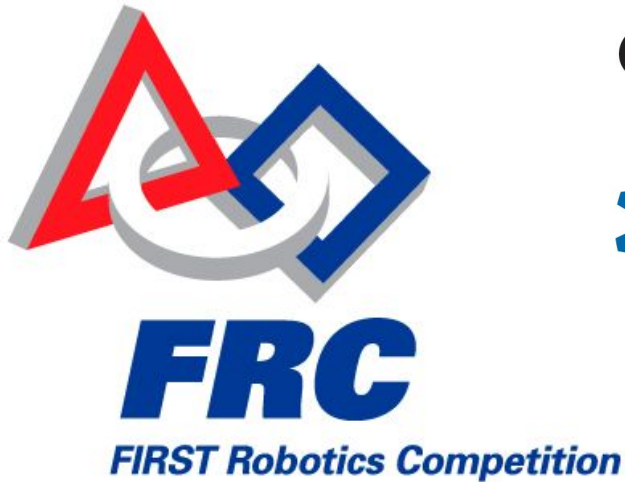


Robotbuilder & Command Based Robot



Team 3244

Robotbuilder & Command Based Robot

About Us

Corey Applegate

- 20 Years Industrial Controls Technician using Rockwell Automation PLC and Motion Control, Kuka Robotics, Fanuc Robotics
- 4 years Mentoring a LEGO Mindstorms Club for 5-6 graders
- 1 season Mentoring VEX
- 1st full year with FIRST was 2016 Stronghold
- Hobbies and skills include
 - 3d Printing, Autodesk Inventor, Raspberry Pi, Arduino

Austin Applegate

- Junior Tech High School
- 1st Year FIRST Stronghold
- Team responsibilities
 - Driver, Build, Some Software
- Hobbies
 - PC builds and Gaming, Football

Team 3244

Robotbuilder & Command Based Robot

What we hope to cover

- Quickly create the backbone of the robot code Using RobotBuilder
- Import project into Eclipse
- Explore the parts of the Project
- Test robot IO
 - without writing a single line of code
- Create commands
 - link Joystick inputs to Subsystems
- Create Methods in Subsystems
 - for the Commands to use
- Test/Tune a PID Subsystem
 - again without writing a single line of code
- Link the setpoints of this PID subsystem to buttons
- Add User code to control the Drivetrain

Team 3244

Robotbuilder & Command Based Robot

Pros and Cons to Command Based programming

Pros

- Can **develop** and **debug** subsystems individually
- Easily create a more **complex autonomous** using command groups
- Well **organized code** (Starting with Robotbuilder)
- Easily **split functions** or subsystem coding responsibilities between multiple students
- **Reuse commands** in both Auto and Teleop

Cons

- **Looks complicated** because there are so many files to create for operator IO, Robot IO, Subsystems, Commands
- Must treat some hardware differently such as limit switches and avoid loops and waits
- Must link and create objects in correct order to avoid errors

Team 3244

Robotbuilder & Command Based Robot

Parts of a Command Based Robot Program SUBSYSTEMS

- **Different subsets** of the robot.
IE Drivetrain, manipulators, vision



<http://wpilib.screenstepslive.com/s/4485/m/13810/l/241892-what-is-command-based-programming>

Team 3244

Robotbuilder & Command Based Robot

Parts of a Command Based Robot Program COMMANDS

- What do you want your subsystem to do?



<http://wpilib.screenstepslive.com/s/4485/m/13810/l/241892-what-is-command-based-programming>

Team 3244

Robotbuilder & Command Based Robot

Parts of a Command Based Robot Program

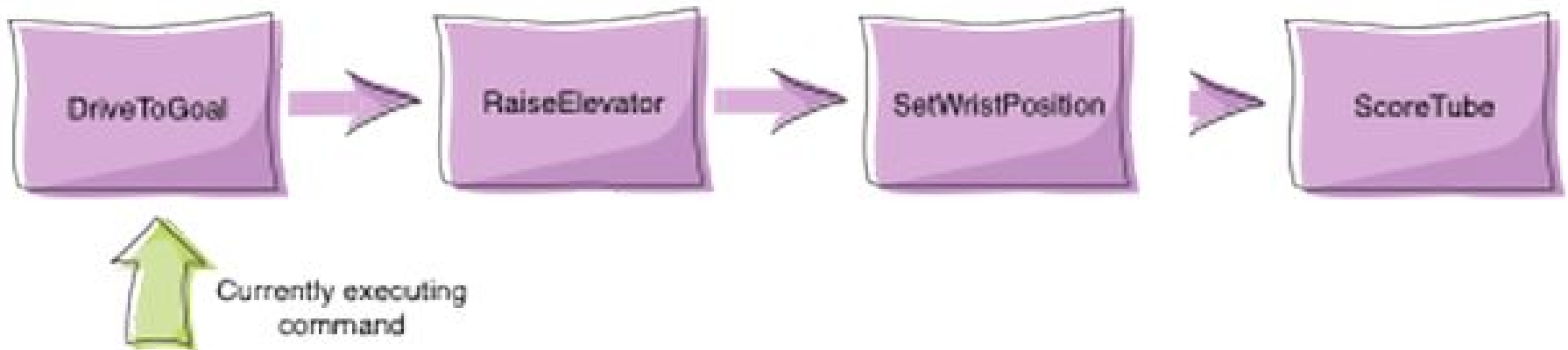
- **Robot**
 - Main program
- **OI**
 - All Joystick controls created and assigned functionality
 - Smartdashboard Buttons
- **RobotMap**
 - Maps all Robot Inputs and Outputs Plugged into the RoboRio

Team 3244

Robotbuilder & Command Based Robot

How Commands work

- Commands let you break up the tasks of operating the robot into small chunks.
 - Each command has an **execute()** method that does some work and an **isFinished()** method that tells if it is done.



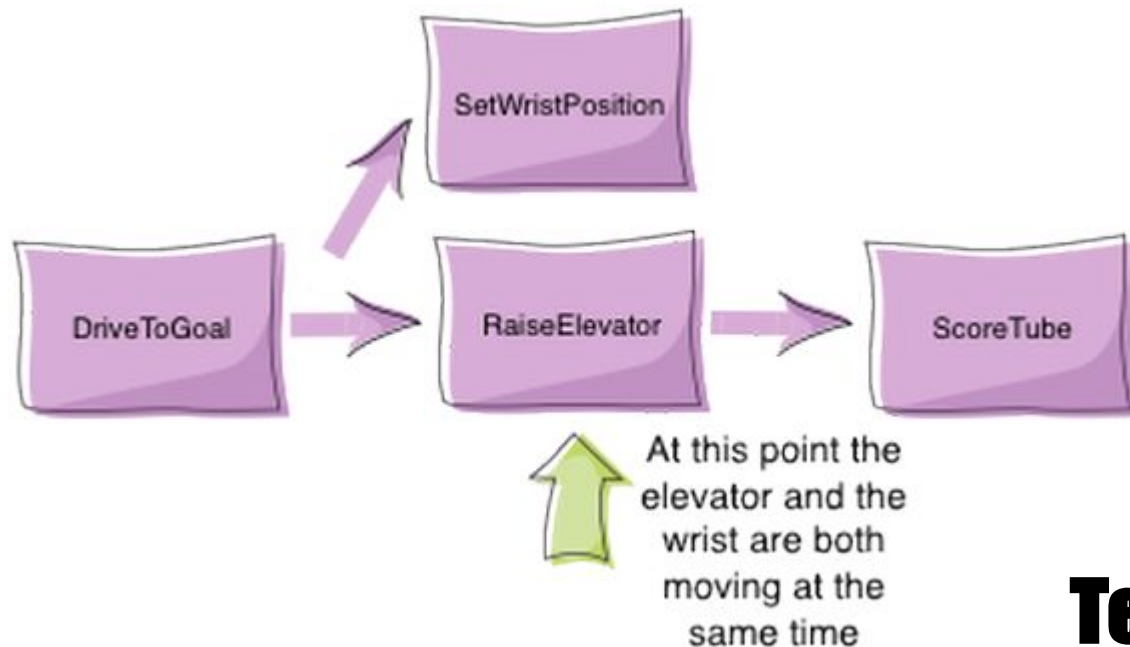
<http://wpilib.screenstepslive.com/s/4485/m/13810/l/241892-what-is-command-based-programming>

Team 3244

Robotbuilder & Command Based Robot

Concurrency

- Sometimes it is desirable to have several operations happening concurrently. In the previous example you might want to set the wrist position while the elevator is moving up. In this case a command group can start a **parallel command** (or command group) running.



Team 3244

Robotbuilder & Command Based Robot

How is Robotbuilder going to help us?

- **Menus and GUI's** to build our subsystems and assign hardware to these systems
- **Error checking** to be sure required components are added to special types of systems or commands
- **Drop menus** populate with only valid options for component
- **Creates** all the associations and **many useful code entries** for all devices attached to the robot.
- **Without a line of code written** we can test components connected to the robot
- **Create a wire map** for the Electricians.

Team 3244

Robotbuilder & Command Based Robot

Start Eclipse and open Robot builder

1. Start [Live Demo](#)
2. There are a few set up steps before robot Programing

Team 3244

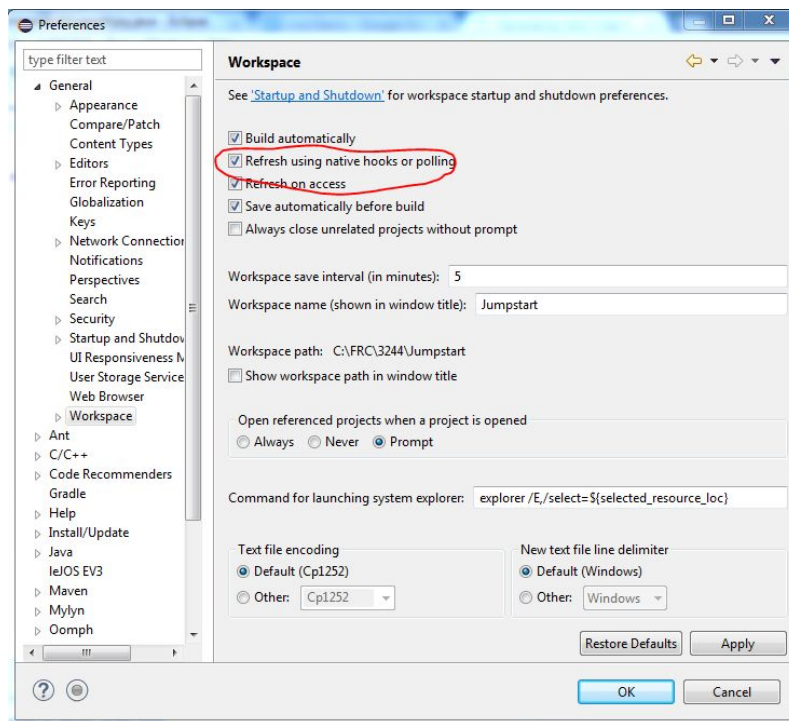
Robotbuilder & Command Based Robot

Live Demo

Start Eclipse

1. After installing Eclipse per the Wpilib.Screenstepslive instructions a new setting must be set.
 - a. Windows>Preferences>General>Workspace

This setting will allow Eclipse to refresh automatically each time you return to Robotbuilder to add or change settings after the initial project import.

