```
/* Copyright (c) 2014 Qualcomm Technologies Inc

All rights reserved.

*/

package com.qualcomm.ftcrobotcontroller.opmodes;

//============== Import Core Functions So They Can Be Used Later In The Program ====================

import com.qualcomm.robotcore.eventloop.opmode.OpMode;
import com.qualcomm.robotcore.hardware.DcMotor;
import com.qualcomm.robotcore.hardware.Servo;
import com.qualcomm.robotcore.util.Range;
import com.qualcomm.robotcore.hardware.TouchSensor;

/**
 * TeleOp Mode
 * <p>
 * Enables control of the robot via the gamepads
 */
public class TeleOP extends OpMode {

//============================== Define Drive Motors Variables ====================================
    public DcMotor motorFR;
    public DcMotor motorFL;
    public DcMotor motorBR;
    public DcMotor motorBL;

//============================== Define Servo Motors Variables ====================================

    // Servo Motor "Java" Names
    public Servo CServoLift;
    public Servo CServoMS;
    public Servo CServoIntS;
    public Servo ServoRF;
    public Servo ServoLF;

    // position of the lift servo.
    double CServoLiftPosition;
```

```
    // amount to change the lift servo position.
    double CServoLiftDelta = 0.1;

    // position of the main sweeper servo.
    double CServoMSPosition;
    // amount to change the main sweeper servo position.
    double CServoMSDelta = 0.1;

    // position of the internal sweeper servo.
    double CServoIntSPosition;
    // amount to change the internal sweeper servo position.
    double CServoIntSDelta = 0.1;

    // position of the right flipper servo.
    double ServoRFPosition;
    // amount to change the right flipper servo position.
    double ServoRFDelta = 0.5;

    // position of the left flipper servo.
    double ServoLFPosition;
    // amount to change the right flipper servo position.
    double ServoLFDelta = 0.5;

//============================== Define Limit Switch Variables ======================================

    // Limit Switch "Java" Names
    public TouchSensor LTLimit;
    public TouchSensor LBLimit;

//================================ Define the Name of This OpMode ====================================

    /**
     * Constructor
     */
    public TeleOP() {

    }

//============================== Initialize the Variables ========================================
```

```
    /*
     * Code to run when the op mode is first enabled goes here
     *
     * @see com.qualcomm.robotcore.eventloop.opmode.OpMode#start()
     */
    @Override
    public void init() {

        /*
         * Use the hardwareMap to get the dc motors and servos by name. Note
         * that the names of the devices must match the names used when you
         * configured your robot and created the configuration file.
         */

        /*
         *   There are four motors "motor_1", "motor_2", "motor_3", "motor_4"
         *   "motor_1" is on the front right side of the bot.
         *   "motor_2" is on the front left side of the bot.
         *   "motor_3" is on the rear right side of the bot.
         *   "motor_4" is on the rear left side of the bot.
         *
         *
         *
         * Motor Layout
         *
         *
         *         Front of Robot
         *           2--------1
         *           |        |
         *           |        |
         *           |        |
         *           |        |
         *           |        |
         *           4--------3
         */


//=================== Match Drive Motors to Hardware Configuration Names ============================

        motorFR = hardwareMap.dcMotor.get("motor_1");
```

```
        motorFL = hardwareMap.dcMotor.get("motor_2");
        motorBR = hardwareMap.dcMotor.get("motor_3");
        motorBL = hardwareMap.dcMotor.get("motor_4");

        motorFR.setDirection(DcMotor.Direction.REVERSE);
        motorBR.setDirection(DcMotor.Direction.REVERSE);

//=============== Match Servo Motors "Java" Names to Hardware Configuration Names ==================

        CServoLift = hardwareMap.servo.get("LiftMotor"); // channel 5
        CServoMS = hardwareMap.servo.get("MSMotor");     // channel 3
        CServoIntS = hardwareMap.servo.get("ISMotor");   // channel 2
        ServoRF = hardwareMap.servo.get("RFMotor");      // channel 1
        ServoLF = hardwareMap.servo.get("LFMotor");      // channel 4

//======================== Set the Starting Positions for the Servo Motors ==========================

        // assign the starting position of the Lift Motor (CServoLift)
        CServoLiftPosition = 0.5;  // 0.5 Stops the Motor

        // assign the starting position of the Main Sweeper Motor (CServoMS)
        CServoMSPosition = 0.5;  // 0.5 Stops the Motor

        // assign the starting position of the Internal Sweeper Motor (CServoIntS)
        CServoIntSPosition = 0.5;  // 0.5 Stops the Motor

        // assign the starting position of the Right Flipper Motor (ServoRF)
        ServoRFPosition = 1;  // 0 = 0 degree position and 1 = 180 degree position

        // assign the starting position of the Left Flipper Motor (ServoLF)
        ServoLFPosition = 0;  // 0 = 0 degree position and 1 = 180 degree position

//================ Match Limit Switch "Java" Names to Hardware Configuration Names ==================

        LTLimit = hardwareMap.touchSensor.get("LiftLimit_T");
        LBLimit = hardwareMap.touchSensor.get("LiftLimit_B");

    }

//==================== Check GamePad Controllers and Tell the Robot what to Do ======================
