LeJOS basic commands reference

For printing on the LCD screen:

`LCD.drawString(String str, int x, int y)`	Displays the provided string on the LCD screen, starting at position (x, y).
`LCD.drawInt(int n, int x, int y)`	Displays the integer `n` on the LCD screen, starting at position (x, y).
`LCD.clear()`	Clears the LCD screen, removing whatever has been printed.

Methos for working with the buttons on the brick. Note that `Button.waitForPress()` is useful as the last command of a program, as otherwise the program will terminate immediately and clear anything which was displayed on the LCD.

`Button.waitForPress()`	Waits for any button on the brick to be pressed before continuing with the program. Returns an `int` which represents which button was pressed (ask for details).
`Button.readButtons()`	Returns an `int` which is 0 if no buttons are currently pressed down, and greater than 0 if any buttons are currently pressed down. It is possible to use this value to determine which buttons in particular are currently pressed, ask for details.

You can access the three motors via the constants `Motor.A`, `Motor.B`, and `Motor.C`, which are instances of the class `NXTRegulatedMotor`. You can tell a motor to do things using the following methods, so for example to tell motor A to start running forward, you would write `Motor.A.forward()`.

`forward()`	Start the motor rotating forward.
`backward()`	Start the motor rotation backward.
`stop()`	Quickly stop the motor rotating.
`getTachoCount()`	Returns the motor angle in degrees.
`setSpeed(int speed)`	Sets the speed in degrees per second, up to a max of around 1000.
`rotate(int angle)`	Rotates the motor the specified angle in degrees.
`rotateTo(int angle)`	Rotates the motor until it's tachometer count (as returned by `getTachoCount()`) equals the specified angle in degree.
`rotate(int angle, true)`	The same as the other `rotate()`, but returns immediately, i.e. the next command in the program runs immediately rather than waiting for the rotation to complete.
`rotateTo(int angle, true)`	The same as the other `rotateTo`, but returns immediately.
`isMoving()`	Returns `true` if the motor is currently moving.

There are some other, more obscure motor methods. These probably won't be as useful, but are listed here for reference.

`resetTachoCount()`	Resets the tachometer count, as returned by `getTachoCount`, to zero.
`setAcceleration(int accel)`	Sets the rate at which the motor changes speed, in degrees per second per second. Try using a small value like 200 to have your robot smoothly accelerate or decelerate.
`getSpeed()`	Returns the speed in degrees per second, as set by `setSpeed()`.
`getActualSpeed()`	Returns the speed that the motor is actually currently moving, in degrees per second.
`getAcceleration()`	Returns the acceleration in degrees per second per second, as set by `setAcceleration()`.
`getLimitAngle()`	Gets the angle to which the motor is currently rotating, after a call to `rotate()` or `rotateTo()`.
`isRotating()`	Returns `true` if the motor is currently moving due to a `rotate()` or `rotateTo()` command. Compare to `isMoving()`, which returns `true` if the motor is moving for any reason.
`flt()`	Stop the motor gradually. Like `stop()`, but allows the motor to stop on its own rather than using power to stop it.