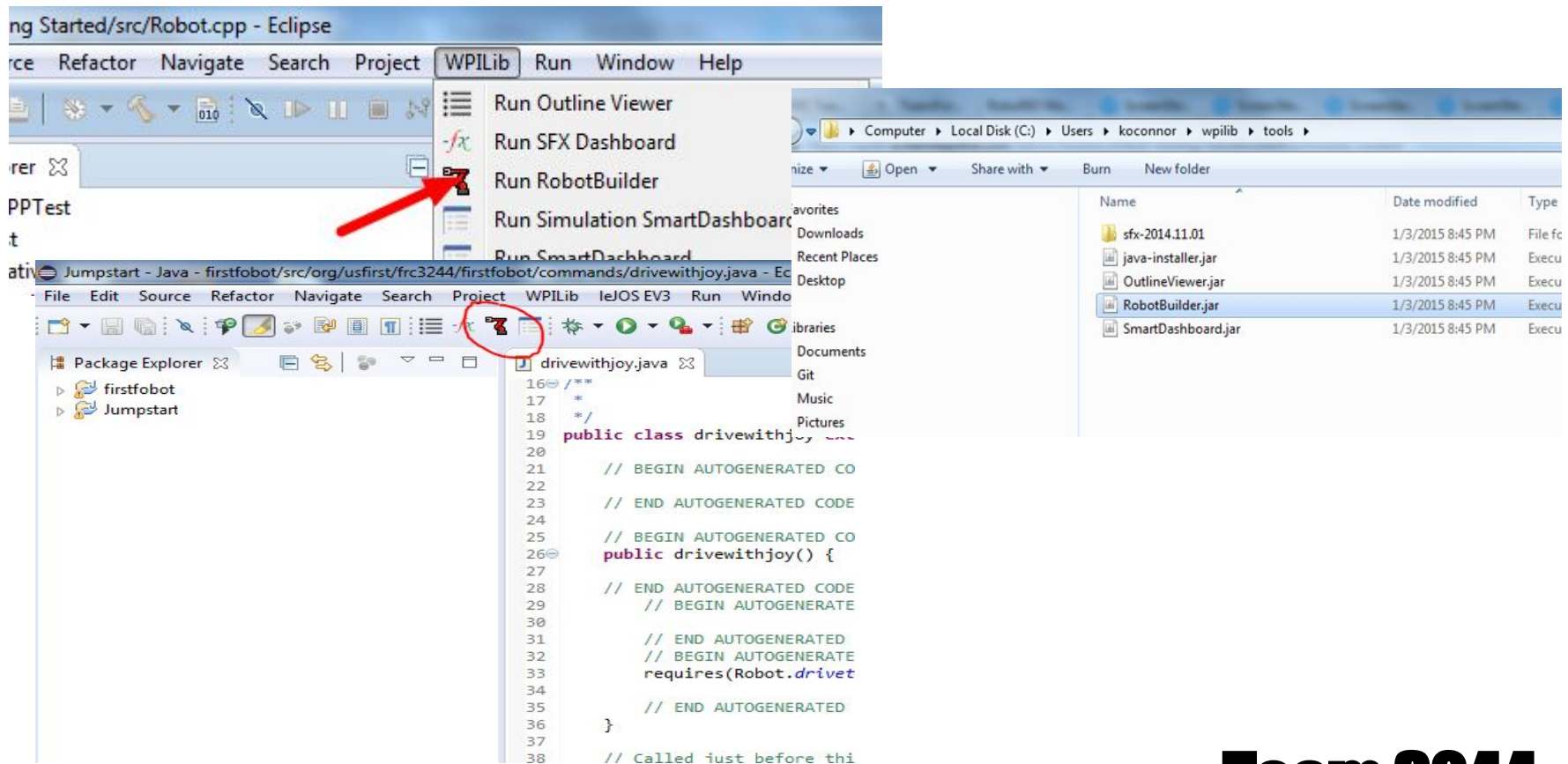


Team 3244

Robotbuilder & Command Based Robot

Live Demo

Start Robotbuilder Tool

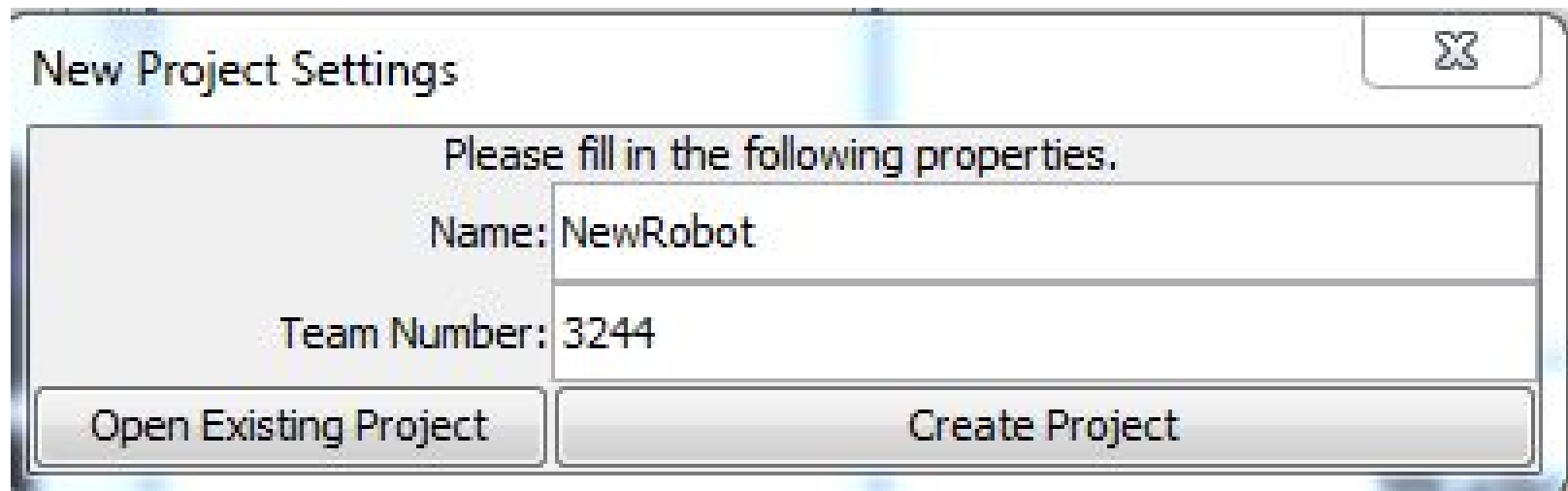


Team 3244

Robotbuilder & Command Based Robot

Live Demo

Start a New Project



The screenshot shows a 'New Project Settings' dialog box. It has a title bar with a close button. Inside, there is a message 'Please fill in the following properties.' followed by two input fields. The first field is labeled 'Name:' and contains the text 'NewRobot'. The second field is labeled 'Team Number:' and contains the text '3244'. At the bottom, there are two buttons: 'Open Existing Project' and 'Create Project'.

New Project Settings	
Please fill in the following properties.	
Name:	NewRobot
Team Number:	3244
Open Existing Project	Create Project