```

```
    /*
     * This method will be called repeatedly in a loop
     *
     * @see com.qualcomm.robotcore.eventloop.opmode.OpMode#run()
     */
    @Override
    public void loop() {

//=================================== GamePad 1 Controls ========================================

        /*
         *
         * Gamepad 1 controls the drive motors via the left and right sticks
         *
         *
         *
         *                                  Front of Gamepad
         *
         *
         *              Left Joystick                              Right Joystick
         *                                        |
         *          Moves Left Side Forward       |          Moves Right Side Forward
         *                   ^ ^                  |                   ^ ^
         *                  -----                 |                  -----
         *                |       |               |                |       |
         *             < |         | >           |             < |         | >
         *             < |         | >           |             < |         | >
         *                |       |              |                |       |
         *                  -----                 |                  -----
         *                   v v                  |                   v v
         *          Moves Right Side Back         |          Moves Right Side Back
         *
         *
         */

//============================ Tank Drive Control ================================

        // note that if y equal 1 then joystick is pushed all of the way forward.
        float leftstick = gamepad1.left_stick_y;
```

```
        float rightstick = gamepad1.right_stick_y;

        // clip the rightstick/leftstick values so that the values never exceed +/- 1
        rightstick = Range.clip(rightstick, -1, 1);
        leftstick = Range.clip(leftstick, -1, 1);

        // scale the joystick value to make it easier to control
        // the robot more precisely at slower speeds.
        rightstick = (float) scaleInput(rightstick);
        leftstick = (float) scaleInput(leftstick);

        // write the values to the motors
        motorFR.setPower(rightstick);
        motorFL.setPower(leftstick);
        motorBR.setPower(rightstick);
        motorBL.setPower(leftstick);

//=================================== GamePad 2 Controls ========================================

        /*
         *
         * Gamepad 2 controls the Lift, Main Sweeper, Internal Sweeper and
         * the Right and Left Flipper Servo Motors
         *
        */

//============================== Lifter Control ==================================

        // Tell the lift servo to move up.
        if (gamepad2.dpad_up) {
            // if the D-Pad Up button is pushed on gamepad2, increment the position of
            // the lift servo.  Note This is a continuous servo so it will run until the stop
            // button is pressed.
            CServoLiftPosition += CServoLiftDelta;
        }

//================== Top Limit Switch ====================

        // Tell the lift servo to stop if the Top Limit Switch is pressed.
        if(LTLimit.isPressed()) {
```

```
            //Stop the lift motor if the Top Limit Switch is pressed.
            CServoLiftPosition = 0.5;
        } else {
            //Keep moving upwards if the Top Limit Switch is NOT pressed
            CServoLiftPosition += 0;
        }

        // Tell the lift servo to move down.
        if (gamepad2.dpad_down) {
            // if the D-Pad Down button is pushed on gamepad2, decrement the position of
            // the lift servo.  Note This is a continuous servo so it will run until the stop
            // button is pressed.
            CServoLiftPosition -= CServoLiftDelta;
        }

//=================== Bottom Limit Switch ====================

        // Tell the lift servo to stop if the Top Limit Switch is pressed.
        if(LBLimit.isPressed()) {
            //Stop the lift motor if the Bottom Limit Switch is pressed.
            CServoLiftPosition = 0.5;
        } else {
            //Keep moving upwards if the Top Limit Switch is NOT pressed
            CServoLiftPosition -= 0;
        }

        // Tell the lift servo to stop
        if (gamepad2.y) {
           // if the Y button is pushed on gamepad2, stop the lift servo
           CServoLiftPosition = 0.5;
        }

        // clip the position value so that it never exceeds the allowed range.
        CServoLiftPosition = Range.clip(CServoLiftPosition, .2, .8);

        // write the position value to the lift servo
        CServoLift.setPosition(CServoLiftPosition);

//========================= Main Sweeper Control ================================
```

```
        // Tell the main sweeper servo to sweep blocks in (CW?).
        if (gamepad2.right_bumper) {
            // if the right bumper button is pushed on gamepad2, increment the position of
            // the main sweeper servo.  Note This is a continuous servo so it will run
            // until the stop button is pressed.
            CServoMSPosition += CServoMSDelta;
        }

        // Tell the main sweeper servo to sweep blocks out (CCW?).
        if (gamepad2.left_bumper) {
            // if the left bumper button is pushed on gamepad2, decrement the position of
            // the main sweeper servo.  Note This is a continuous servo so it will run
            // until the stop button is pressed.
