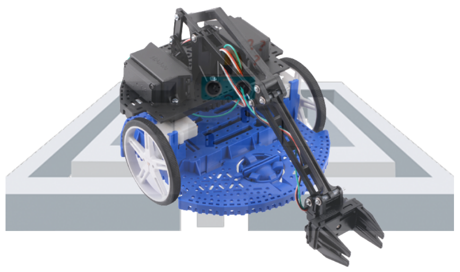
ETECH Romi Workshop #3: Autonomous Game



Objective 1: Run the CatWalk Autonomous code!

- **Task 1A:** Go to Team Discord and copy the link for the new Starter Code
- **Task 1B:** Open VSCode, and clone the shared repository
    - Type **[CTRL] + [SHIFT] + [P]** to open VSCode commands
    - Type `Git clone`
    - Paste the appropriate GitHub repository link
    - Save to a new folder on your computer
- **Task 1C:** Connect your computer to your Romi's WIFI network
    - The default SSID name is "WPILib…." Or something like "WHITE_ROMI"
    - The default password is: **`WPILib2021!`**
    - (You can check that the Romi firmware is up-to-date by going to `10.0.0.2` in your browser)
- **Task 1D:** Deploy your code into your Romi!
    - Click the **[F5]** key
- **Task 1E:** In the simulation window that pops up
    - Click on **"Auto"**
    - Click on **"Run"**
    - What happened???

What is a WPILib Command class?

In the WPILib command-based code, a Command is a Java class which defines one specific action that our robot is capable of doing. Another piece of code in our project -- called the `CommandScheduler` -- will be able to properly run our commands at the right time if we structure our command class the right way.

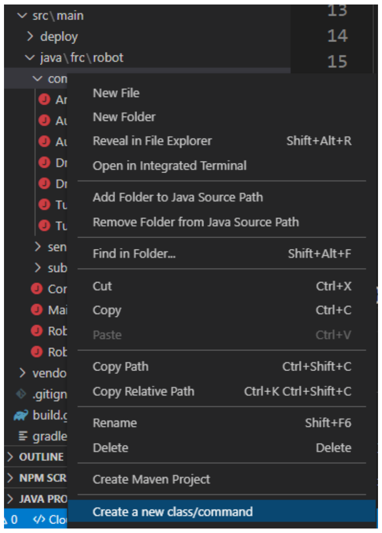
A WPILib Command class will have the following parts inside it:

Part of Command Class	**Description**	*Look in:* `src\ main\ java\ frc\ robot\` **`commands\ ArcadeDrive .java`**
CLASS FIELDS	*All classes in Java contain variables to keep track of important values that will be used in the class.*	*Lines 12-14*
CLASS CONSTRUCTOR	The main job of a constructor method is to pass the parameters to the class fields. All Java classes have constructors, although like with the fields, they aren't always explicitly declared.	*Lines 24-32*
`initialize()` **METHOD**	*Runs only one time - right away when the command is sent to the `CommandScheduler` to be run.*	*Lines 34-36*
`execute()` **METHOD**	*Called constantly while the command is running. For the `ArcadeDrive` command, it's how you drive your robot.*	*Lines 38-42*
`end()` **METHOD**	*Called one time when the command is either interrupted, such as by another command overriding it using the same subsystem, or when the command finishes.*	*Lines 44-46*
`isFinished()` **METHOD**	*Called constantly while the command is running to check WHEN the command should finish.* ● *If the method returns `true`, the `CommandScheduler` will terminate the command and then run the `end()` method.* ● *For the `ArcadeDrive` command, we will always return `false` because we don't ever want to stop driving the robot.* ● *If we wanted to drive forward 24 inches, we would return true only when our sensor has reached 24.*	*Lines 48-52*

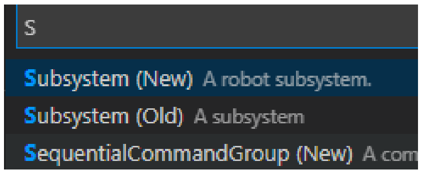
Look through the `DriveDistance.java` command class.

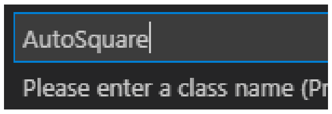
1. What subsystem does the command use?
2. What parameters does the constructor need to properly setup the command?
3. What happens repeatedly throughout the command?
4. What triggers the command to be finished?

Objective 2: Get your robot to drive forward exactly 24 inches!


src\main
deploy
java\frc\robot
New File
New Folder
Reveal in File Explorer Shift+Alt+R
Open in Integrated Terminal
Add Folder to Java Source Path
Remove Folder from Java Source Path
Find in Folder... Shift+Alt+F
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Copy Path Ctrl+Shift+C
Copy Relative Path Ctrl+K Ctrl+Shift+C
Rename Shift+F6
Delete Delete
Create Maven Project
Create a new class/command
OUTLINE
NPM SCR
JAVA PRO

- **Task 2E: Create a new `SequentialCommandGroup` class called `AutoSquare.java`**
  - Right-click on the **`command`** folder
  - Click on **Create a new class/command**
  - Click on **SequentialCommandGroup(NEW)**
  - Name the class **`AutoSquare`**