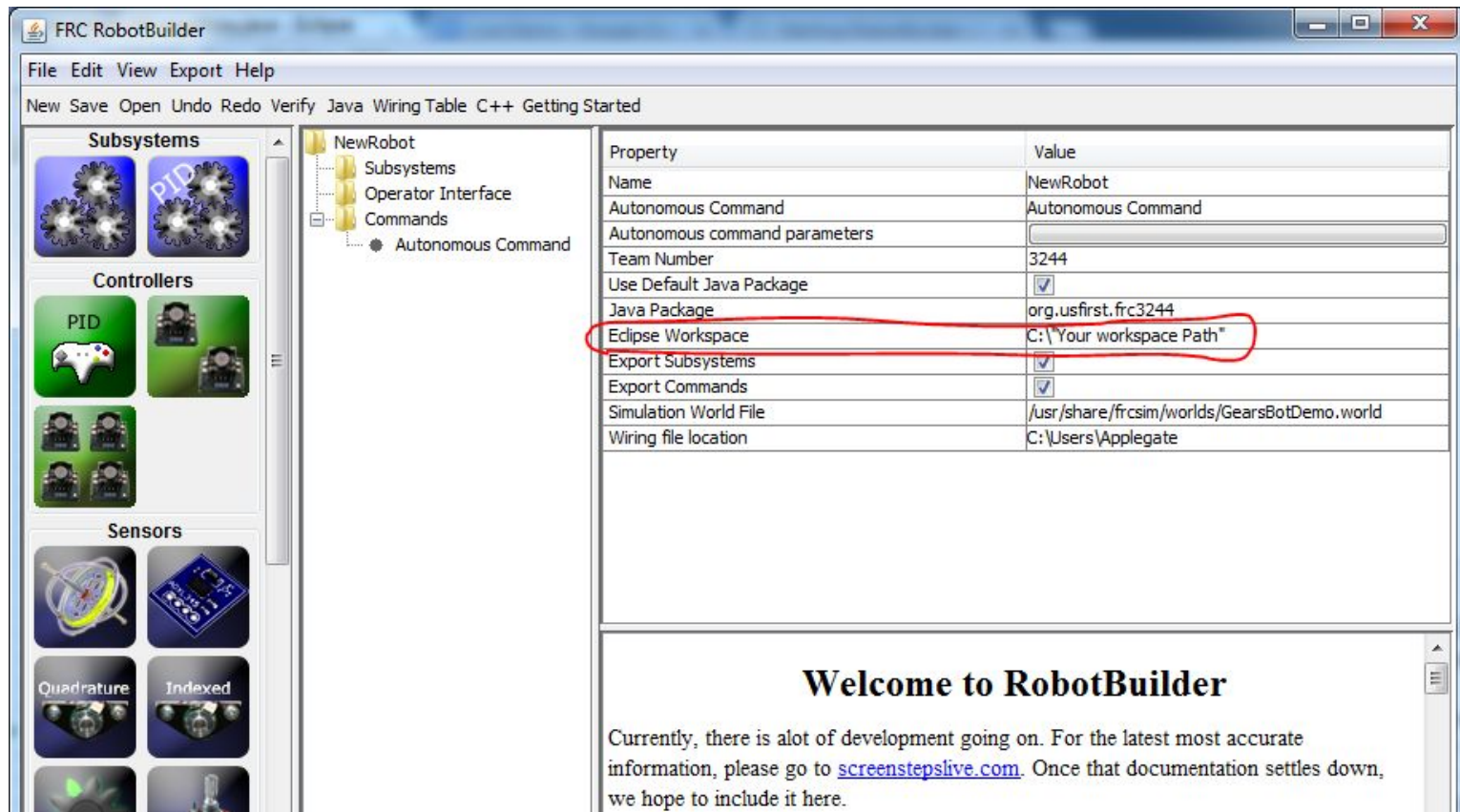
1. Fill in Name and Team Number

Team 3244

Robotbuilder & Command Based Robot

Live Demo

Complete the Eclipse Workspace location



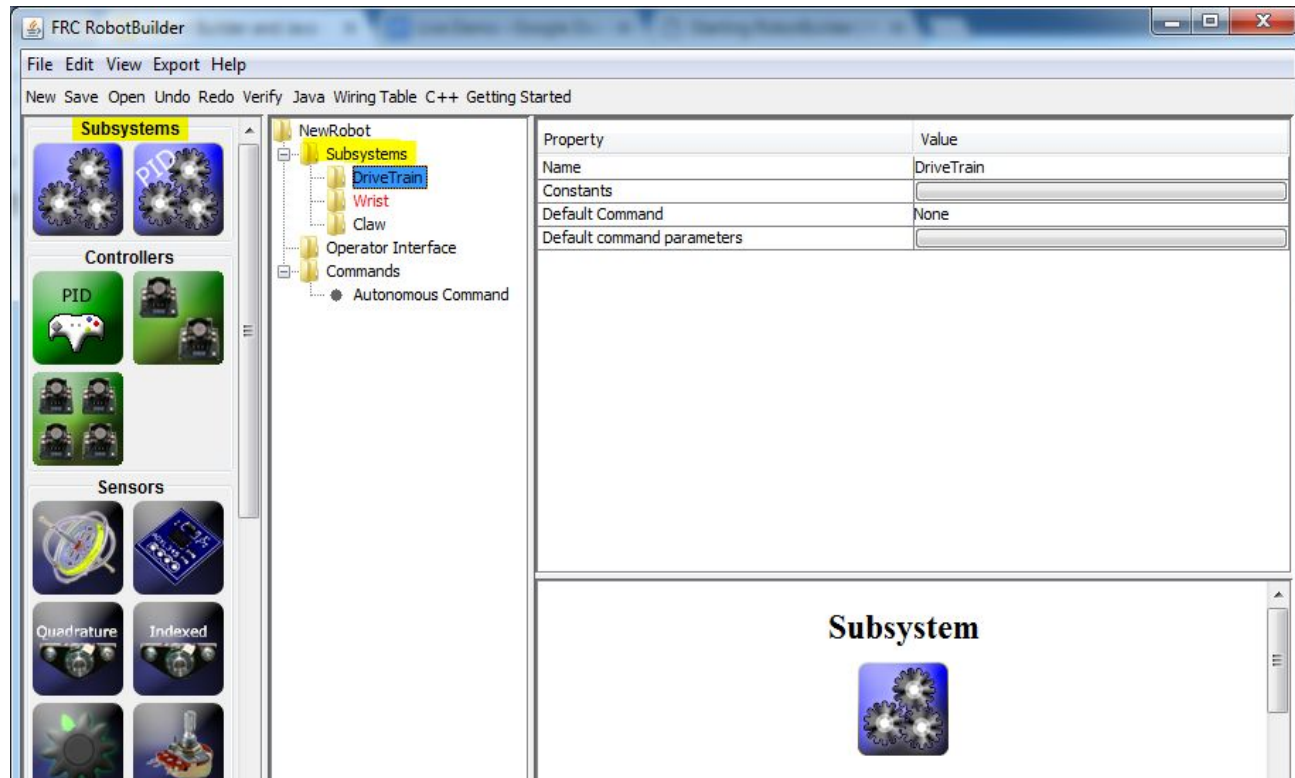
Team 3244

Robotbuilder & Command Based Robot

Live Demo

Divide Robot into Subsystems

1. Subsystem 1
 - a. Standard Type
“Drivetrain”
2. Subsystem 2
 - a. PID Type
“Wrist”
3.



Team 3244

Robotbuilder & Command Based Robot

Live Demo

Add Actuators and Sensors

1. Add Actuators to Subsystems by either dragging into the workspace from the sidebar or right click menu
 - a. Speed Controller select Type and PWM Port
 - b. CAN Speed Controllers Have their own icon.
2. Add Sensors the same way
 - a. Set DIO/Analog Channels.
 - b. Set other special setting per device type.

The screenshot displays the Robotbuilder software interface. On the left, the 'Subsystems' sidebar contains icons for 'PID', 'Controllers', and 'Sensors'. The 'Controllers' section shows a 'PID' controller icon. The 'Sensors' section shows various sensor icons. The main workspace shows a tree view of the robot structure, including 'NewRobot', 'Subsystems', 'DriveTrain', 'Robot Drive 2 1', 'Wrist', 'Claw', 'Operator Interface', 'Commands', and 'Autonomous Command'. The 'Speed Controller' configuration panel is open, showing a 'Motor Right' icon and a 'Pot' icon. The panel includes a 'Property' table with columns 'Property' and 'Value'.

Property	Value
Name	Motor Right
Type	Victor
Output Channel (PWM)	2

Speed Controller

What is it?
A speed controller that uses PWM signals to control the speed of the attached motor. There are seven main PWM speed controllers available for use in FRC: Victor, Talon, Jaguar, Victor SP, Talon SRX, Spark, and SD540.

Properties

Team 3244

Robotbuilder & Command Based Robot

Live Demo

Correct Red Highlighted Errors

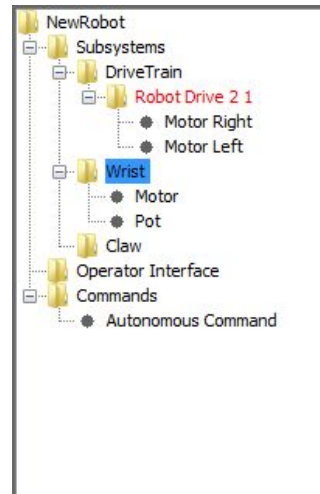
1. Robot Drive will always Prefill with the first Speed Controller. You will need to Manually select in drop downs



The screenshot shows the RobotBuilder interface with the DriveTrain subsystem selected. The 'Robot Drive 2 1' property is highlighted in red. The error message states: 'Error: Hover over the red property names for details of how to fix.'

Property	Value
Name	Robot Drive 2 1
Left Motor	DriveTrain Motor Right
Left Motor Inverted	<input type="checkbox"/>
Right Motor	DriveTrain Motor Right
Right Motor Inverted	<input type="checkbox"/>
Sensitivity	0.5
Maximum Output	1.0
Safety Enabled	<input checked="" type="checkbox"/>
Safety Expiration Time	0.1

2. PID Type Subsystems require a Speed Controller and an input. Analog or Encoder



The screenshot shows the RobotBuilder interface with the Wrist subsystem selected. The 'Wrist' property is highlighted in red. The error message states: 'Error: Hover over the red property names for details of how to fix.'

Property	Value
Name	Wrist
Constants	
Default Command	None
Default command parameters	
Input	Wrist Pot
Output	Wrist Motor
P	1.0
I	0.0
D	0.0
F	0.0
Tolerance	0.2
Continuous	<input type="checkbox"/>
Limit Input	<input type="checkbox"/>
Minimum Input	0.0
Maximum Input	5.0
Limit Output	<input type="checkbox"/>
Minimum Output	-1.0
Maximum Output	1.0

Team 3244

Robotbuilder & Command Based Robot

Live Demo

Add Operator Interface

1. Drag or right click Operator Interface and add Joystick
 - a. Name Joystick
 - b. Set Device ID

The screenshot displays the RobotBuilder software interface. On the left, a sidebar contains categories: Subsystems, Controllers, and Sensors. The 'Operator Interface' component is highlighted in the 'Controllers' section. The main workspace shows a hierarchical tree of the robot's structure, including 'NewRobot', 'Subsystems', 'DriveTrain', 'Robot Drive 2 1', 'Wrist', 'Claw', and 'Commands'. The 'Operator Interface' is being added to the 'Commands' section. On the right, a 'Property' table is visible, showing the 'Name' property set to 'Operator Interface'.

Property	Value
Name	Operator Interface

Team 3244

Robotbuilder & Command Based Robot

Live Demo

Create a Command

1. Right Click Commands or drag in from tools a simple command.
 - a. Name Command.
Use a good
Descriptive title to
identify its intentions.
*I like to start the name with the sub system
 - b. Set the Requires to
the Subsystem this
command is going to
use.
 - c. Uncheck Button on
SmartDashboard if
you DO NOT wish to
see this command on
driverstation

The screenshot displays the RobotBuilder software interface. On the left, a tree view shows the project structure under 'NewRobot', including 'Subsystems' (DriveTrain, Robot Drive 2 1, Wrist, Claw), 'Operator Interface' (Driver Xbox Controller), and 'Commands' (Autonomous Command, Drive With Joysticks). Below the tree is a 'Commands' panel with icons for various command types like 'PID', 'B', 'timed', and 'instant'. The 'Drive With Joysticks' command is selected. On the right, the 'Property' panel shows the configuration for 'Drive With Joysticks': Name is 'Drive With Joysticks', Requires is 'DriveTrain', and the 'Button on SmartDashboard' checkbox is unchecked. Below the property panel is a 'Commands' diagram showing a central cloud labeled 'Commands' with arrows pointing to various command names: 'MoveToScoringPosition', 'StowWrist', 'LaunchMiniBot', 'DriveToGoal', 'ElevatorToHighPosition', 'DeployMiniBot', and 'DriveForwardAndScore'. At the bottom of the property panel, there is a 'What is it?' section with a text area for a description.