            CServoMSPosition -= CServoMSDelta;
        }

        // Tell the main sweeper servo to stop
        if (gamepad2.y) {
           // if the Y button is pushed on gamepad2, stop the main sweeper servo
           CServoLiftPosition = 0.5;
        }

        // clip the position values so that they never exceed their allowed range.
        CServoMSPosition = Range.clip(CServoMSPosition, .2, .8);

        // write position values to the mains sweeper servo
        CServoMS.setPosition(CServoMSPosition);

//======================== Internal Sweeper Control ================================

        // Tell the internal sweeper servo to sweep right (CW?).
        if (gamepad2.dpad_right) {
            // if the D-Pad right button is pushed on gamepad2, increment the position of
            // the internal sweeper servo.  Note This is a continuous servo so it will run
            //  until the stop button is pressed.
            CServoIntSPosition += CServoIntSDelta;
        }

        // Tell the internal sweeper servo to sweep left (CCW?).
        if (gamepad2.dpad_left) {
```

```
            // if the D-Pad left button is pushed on gamepad2, decrement the position of
            // the internal sweeper servo.  Note This is a continuous servo so it will run
            // until the stop button is pressed.
            CServoIntSPosition -= CServoIntSDelta;
        }

        // Tell the internal sweeper servo to stop
        if (gamepad2.y) {
           // if the Y button is pushed on gamepad2, stop the internal sweeper servo
           CServoIntSPosition = 0.5;
        }

        // clip the position value so that it never exceeds the allowed range.
        CServoIntSPosition = Range.clip(CServoIntSPosition, .2, .8);

        // write the position value to the internal sweeper servo
        CServoIntS.setPosition(CServoIntSPosition);

//======================== Flipper Control ================================

        // Tell the flipper servos to flip up.
        if (gamepad2.a) {
            // if the A button is pushed on gamepad2, set the position of
            // the right and left flipper servos to 90 degrees.
            ServoRFPosition -= ServoRFDelta;
            ServoLFPosition += ServoLFDelta;
        }

        if (gamepad2.b) {
            // if the B button is pushed on gamepad2, set the position of
            // the right and left flipper servos (RF) to 180 and (LF) to 0 degrees.
            ServoRFPosition += ServoRFDelta;
            ServoLFPosition -= ServoLFDelta;
        }

        // clip the position values so that they never exceed the allowed ranges.
        ServoRFPosition = Range.clip(ServoRFPosition, 0.0, 1.0);  // change these when known
        ServoLFPosition = Range.clip(ServoLFPosition, 0.0, 1.0);  // change these when known

        // write the position values to the flipper servos
```

```
        ServoRF.setPosition(ServoRFPosition);
        ServoLF.setPosition(ServoLFPosition);

//================= Telemetry (These Values are Displayed on the Drive Station) ====================
        /*
         * Send telemetry data back to driver station. Note that if we are using
         * a legacy NXT-compatible motor controller, then the getPower() method
         * will return a null value. The legacy NXT-compatible motor controllers
         * are currently write only.
         */

        telemetry.addData("Text", "*** Robot Data***");
        telemetry.addData("leftstick", "PWR: " + String.format("%2f", leftstick));
        telemetry.addData("rightstick", "PWR: " + String.format("%2f", rightstick));
        telemetry.addData("LiftServo", "Postion:  " + String.format("%.2f", CServoLiftPosition));

        }
    /*
     * Code to run when the op mode is first disabled goes here
     *
     * @see com.qualcomm.robotcore.eventloop.opmode.OpMode#stop()
     */
     @Override
     public void stop() {

            }

//========== Scale Settings (Round off the Stick Position Values for Drive Speed) ==================
    /*
     * This method scales the joystick input so for low joystick values, the
     * scaled value is less than linear.  This is to make it easier to drive
     * the robot more precisely at slower speeds.
     */
             double scaleInput(double dVal)  {
            double[] scaleArray = { 0.0, 0.05, 0.09, 0.10, 0.12, 0.15, 0.18, 0.24,
                    0.30, 0.36, 0.43, 0.50, 0.60, 0.72, 0.85, 1.00, 1.00 };

            // get the corresponding index for the scaleInput array.
            int index = (int) (dVal * 16.0);
```

```
            // index should be positive.
            if (index < 0) {
                index = -index;
            }

            // index cannot exceed size of array minus 1.
            if (index > 16) {
                index = 16;
            }

            // get value from the array.
            double dScale = 0.0;
            if (dVal < 0) {
                dScale = -scaleArray[index];
            } else {
                dScale = scaleArray[index];
            }

            // return scaled value.
            return dScale;
        }

}
```