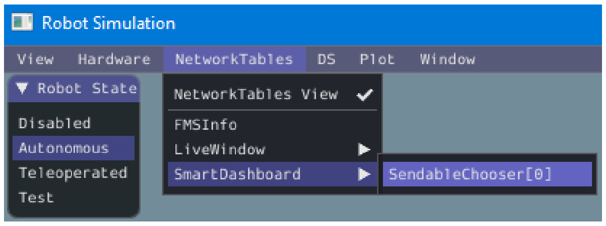

S
Subsystem (New) A robot subsystem.
Subsystem (Old) A subsystem
SequentialCommandGroup (New) A com



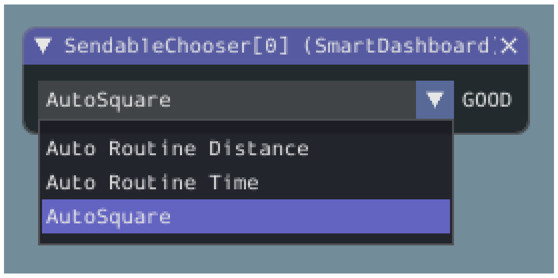
Please enter a class name (Pr

Task	What to Code	Where to Code It	
Task 2F	*Create AutoSquare.java command*	*Look in:* *`src\ main\ java\ frc\ robot\`* ***`commands\`***	
Task 2G	*Add the `Drivetrain` subsystem as a parameter for the `AutoSquare` constructor* ● *(See line 23 of `AutoDistance.java` for inspiration.)*	*Code in:* *`src\ main\ java\ frc\ robot\`* ***`commands\ AutoSquare.java`*** *See for inspiration:* ***`commands\ AutoDistance.java`***	Line 16
Task 2H	*Import the Drivetrain class*		Line 6
Task 2I	*Add a `DriveDistance` command to make your Romi go forward for 24 inches.* ● *(Go look at line 28 of the `DriveDistance.java` class to understand which parameter refers to inches.)*		Line 17
Task 2J	*Import the `AutoSquare` class*	*Look in:* *`src\ main\ java\ frc\ robot\`* ***`RobotContainer.java`***	*Line 13*
Task 2K	*Construct an `AutoSquare` command and add it to be another autonomous option to be run*		*Line 77*

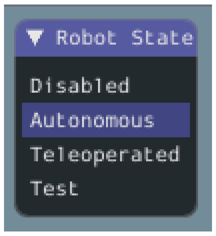
- **Task 2G:** Test your autonomous code!
  - [F5] to build your code and open the Robot Simulation window
  - Click on **NetworkTables > SmartDashboard > SendableChooser**



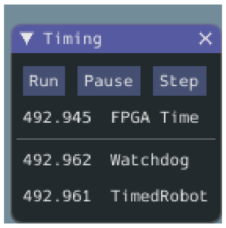
  - Select the **AutoSquare** command in the drop-down menu:



  - Select **Autonomous** Robot State:



  - Click **Run** in the Timing window to start running your code!



```
492.945   FPGA Time
492.962   Watchdog
492.961   TimedRobot
```

  - Use a ruler to measure how far it actually went!

Objective 3: Get your robot to travel in a Square around you!

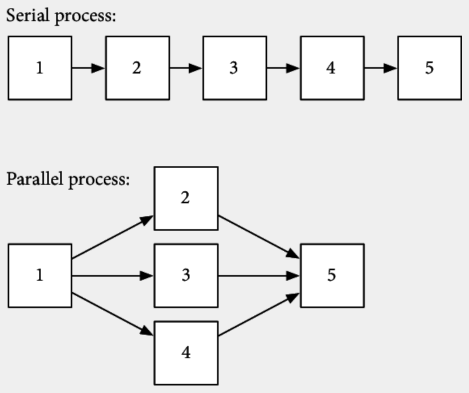
- **Task 3A:** Add a `TurnDegrees` command to your `AutoSquare` command group
    - Include a comma (,) in between the two commands
    - *How do you make it turn exactly 90 degrees?*
    - Check out the `TurnDegrees` command class to see which parameter controls degrees
- **Task 3B**: Test if your code makes the robot go forward and turn 90 degrees
- **Task 3C:** Add more commands into your AutoSquare command group to complete an entire square!
- **Task 3D:** Test!
- **Task 3E:** Try making your robot travel in the shape of an equilateral triangle with 9” sides

What is a CommandGroup?

Command groups are … groups of commands!

In WPILib Java code, you can run commands one after another -- `SequentialCommandGroup`, or you can have multiple commands running at the same time if they are requiring different subsystems -- `ParallelCommandGroup`.

Look at the Command group `AutoDistance.java` to see an example of a SequentialCommandGroup.



What does `Command...` mean in Java ?
In Java, the three periods means that the method can take a variable number of objects of the specified type. It could be one, it could be zero, it could be ten or a hundred.

Objective 4: Get your robot to navigate the Maze!

Objective 5: RubberDuck Challenge

Your Romi robot will have 60 seconds to secure as many Rubber Ducks into the goal area as possible!

- The first 20 seconds, your robot must move autonomously
- The last 40 seconds, you can use a joystick to drive the robot around