You should skim through <https://en.wikipedia.org/wiki/Finite-state_machine> to get a basic understanding of finite state machines. The basic advantage is to prevent the repetition of code and to make the programs more easy to read. In addition, each of these states have been tested so the likelihood that they are no longer functioning is much less.

1. You need to have statemachine.java in your teamcode so that other files may reference it.
2. The ConceptStateMachine.java is a file to demonstrate how to use state machines. In order to use any state machines, make sure you have all the states you want in your Team Code Folder and create new objects of each state.
3. Once you have created an object of each state, initialize the state with the required information such as power or distance values.
4. You need to set what the next state will be after the first state is finished. This can be done by calling the setNextState(State nextState) on the state you want to establish the transition for.

Example Code:

import org.firstinspires.ftc.teamcode.StateMachine; //necessary

import org.firstinspires.ftc.teamcode.StateMachine.State; //necessary

import java.util.ArrayList;

DcMotor leftFront;

DcMotor rightFront;

DcMotor leftBack;

DcMotor rightBack;

rangeCalibrationState rangeState;

distanceMoveState distanceState;

timeState ts;

private StateMachine machine;

ArrayList<DcMotor> motors = new ArrayList<DcMotor>(); This is an arraylist that you pass into the state with all the motors in a specific order

motors.add(rightFront);

motors.add(leftFront);

motors.add(rightBack);

motors.add(leftBack);

rangeState = new rangeCalibrationState(motors, mrrs, 3.0); Initializing the individual states with the various parameters, please look at the code to see which parameters

ts = new timeState(0,0,motors, "forward");

// distanceState.setNextState(rangeState);

rangeState.setNextState(ts); Setting the following state of each state. This means that once the transition of the state is executed, the ts state will be called next

ts.setNextState(null);

}

@Override

public void start(){

machine = new StateMachine(rangeState); Initializing the state machine with the initial state to be rangeState

}

@Override

public void loop() {

machine.update(); Updating the state machine and state transitions every loop

}

}