**Purpose:** Provide an abstract way to handle the motors of the drive train without having to worry about the individual motors. Intended to encapsulate drive motor access, so programmers usually only have to set the speed per side instead of per motor, with the DriveSystem class handling all the specifics. Any changes made to the physical robot that affect the drive train should be accounted for in this class if possible to prevent ripple effects (changes in classes/methods that cause problems in other classes that use them).

|  |  |
| --- | --- |
| **Priority:** Medium-High | **Reason:** Robot can move without this, but it should be created and integrated early on otherwise it will take longer to implement. |

**Primary Programmers:**

1. Adam

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

* <required type if any> <NAME>
  + Purpose: <1-4 sentences> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque in turpis posuere, pretium velit in, tempor turpis.

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

* (0 arguments)
  + Must Initialize: All motors/devices part of the drive system (use classes in ‘com.lmrobotics.devices’ like the ‘Motor’ class).
  + Arguments: N/A

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

* public <other tags> <return type> <method name>(<#> arguments)
  + Purpose: <1-4 sentences> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque in turpis posuere, pretium velit in, tempor turpis.
  + Priority: <Same options as the class priority> (<reason>)
  + Arguments:
    - <type> <arg1>
      * What to put there (see also: <something else in this file>)
  + Returns: (<type>) <what data is returned>
    - <Can go onto a separate bullet(s) if it would be cleaner>
* public synchronized void setPower(1 argument)
  + Purpose: Adjust the motor power, and start or stop it when necessary.
  + Priority: Very High (cannot start the motor without it)
  + Arguments:
    - int power
      * The power to set the motor to (see also: MAX\_MOTOR\_POWER)
  + Returns: N/A
* 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

**Interface Static Methods (used without an instance of a class):**

* public static synchronized long getCurrentTime(1 argument)
  + Purpose: Get how much time has elapsed, syntactically equivalent to ‘System.currentTimeMillis() - someothertime’
  + Priority: Low (prevents other programmers from having to do this manually every time)
  + Arguments:
    - long start
      * when this ‘timer’ started (specify 0 to get an initial start time)
  + Returns: (long) the current system time in milliseconds