Property	Value
Name	Drive With Joysticks
Requires	DriveTrain
Button on SmartDashboard	<input type="checkbox"/>
Parameters	
Parameter presets	

Commands

MoveToScoringPosition

StowWrist

LaunchMiniBot

DriveToGoal

ElevatorToHighPosition

DeployMiniBot

DriveForwardAndScore

What is it?

A paragraph or so describing what it is

A command is a single action performed by the robot. Commands usually require at least one subsystem which they act with. After being started, a command runs for a

Team 3244

Robotbuilder & Command Based Robot

Live Demo

With Command Created Set Subsystem default Command

1. This will be the command that runs if no other commands is using this subsystem.

The screenshot displays the RobotBuilder software interface. On the left, a tree view shows the robot's structure: NewRobot > Subsystems > DriveTrain. Under DriveTrain, there is a 'Robot Drive 2 1' subsystem containing 'Motor Right' and 'Motor Left', and a 'Wrist' subsystem containing 'Motor' and 'Pot'. Below these are 'Claw', 'Operator Interface', 'Driver Xbox Controller', and 'Commands'. The 'Commands' folder contains 'Autonomous Command' and 'Drive With Joystics'. On the right, a table shows the properties for the selected 'DriveTrain' subsystem.

Property	Value
Name	DriveTrain
Constants	
Default Command	Drive With Joystics
Default command parameters	None
	Autonomous Command
	Drive With Joystics

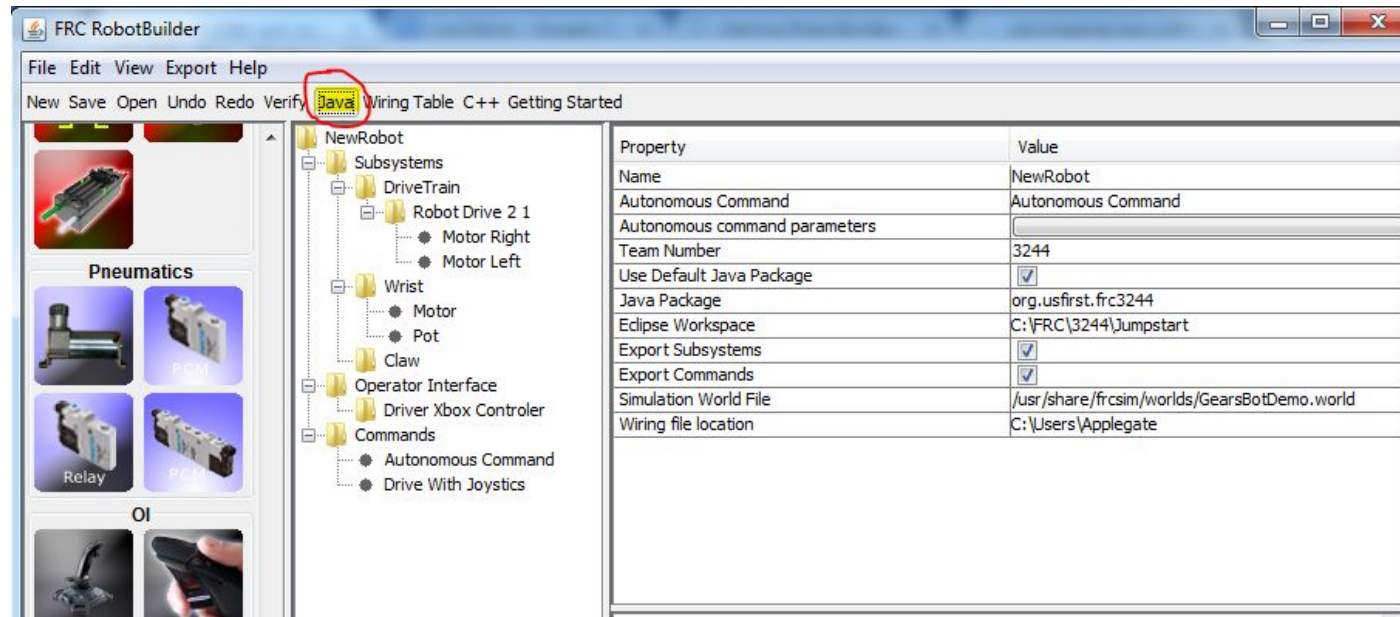
Team 3244

Robotbuilder & Command Based Robot

Live Demo

Save and generate Java Code

1. Save Project using standard windows steps
2. Make sure the Eclipse Workspace is set
3. Click Java Button



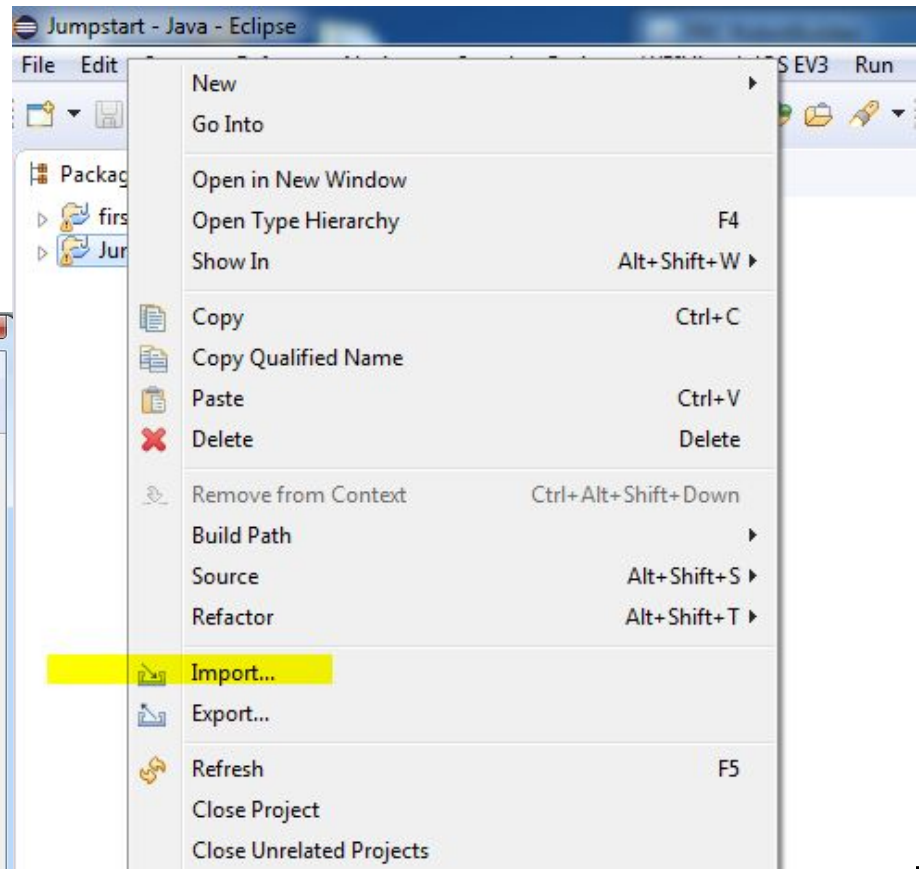
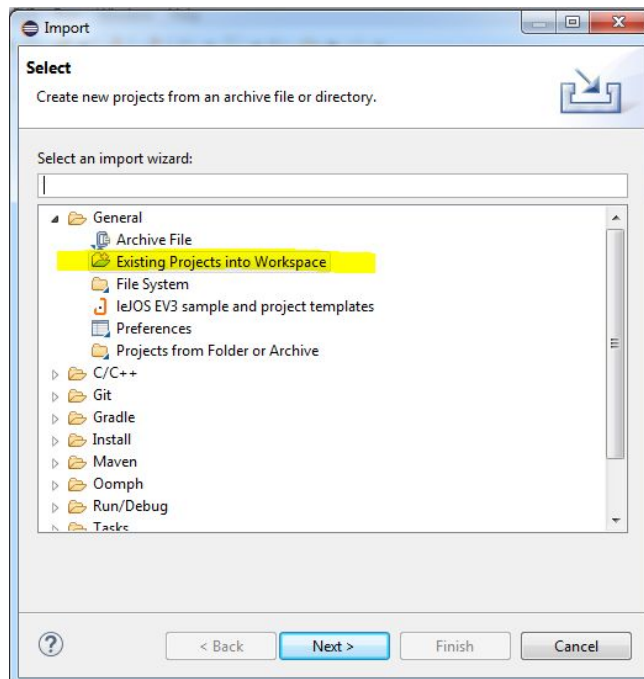
Team 3244

Robotbuilder & Command Based Robot

Live Demo

Import Robotbuilder code into Eclipse

1. Click “File” or right click “Package Explorer” and select Import...
2. Select “Existing Projects into Workspace” then “Next”



