**Purpose:** Provide a layer over the FTC motor library for our code to use. The intent of this is to future-proof the code, so if any libraries change we only have to change the code once, in this class. Also useful for creating stub methods while the rest of the code is in development for testing purposes.

|  |  |
| --- | --- |
| **Priority:** Very High | **Reason:** The robot probably isn’t going anywhere without drive motors. |

**Primary Programmers:**

* Adam

**Public Constants (public static final):**

* int? MAX\_POWER
  + Purpose: The maximum value for motor power. Useful for algorithms including motor power scaling and course correction.
* MODE\_NORMAL
  + Purpose: Indicate that the motor should behave normally
* MODE\_REVERSE
  + Purpose: Indicate that the motor power should be reversed (multiply the new power by -1 before setting it)

**Constructors (called when an object instance is created):**

* public Motor(2 arguments)
  + Must Initialize: Stuff needed to initialize a motor in the FTC libraries, and store info needed to access that motor in instance variables in the object (for example, storing the port number in a variable).
  + Arguments:
    - <type> port
      * The port or location the motor is connected at
    - mode
      * One of the public constants starting with ‘MODE\_’, reccomended to just call ‘setMode(mode)’ in this constructor

**Interface Instance Methods (used on an instance of this class):**

* public synchronized void setPower(1 argument)
  + Purpose: Adjust the motor power, and start or stop it when necessary.
  + Priority: Very High (cannot move the motor without it)
  + Arguments:
    - int? power
      * The power to set the motor to (see also: MAX\_POWER)
  + Returns: N/A
* public synchronized void setMode(1 arguments)
  + Purpose: Change how the motor is running (for example reverse it).
  + Priority: High (need to be able to reverse a drive motor that is rotating opposite what it needs to)
  + Arguments:
    - mode
      * One of the public constants which starts with ‘MODE\_’, adjusts anything that needs changed depending on what mode was specified
* public synchronized void stopMotor(0 arguments)
  + Purpose: Stop the motor, syntactically equivalent to ‘setPower(0)’.
  + Priority: Medium (only a shortcut method, but all it needs to do is call setPower(0))
  + Arguments: N/A
  + Returns: N/A
* public synchronized int? getCurrentPower(0 arguments)
  + Purpose: Get the current power level of the motor (with MAX\_POWER as the maximum).
  + Priority: Low (may not be highly used, but nice to have in case someone does need it)
  + Arguments: N/A
  + Returns: (int?) the current motor power