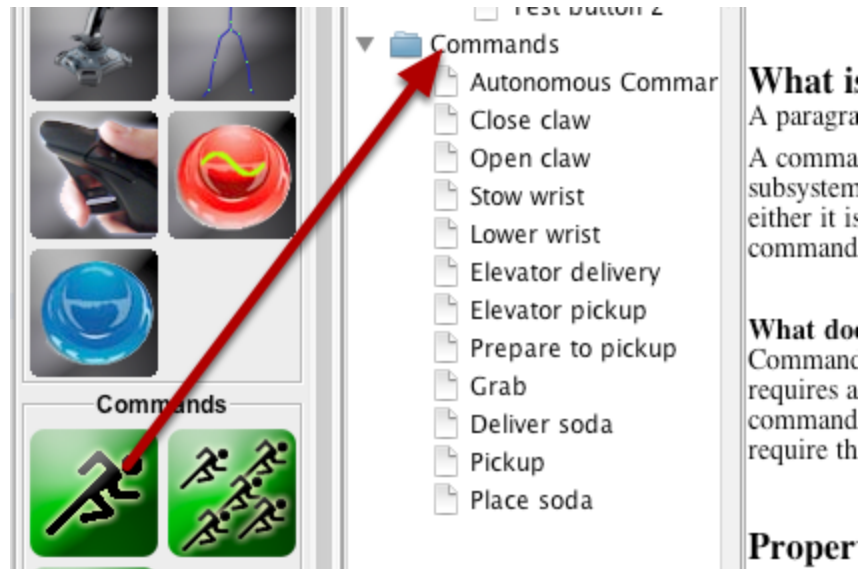


Writing the code for a simple command in Java

Writing the code for a simple command in Java

Subsystem classes get the mechanisms on your robot moving, but to get it to stop at the right time and sequence through more complex operations you write Commands. Previously in [Writing the code for a subsystem in Java](#) we developed the code for the Claw subsystem on a robot to start the claw opening, closing, or to stop moving. Now we will write the code for a command that will actually run the Claw motor for the right time to get the claw to open and close. Our claw example is a very simple mechanism where we run the motor for 1 second to open it or 0.9 seconds to close it.

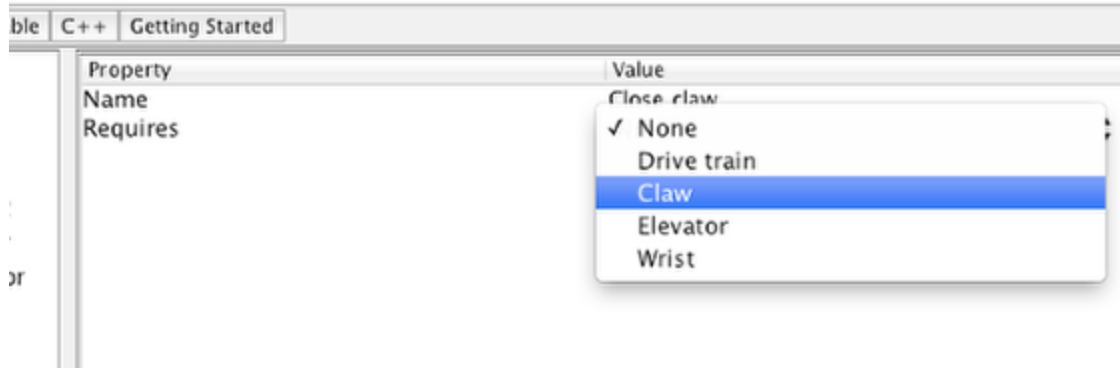
Adding a command



To create a command drag the command icon from the palette to the Commands folder in the robot description. This will create a command with a default name, then rename the command to be something meaningful, in this case "Open claw" or "Close claw".

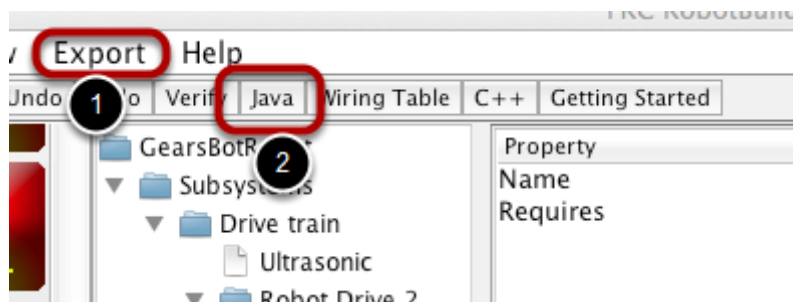
Writing the code for a simple command in Java

Setting the Requires property to the correct subsystem



Set the Requires property to the subsystem that this command is controlling. In this case, the "Close claw" command controls the Claw subsystem. If the "Close claw" command is scheduled while another command that uses the Claw is also running, the "Close claw" command will preempt the other command and start. For example, if the "Open claw" was running, then the robot operator decided that they really wanted to close it, since they both require the Claw, the second command (Close claw) would cancel the running open command.

Generate code for the project



Using either the Export menu (1) or the Java toolbar item (2), generate the code for the project.

Writing the code for a simple command in Java

Write the code to close the claw

```
7
8 public class Closeclaw extends Command {
9
10     public Closeclaw() {
11         // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=RE
12         // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=REQU
13     }
14
15     // Called just before this Command runs the first time
16     protected void initialize() {
17         setTimeout(0.9);
18         Robot.claw.closeClaw();
19     }
20
21     // Called repeatedly when this Command is scheduled to run
22     protected void execute() {
23     }
24
25     // Make this return true when this Command no longer needs
26     // to be executed because it just completed.
27     protected boolean isFinished() {
28         return isTimedOut();
29     }
30
31     // Called once after isFinished returns true
32     protected void end() {
33         Robot.claw.stop();
34     }
35
36     // Called when another command which requires one or more
37     // subsystems is scheduled to run
38     protected void interrupted() {
39         end();
40     }
41 }
```

Add these 5 lines of code to the Closeclaw command to it can be used:

1. The claw needs to run in the close direction for 0.9 seconds to completely get closed. Setting a timeout initializes the timer for this command. Each command can have a single timer that can be used for timing operations or timeouts to make sure that commands don't get stuck in the case of a broken sensor. Then start the claw closing. Notice that we only need to start the claw closing once, so having it in the initialize method is sufficient.
2. The claw should continue closing until the timer runs out. The command has an `isTimedOut()` method that returns true if the timer that was set in the `initialize()` method is done.
3. In the `end()` method stop the claw from moving. This is called when the `isFinished()` method returns true (the timer runs out in this case).
4. The `interrupted()` method is called when this command is preempted by another command using the Claw subsystem. In this case, we just call the `end()` method to stop the claw from moving.

That's all that's required to get the claw to run when the command is run. Notice that you can refer to any subsystem by using the class name `Robot` since subsystem references are automatically

Writing the code for a simple command in Java

generated as static variables. For example, "Robot.claw.closeClaw()" gives you access to the claw class and it's methods.

This command can be part of a more complex Command Group (see: [Creating a command that runs other commands](#)) or run from an operator interface button such as a joystick button (see: [Connecting the operator interface to a command](#)).