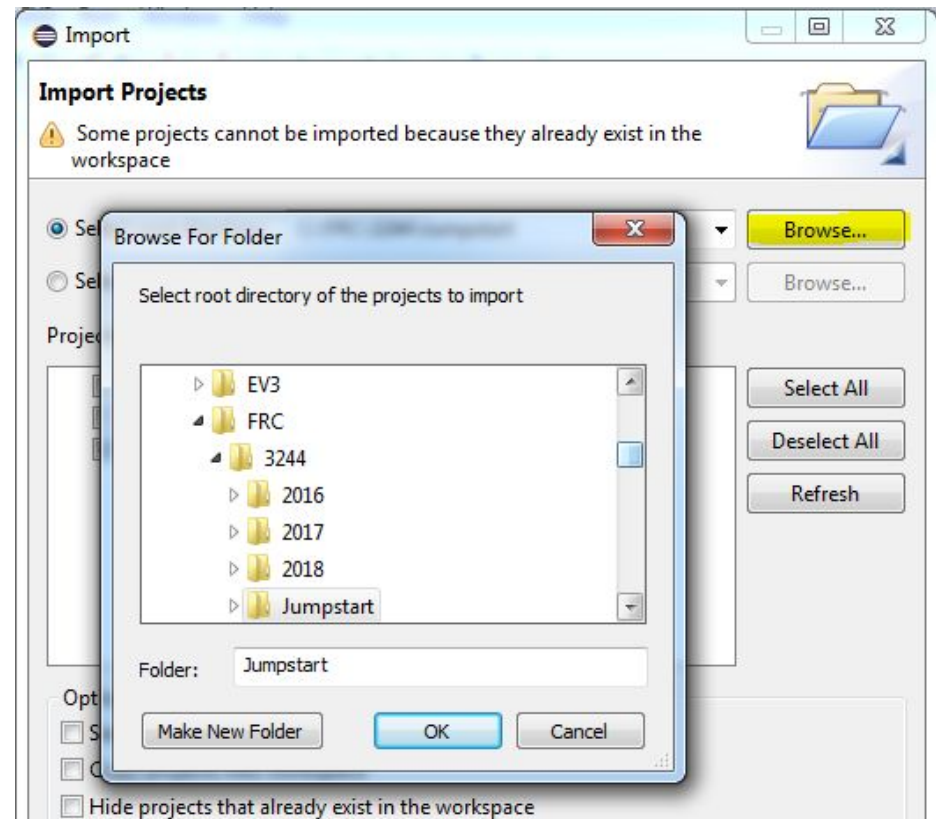
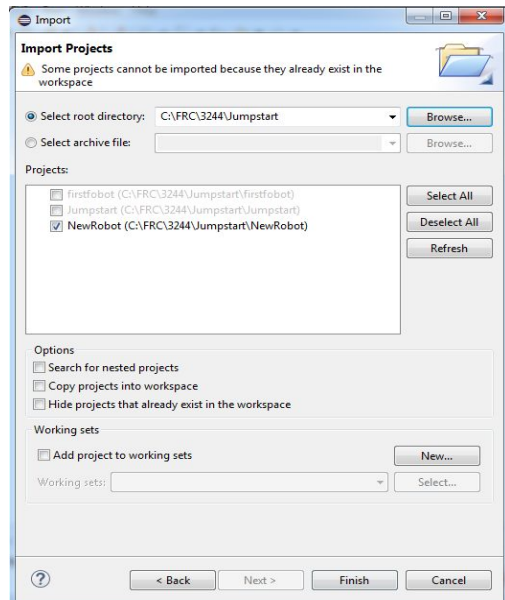
Team 3244

Robotbuilder & Command Based Robot

Live Demo

Import Robotbuilder code into Eclipse

3. Browse for the Eclipse Workspace set in Robotbuilder. This should be the current folder and already selected. Click “OK”
4. Check the Project and click “Finish”



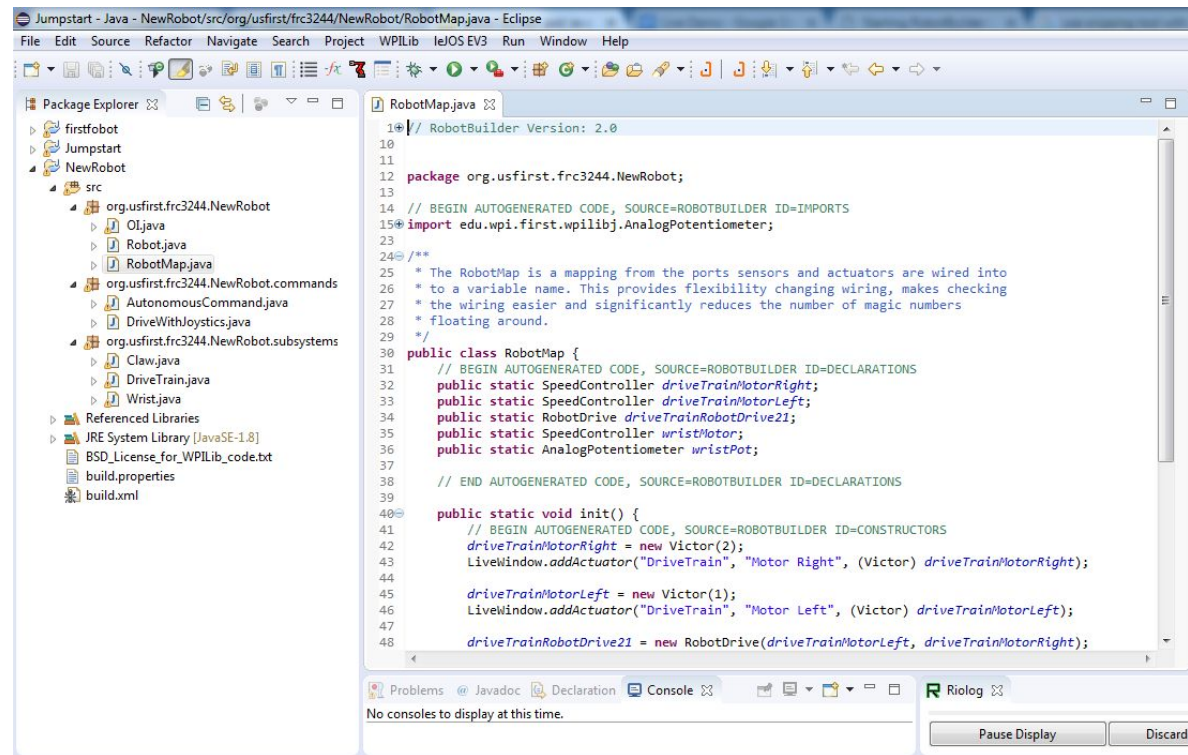
Team 3244

Robotbuilder & Command Based Robot

Live Demo

Imported Code

1. Imported Code nicely organized
 - a. NewRobot
 - b. NewRobot.Commands
 - c. NewRobot.Subsystems
2. In the classes there are comments like
// BEGIN AUTOGENERATED CODE
...
// END AUTOGENERATED CODE
These areas are code generated by Robotbuilder and will be overwritten each time we make changes in Robotbuilder.



```
Jumpstart - Java - NewRobot/src/org/usfirst/frc3244/NewRobot/RobotMap.java - Eclipse
File Edit Source Refactor Navigate Search Project WPIlib leJOS EV3 Run Window Help

Package Explorer
  firstrobot
  Jumpstart
  NewRobot
    src
      org.usfirst.frc3244.NewRobot
        Ol.java
        Robot.java
        RobotMap.java
      org.usfirst.frc3244.NewRobot.commands
        AutonomousCommand.java
        DriveWithJoysticks.java
      org.usfirst.frc3244.NewRobot.subsystems
        Claw.java
        DriveTrain.java
        Wrist.java
  Referenced Libraries
    JRE System Library [JavaSE-1.8]
    BSD_License_for_WPIlib_code.txt
    build.properties
    build.xml

RobotMap.java
1 // RobotBuilder Version: 2.0
10
11
12 package org.usfirst.frc3244.NewRobot;
13
14 // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=IMPORTS
15 import edu.wpi.first.wpilibj.AnalogPotentiometer;
23
24 /**
25  * The RobotMap is a mapping from the ports sensors and actuators are wired into
26  * to a variable name. This provides flexibility changing wiring, makes checking
27  * the wiring easier and significantly reduces the number of magic numbers
28  * floating around.
29  */
30 public class RobotMap {
31     // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DECLARATIONS
32     public static SpeedController driveTrainMotorRight;
33     public static SpeedController driveTrainMotorLeft;
34     public static RobotDrive driveTrainRobotDrive21;
35     public static SpeedController wristMotor;
36     public static AnalogPotentiometer wristPot;
37
38     // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DECLARATIONS
39
40     public static void init() {
41         // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=CONSTRUCTORS
42         driveTrainMotorRight = new Victor(2);
43         LiveWindow.addActuator("DriveTrain", "Motor Right", (Victor) driveTrainMotorRight);
44
45         driveTrainMotorLeft = new Victor(1);
46         LiveWindow.addActuator("DriveTrain", "Motor Left", (Victor) driveTrainMotorLeft);
47
48         driveTrainRobotDrive21 = new RobotDrive(driveTrainMotorLeft, driveTrainMotorRight);
49     }
50 }
```

