**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
2. ???

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

* None

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

* None required by interface

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

* public synchronized void setLeft(1 argument)
  + Purpose: Sets the power of all the drive motors on the left side of the robot.
  + Priority: High (Sets all the left drive motors)
  + Arguments:
    - int power
      * The power to set to (See also: Motor.MAX\_POWER)
  + Returns: N/A
* public synchronized void setRight(1 argument)
  + Purpose: Set the power of all right-side drive motors
  + Priority: High (sets the power of all right motors)
  + Arguments:
    - int power
      * The power to set the motors to
  + Returns: N/A
* public synchronized void setPower(2 arguments)
  + Purpose: Set both left and right sides, syntactically equivalent to
    - setLeft(leftPower);
    - setRight(rightPower);
  + Priority: Medium (only a shortcut method, but all it needs to do is call setLeft() and setRight())
  + Arguments:
  + int leftPower
    - The power for the left drive motors
  + Int rightPower
    - The power for the right drive motors
  + Returns: N/A

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

* None