

Simple subsystems

Subsystems are the parts of your robot that are independently controller like collectors, shooters, drive bases, elevators, arms, wrists, grippers, etc. Each subsystem is coded as an instance of the Subsystem class. Subsystems should have methods that define the operation of the actuators and sensors but not more complex behavior that happens over time.

Creating a subsystem

```
1 package org.usfirst.frc190.GearsBotME.subsystems;
2 import edu.wpi.first.wpilibj.*;
3 import edu.wpi.first.wpilibj.command.Subsystem;
4 import org.usfirst.frc190.GearsBotME.RobotMap;
5
6 public class Claw extends Subsystem {
7
8     Victor motor = RobotMap.clawMotor;
9
10    public void initDefaultCommand() {
11    }
12
13    public void open() {
14        motor.set(1);
15    }
16
17    public void close() {
18        motor.set(-1);
19    }
20
21    public void stop() {
22        motor.set(0);
23    }
24 }
25
```

This is an example of a fairly straightforward subsystem that operates a claw on a robot. The claw mechanism has a single motor to open or close the claw and no sensors (not necessarily a good idea in practice, but works for the example). The idea is that the open and close operations are simply timed. There are three methods, open(), close(), and stop() that operate the claw motor. Notice that there is not specific code that actually checks if the claw is opened or closed. The open method gets the claw moving in the open direction and the close method gets the claw moving in the close direction. Use a command to control the timing of this operation to make sure that the claw opens and closes for a specific period of time.

Simple subsystems

Operating the claw with a command

```
1 package org.usfirst.frc190.GearsBotME.commands;
2 import edu.wpi.first.wpilibj.command.Command;
3 import org.usfirst.frc190.GearsBotME.Robot;
4 /**
5  *
6  */
7 public class OpenClaw extends Command {
8     public OpenClaw() {
9         requires(Robot.claw);
10        setTimeout(.9);
11    }
12
13    protected void initialize() {
14        Robot.claw.open();
15    }
16
17    protected void execute() {
18    }
19
20    protected boolean isFinished() {
21        return isTimedOut();
22    }
23
24    protected void end() {
25        Robot.claw.stop();
26    }
27
28    protected void interrupted() {
29        end();
30    }
31 }
32
```

Liquid error: undefined method `account' for nil:NilClass