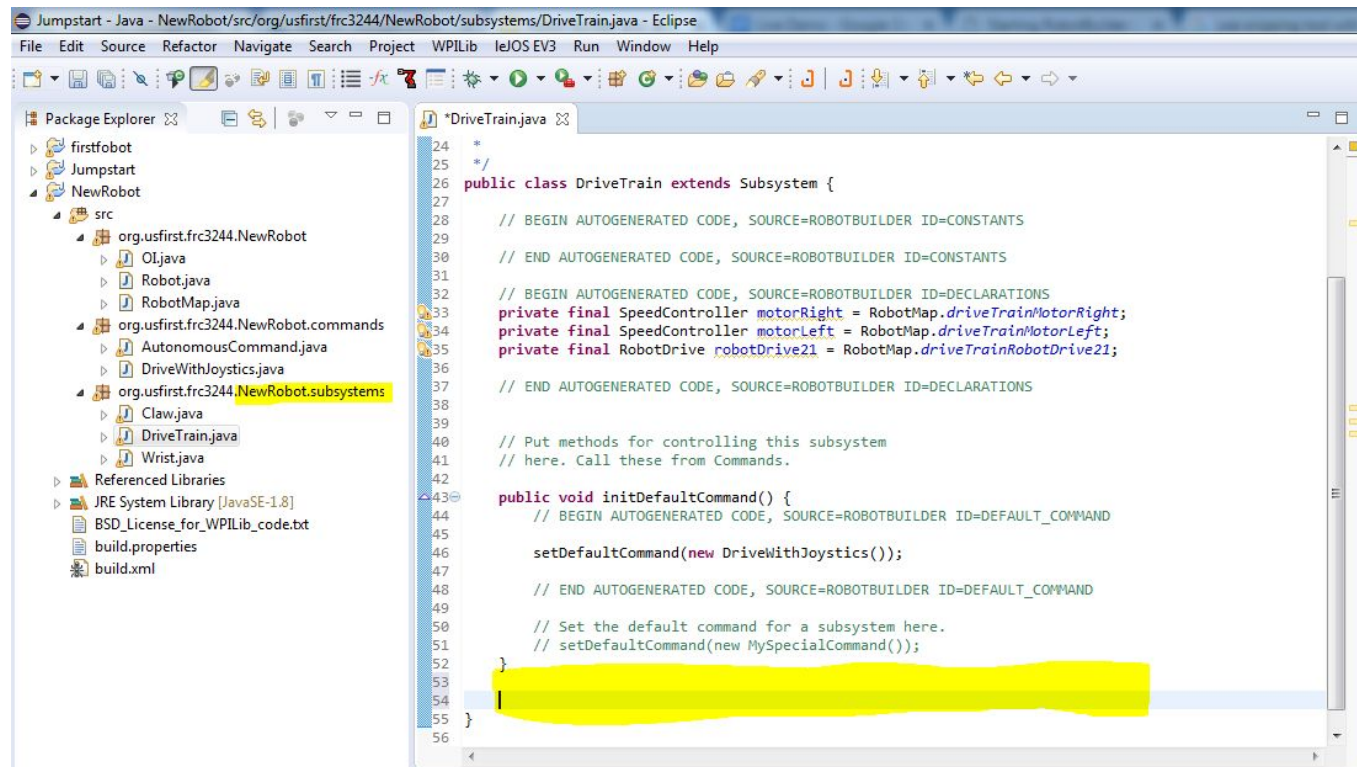
Team 3244

Robotbuilder & Command Based Robot

Live Demo

Subsystems

1. At the top are the objects available for this subsystem to control.
2. A Method created for us to set the default command
3. Highlighted area will be where we write our methods this subsystem can do, get, and set methods for Data.



```
Jumpstart - Java - NewRobot/src/org/usfirst/frc3244/NewRobot/subsystems/DriveTrain.java - Eclipse
File Edit Source Refactor Navigate Search Project WPILib IeIOS EV3 Run Window Help

Package Explorer
  firstrobot
  Jumpstart
  NewRobot
    src
      org.usfirst.frc3244.NewRobot
        Ol.java
        Robot.java
        RobotMap.java
      org.usfirst.frc3244.NewRobot.commands
        AutonomousCommand.java
        DriveWithJoystics.java
      org.usfirst.frc3244.NewRobot.subsystems
        Claw.java
        DriveTrain.java
        Wrist.java
      Referenced Libraries
        JRE System Library [JavaSE-1.8]
        BSD_License_for_WPILib_code.txt
        build.properties
        build.xml

*DriveTrain.java
24  *
25  */
26  public class DriveTrain extends Subsystem {
27
28      // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=CONSTANTS
29
30      // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=CONSTANTS
31
32      // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DECLARATIONS
33      private final SpeedController motorRight = RobotMap.driveTrainMotorRight;
34      private final SpeedController motorLeft = RobotMap.driveTrainMotorLeft;
35      private final RobotDrive robotDrive21 = RobotMap.driveTrainRobotDrive21;
36
37      // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DECLARATIONS
38
39
40      // Put methods for controlling this subsystem
41      // here. Call these from Commands.
42
43      public void initDefaultCommand() {
44          // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DEFAULT_COMMAND
45
46          setDefaultCommand(new DriveWithJoystics());
47
48          // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DEFAULT_COMMAND
49
50          // Set the default command for a subsystem here.
51          // setDefaultCommand(new MySpecialCommand());
52      }
53
54
55  }
```

Team 3244

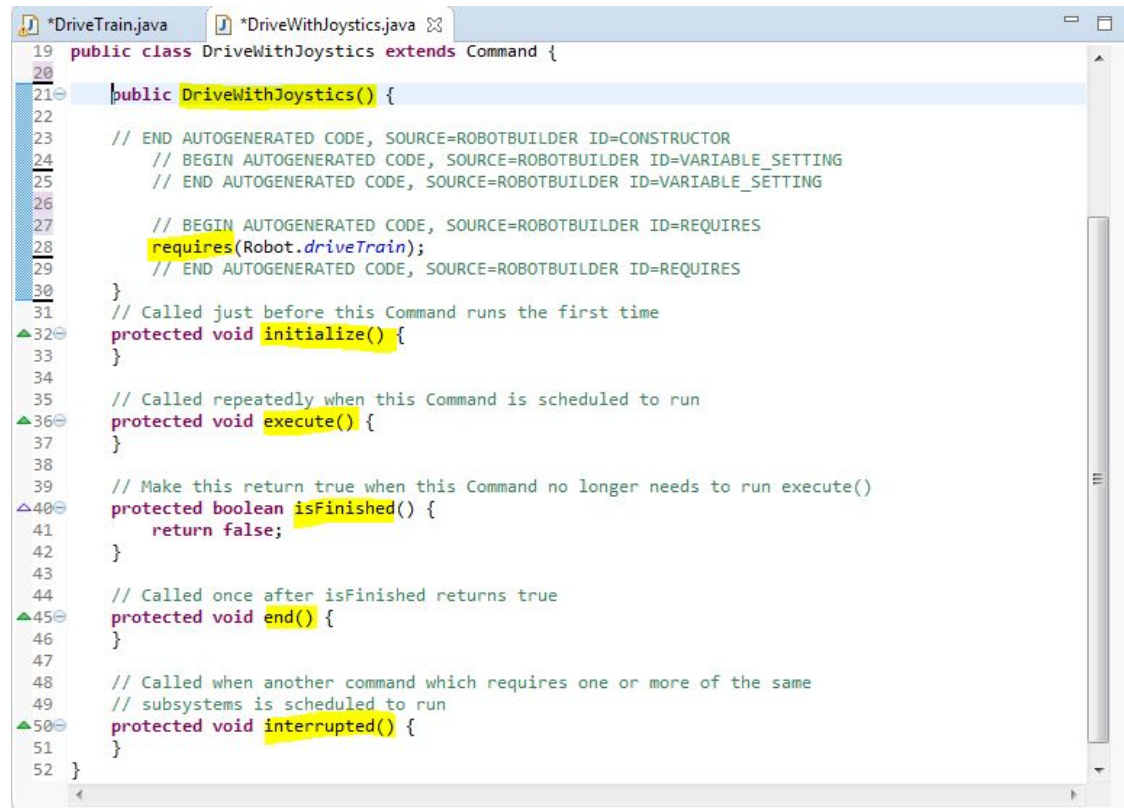
Robotbuilder & Command Based Robot

Live Demo

Commands

This will tell our Subsystems what to do.

1. **Command Constructor**
Sets the Requires
2. **initialize()** Called just before this Command runs the first time
3. **execute()** Called repeatedly when this Command is scheduled to run
 - a. Most of our code will go here
4. **isFinished()** Make this return true when this Command no longer needs to run execute()
5. **end()** Called once after isFinished returns true
6. **interrupted()** Called when another command which requires one or more of the same subsystems is scheduled to run



```
19 public class DriveWithJoystics extends Command {
20
21     public DriveWithJoystics() {
22
23         // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=CONSTRUCTOR
24         // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=VARIABLE_SETTING
25         // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=VARIABLE_SETTING
26
27         // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=REQUIRES
28         requires(Robot.driveTrain);
29         // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=REQUIRES
30     }
31     // Called just before this Command runs the first time
32     protected void initialize() {
33     }
34
35     // Called repeatedly when this Command is scheduled to run
36     protected void execute() {
37     }
38
39     // Make this return true when this Command no longer needs to run execute()
40     protected boolean isFinished() {
41         return false;
42     }
43
44     // Called once after isFinished returns true
45     protected void end() {
46     }
47
48     // Called when another command which requires one or more of the same
49     // subsystems is scheduled to run
50     protected void interrupted() {
51     }
52 }
```

