransition(5, 'B', q5\_B);

String inTape = "000100";

System.***out***.println(inTape);

String outTape = machine.execute(inTape);

System.***out***.println(outTape);

// Let's process another user input.

System.***out***.println();

// Do not forget to reset the machine!!!

machine.reset();

inTape = "0100";

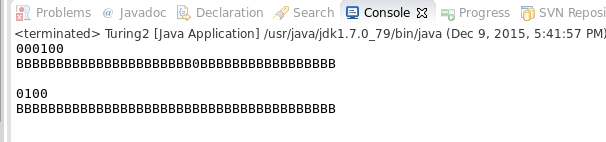
System.***out***.println(inTape);

outTape = machine.execute(inTape);

System.***out***.println(outTape);

}

}



Picture 3. Program output from Task 2 implementation/main method/test driver.

## Task 3

Solution/imlementation to Task 3.

**package** task123;

**public** **class** Turing3 {

**public** **static** **void** main(String[] args) {

// This machine will have 7 states (numbered from 0 to 6)

// 7th state or state6 is the HALT\_STATE.

TuringMachine machine = **new** TuringMachine(6);

Transition q0\_1 = **new** Transition('0', Transition.***RIGHT***, 1);

Transition q1\_1 = **new** Transition('1', Transition.***RIGHT***, 1);

Transition q1\_B = **new** Transition('B', Transition.***RIGHT***, 2);

Transition q2\_B = **new** Transition('1', Transition.***LEFT***, 3);

Transition q2\_1 = **new** Transition('1', Transition.***RIGHT***, 2);

Transition q3\_1 = **new** Transition('1', Transition.***LEFT***, 3);

Transition q3\_B = **new** Transition('B', Transition.***LEFT***, 4);

Transition q4\_1 = **new** Transition('1', Transition.***LEFT***, 4);

Transition q4\_0 = **new** Transition('0', Transition.***RIGHT***, 0);

Transition q0\_B = **new** Transition('B', Transition.***LEFT***, 5);

Transition q5\_0 = **new** Transition('1', Transition.***LEFT***, 5);

Transition q5\_B = **new** Transition('B', Transition.***RIGHT***, 6);

machine.addTransition(0, '1', q0\_1);

machine.addTransition(1, '1', q1\_1);

machine.addTransition(1, 'B', q1\_B);

machine.addTransition(2, 'B', q2\_B);

machine.addTransition(2, '1', q2\_1);

machine.addTransition(3, '1', q3\_1);

machine.addTransition(3, 'B', q3\_B);

machine.addTransition(4, '1', q4\_1);

machine.addTransition(4, '0', q4\_0);

machine.addTransition(0, 'B', q0\_B);

machine.addTransition(5, '0', q5\_0);

machine.addTransition(5, 'B', q5\_B);

String inTape = "";

String outTape = "";

**for** (**int** i = 0; i < 20; i++) {

System.***out***.println(inTape);

// Replace old tape with a new tape with 80 cells.

machine.reset(80);

outTape = machine.execute(inTape);

System.***out***.println(outTape);

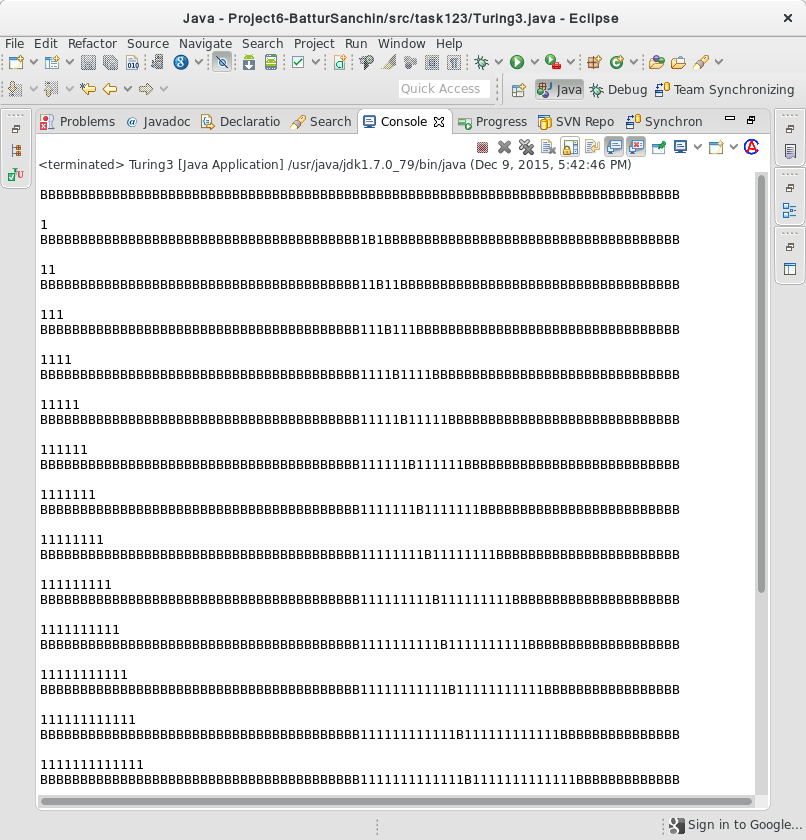
System.***out***.println();

inTape = inTape + "1";

}

}

}



*Picture 4. Program output from Task 3 implementation/main method/test driver.*