```
//This organized the app and places the files in the teamcode section.
//This would be important if you wanted to do some more complicated
//programming and wanted access to a environment such as Android Studio.

package org.firstinspires.ftc.teamcode;

//These next few statements add specific coding to this class so that you
//can access the build-in methods of these other classes.

import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
import com.qualcomm.robotcore.hardware.DcMotor;

//This section is important because it determines how to organize
//the class within the context of the robot driver station app on the phone.
//It is normal to get nervous during competition and it is also common to launch
//wrong program when stressed. It is usually best to have a single opMode for
//teleOp as you can't make the wrong choice if there is only a single choice.

@TeleOp(name = "Whatever", group = "Intermediate")
//@Disabled

public class Rookie_With_Camera extends LinearOpMode {


   //These statements create instances of the motor class so that you
   //can control the motors.
   public DcMotor driveFR = null;
   public DcMotor driveBR = null;
```

```
public DcMotor driveFL = null;
public DcMotor driveBL = null;

public void runOpMode() {

   //mapping software objects to hardware objects through
   //configuration file.
   //The left and right is based on observing
   //the robot from above and facing the forward direction.

   //Your robot cam configured so that you could start working right away.
   //the left motor does into the 0 motor port. It is called driveLeft
   //in the configuration file.
   //the right motor does into the 1 motor port. It is called driveRight
   //in the configuration file.

   //Using conventions makes it easier to operate your robot. Some
   //teams perfer more language, such as motorDriveLeft. In this case,
   //I kept things simple for you so that you can focus your time and
   //energy in getting up to speed with the robot.

   driveFR = hardwareMap.get(DcMotor.class, "driveFR");
   driveBR = hardwareMap.get(DcMotor.class, "driveBR");
   driveFL = hardwareMap.get(DcMotor.class, "driveFL");
   driveBL = hardwareMap.get(DcMotor.class, "driveBL");


      waitForStart();
```

```
double power=.25;

//driveStrafeRight(power);
//sleep(3500);
//driveAxialForward(power);
//sleep(5500);
//driveStrafeLeft(power);
//sleep(4000);
//driveAxialBackward(power);
//sleep(1000);
//RotateRight(power);
//sleep(750);
//driveStrafeLeft(power);
//sleep(1000);
//driveAxialForward(power);
//sleep(2000);



//driveAllStop();

   while (opModeIsActive()) {
      if(gamepad1.dpad_up){
         driveAxialForward(power);
      }
      else if(gamepad1.dpad_down){
         driveAxialBackward(power);
      }
      else if(gamepad1.dpad_left){
```

```
            driveStrafeLeft(power);
        }
        else if(gamepad1.dpad_right){
            driveStrafeRight(power);
        }
        else if(gamepad1.left_bumper){
            driveRotateLeft(power);
        }
        else if(gamepad1.right_bumper){
            driveRotateRight(power);
        }
        else
        {
            driveAllStop();
        }
    }

}
  //This next section contains examples of methods. This is to inspire
   //you to want to learn more about programming and Java specifically.

   //When driving forward, the left motor is positive and the right motor
   //is negative. Motors generally rotate counter clockwise, so the left
   //motor spins in the correct direction to go forward.
   //The right motor would spin in the opposite direction, cause the
   //robot to spin. This is the desired behavio for other situations
   //but not this one.
   //The right motor is set to - power, which rotates it in the opposite
   //direction.
```

```
    //All of the other movements can be derived from this observations.



public void driveAxialForward(double power) {
    driveFR.setPower(-power);
    driveBR.setPower(power);

    driveFL.setPower(power);
    driveBL.setPower(-power);
}
public void driveAxialBackward(double power) {
    driveFR.setPower(power);
    driveBR.setPower(-power);

    driveFL.setPower(-power);
    driveBL.setPower(power);
}
public void driveRotateLeft(double power) {
    driveFR.setPower(power);
    driveBR.setPower(-power);

    driveFL.setPower(power);
    driveBL.setPower(-power);
}
public void driveRotateRight(double power) {
    driveFR.setPower(-power);
    driveBR.setPower(power);
    driveFL.setPower(-power);
```

```
        driveBL.setPower(power);
    }
    public void driveStrafeLeft(double power) {
        driveFR.setPower(power);
        driveBR.setPower(power);
        driveFL.setPower(power);
        driveBL.setPower(power);
    }
    public void driveStrafeRight(double power) {
        driveFR.setPower(-power);
        driveBR.setPower(-power);
        driveFL.setPower(-power);
        driveBL.setPower(-power);
    }
    public void driveAllStop() {
        //driveLeft.setPower(0);
        driveFR.setPower(0);
        driveBR.setPower(0);
        driveFL.setPower(0);
        driveBL.setPower(0);
    }
}
```