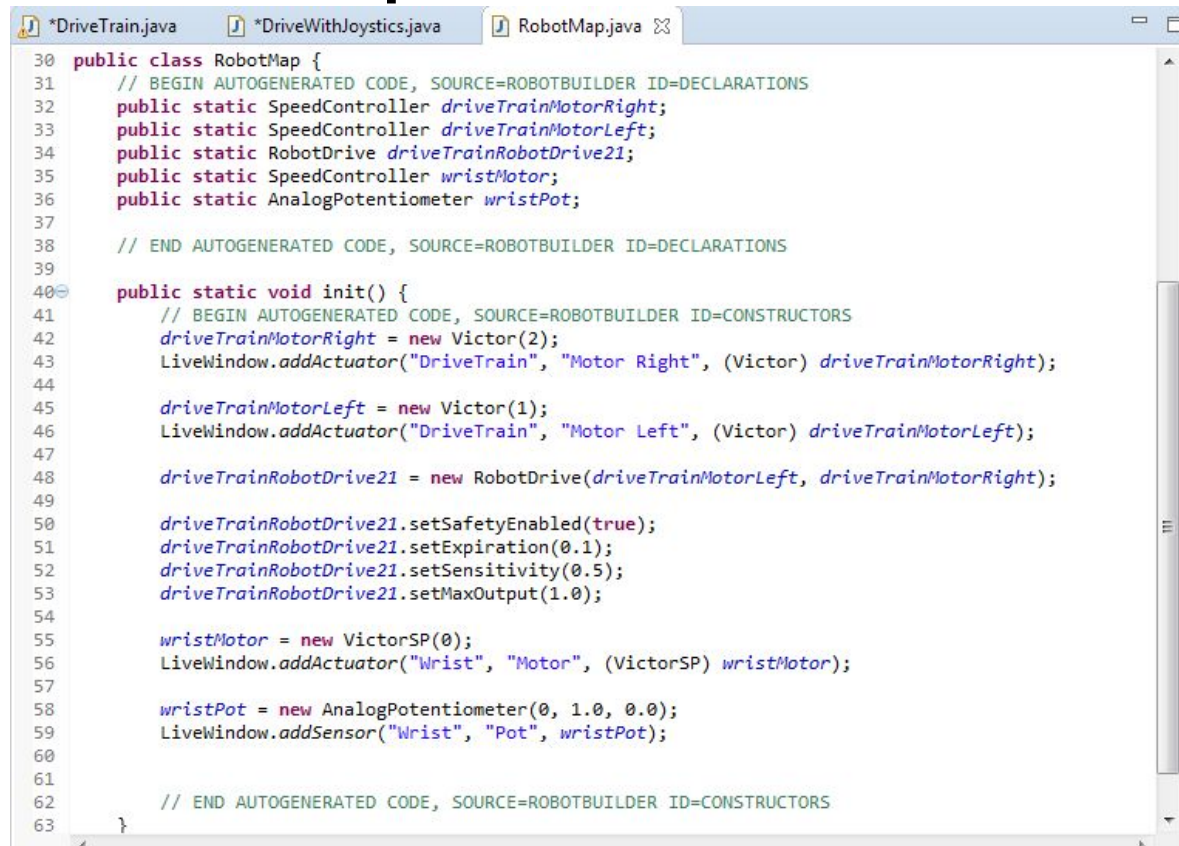
Team 3244

Robotbuilder & Command Based Robot

Live Demo

1. Container that maps the port numbers all PWM, DIO, Relay, Analog, and other components connected to the robot.

Robot Map



```
30 public class RobotMap {
31     // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DECLARATIONS
32     public static SpeedController driveTrainMotorRight;
33     public static SpeedController driveTrainMotorLeft;
34     public static RobotDrive driveTrainRobotDrive21;
35     public static SpeedController wristMotor;
36     public static AnalogPotentiometer wristPot;
37
38     // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DECLARATIONS
39
40     public static void init() {
41         // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=CONSTRUCTORS
42         driveTrainMotorRight = new Victor(2);
43         LiveWindow.addActuator("DriveTrain", "Motor Right", (Victor) driveTrainMotorRight);
44
45         driveTrainMotorLeft = new Victor(1);
46         LiveWindow.addActuator("DriveTrain", "Motor Left", (Victor) driveTrainMotorLeft);
47
48         driveTrainRobotDrive21 = new RobotDrive(driveTrainMotorLeft, driveTrainMotorRight);
49
50         driveTrainRobotDrive21.setSafetyEnabled(true);
51         driveTrainRobotDrive21.setExpiration(0.1);
52         driveTrainRobotDrive21.setSensitivity(0.5);
53         driveTrainRobotDrive21.setMaxOutput(1.0);
54
55         wristMotor = new VictorSP(0);
56         LiveWindow.addActuator("Wrist", "Motor", (VictorSP) wristMotor);
57
58         wristPot = new AnalogPotentiometer(0, 1.0, 0.0);
59         LiveWindow.addSensor("Wrist", "Pot", wristPot);
60
61         // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=CONSTRUCTORS
62     }
63 }
```

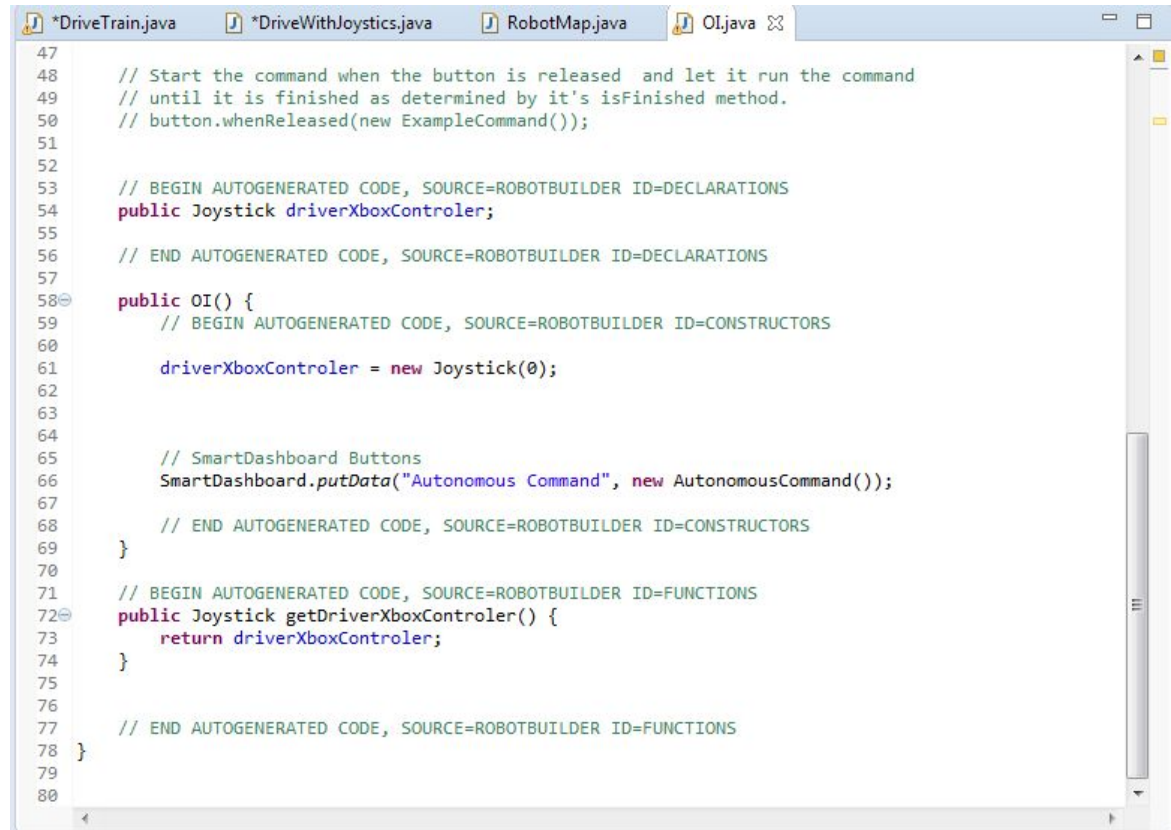
Team 3244

Robotbuilder & Command Based Robot

Live Demo

OI

1. Container that controls all the Operator Interface devices like Joysticks, Buttons, and SmartDashboard Buttons.



```
47
48 // Start the command when the button is released and let it run the command
49 // until it is finished as determined by it's isFinished method.
50 // button.whenReleased(new ExampleCommand());
51
52
53 // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DECLARATIONS
54 public Joystick driverXboxController;
55
56 // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DECLARATIONS
57
58 public OI() {
59     // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=CONSTRUCTORS
60
61     driverXboxController = new Joystick(0);
62
63
64     // SmartDashboard Buttons
65     SmartDashboard.putData("Autonomous Command", new AutonomousCommand());
66
67     // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=CONSTRUCTORS
68 }
69
70
71 // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=FUNCTIONS
72 public Joystick getDriverXboxController() {
73     return driverXboxController;
74 }
75
76
77 // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=FUNCTIONS
78 }
79
80
```

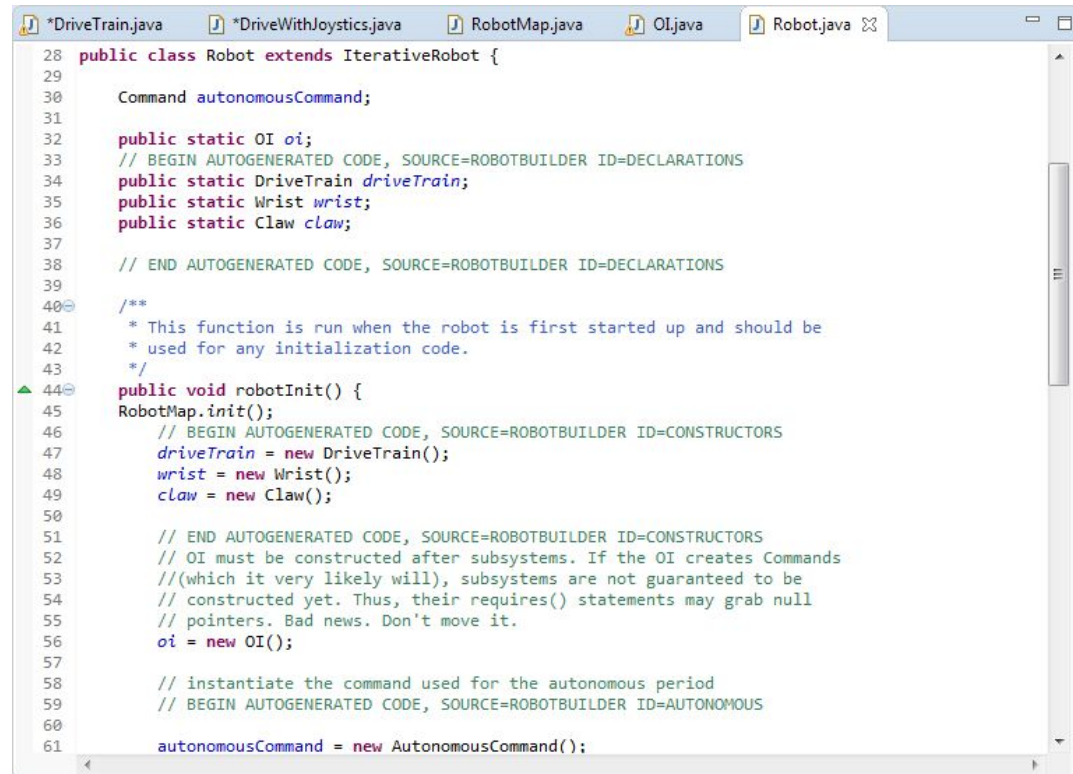
Team 3244

Robotbuilder & Command Based Robot

Live Demo

1. Container with everything related to the robot and the modes the robot will transition through when enabled.
**Init runs once before Periodic runs*
2. `robotInit()` initialization code
3. `disabledInit()`, `disabledPeriodic()` code run when robot disabled
4. `autonomousInit()`, `autonomousPeriodic()` code run when robot is autonomous
5. `teleopInit()`, `teleopPeriodic()` code run when robot is teleop
6. `testPeriodic()` code run when robot in testMode

Robot



```
*DriveTrain.java *DriveWithJoystics.java RobotMap.java OI.java Robot.java
28 public class Robot extends IterativeRobot {
29
30     Command autonomousCommand;
31
32     public static OI oi;
33     // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DECLARATIONS
34     public static DriveTrain driveTrain;
35     public static Wrist wrist;
36     public static Claw claw;
37
38     // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DECLARATIONS
39
40     /**
41      * This function is run when the robot is first started up and should be
42      * used for any initialization code.
43      */
44     public void robotInit() {
45         RobotMap.init();
46         // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=CONSTRUCTORS
47         driveTrain = new DriveTrain();
48         wrist = new Wrist();
49         claw = new Claw();
50
51         // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=CONSTRUCTORS
52         // OI must be constructed after subsystems. If the OI creates Commands
53         //(which it very likely will), subsystems are not guaranteed to be
54         // constructed yet. Thus, their requires() statements may grab null
55         // pointers. Bad news. Don't move it.
56         oi = new OI();
57
58         // instantiate the command used for the autonomous period
59         // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=AUTONOMOUS
60
61         autonomousCommand = new AutonomousCommand();
```

Team 3244

Robotbuilder & Command Based Robot

Live Demo

Download Code to Robot *Test Mode

1. ToDo Test Mode

Team 3244

Robotbuilder & Command Based Robot

Live Demo

Code DriveTrain and Joysticks

Subsystem Code

1. Added Method “myArcadeDrive”
This is the action this subsystem can do.
2. The method accepts two parameters to control the speed and direction of the robot
 - a. moveValue
 - b. rotateValue

```
DriveTrain.java
24  *
25  */
26  public class DriveTrain extends Subsystem {
27
28      // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=CONSTANTS
29
30      // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=CONSTANTS
31
32      // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DECLARATIONS
33      private final SpeedController motorRight = RobotMap.driveTrainMotorRight;
34      private final SpeedController motorLeft = RobotMap.driveTrainMotorLeft;
35      private final RobotDrive robotDrive21 = RobotMap.driveTrainRobotDrive21;
36
37      // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DECLARATIONS
38
39      // Put methods for controlling this subsystem
40      // here. Call these from Commands.
41
42
43      public void initDefaultCommand() {
44          // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DEFAULT_COMMAND
45
46          setDefaultCommand(new DriveWithJoysticks());
47
48          // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DEFAULT_COMMAND
49
50          // Set the default command for a subsystem here.
51          // setDefaultCommand(new MySpecialCommand());
52      }
53
54      public void myArcadeDrive(double moveValue, double rotateValue){
55          robotDrive21.arcadeDrive(moveValue, rotateValue);
56      }
57  }
```

Team 3244

Robotbuilder & Command Based Robot

Live Demo

Code DriveTrain and Joysticks

Command Code

1. `execute()` Called repeatedly
 - a. two variables to store Joystick values from the Robot.oi
 - b. Send Joystick values to the Robot.driveTrain.myArcade Drive method
2. `isFinished()` never complete
3. `end()`
 - a. Turn off motors if this command ends
4. `interrupted()`
 - a. Call `end()` if a new command requires driveTrain

```
DriveTrain.java  *DriveWithJoysticks.java
24 // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=VARIABLE_SETTING
25 // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=VARIABLE_SETTING
26
27 // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=REQUIRES
28 requires(Robot.driveTrain);
29 // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=REQUIRES
30 }
31 // Called just before this Command runs the first time
32 protected void initialize() {
33 }
34
35 // Called repeatedly when this Command is scheduled to run
36 protected void execute() {
37     double moveValue = Robot.oi.driverXboxController.getRawAxis(0);
38     double rotateValue = Robot.oi.driverXboxController.getRawAxis(3);
39     Robot.driveTrain.myArcadeDrive(moveValue, rotateValue);
40 }
41
42 // Make this return true when this Command no longer needs to run execute()
43 protected boolean isFinished() {
44     return false;
45 }
46
47 // Called once after isFinished returns true
48 protected void end() {
49     Robot.driveTrain.myArcadeDrive(0, 0);
50 }
51
52 // Called when another command which requires one or more of the same
53 // subsystems is scheduled to run
54 protected void interrupted() {
55     end();
56 }
57 }
```

Team 3244

Robotbuilder & Command Based Robot

Live Demo

Tune PID Using Test Mode

1. ToDo

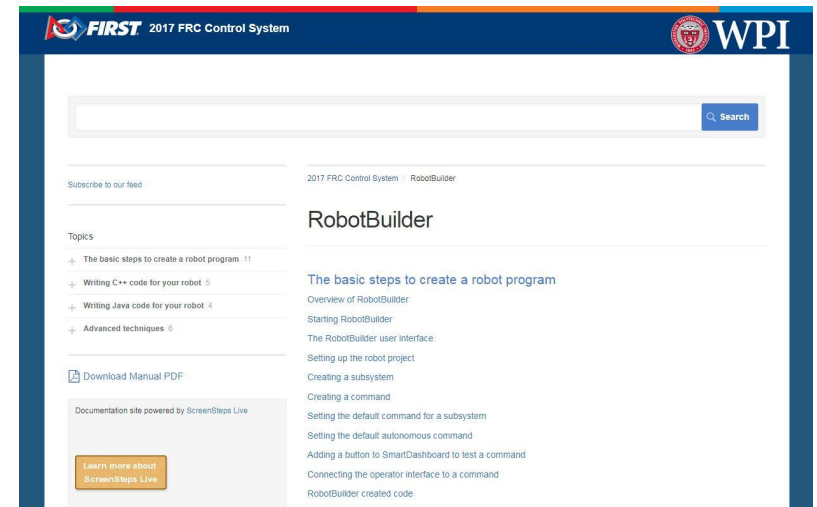
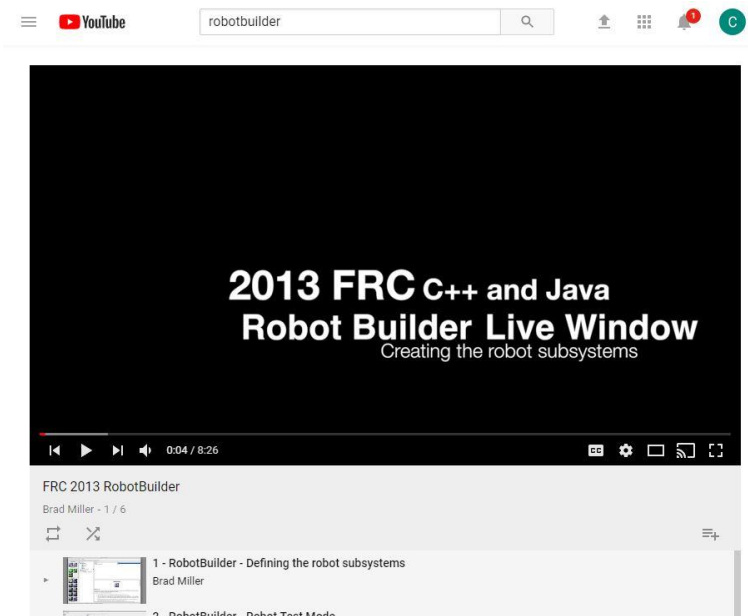
Team 3244

Robotbuilder & Command Based Robot

Find More Information on Robotbuilder

YouTube Brad Miller from WPI created 6 videos in 2013 that are still a great video series.

Search [FRC 2013 Robotbuilder](#)

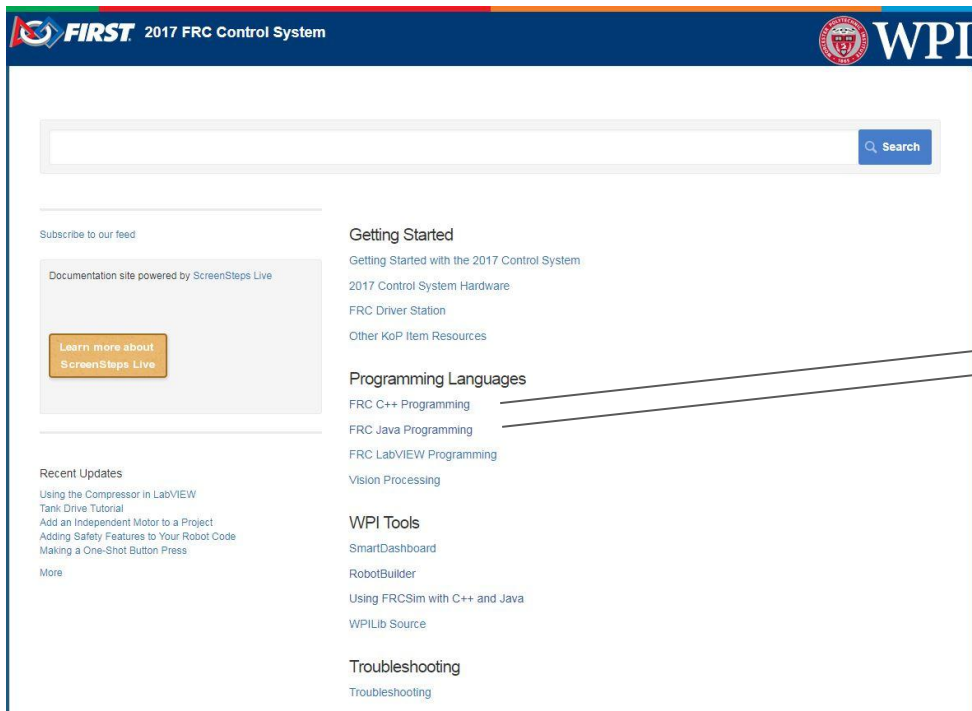


Everyone should be familiar with wpilib.screenstepslive.com

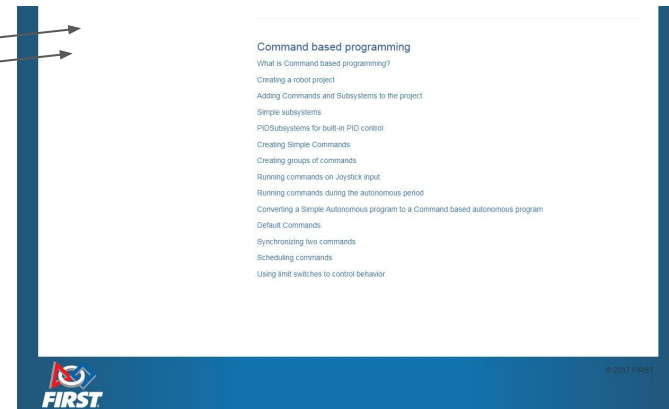
Team 3244

Robotbuilder & Command Based Robot

Find More Information On Command Based Programing



At the bottom of both the C++ and Java Programming Languages will be a Section for Command Based Programming



Team 3244

Questions



Team 3244