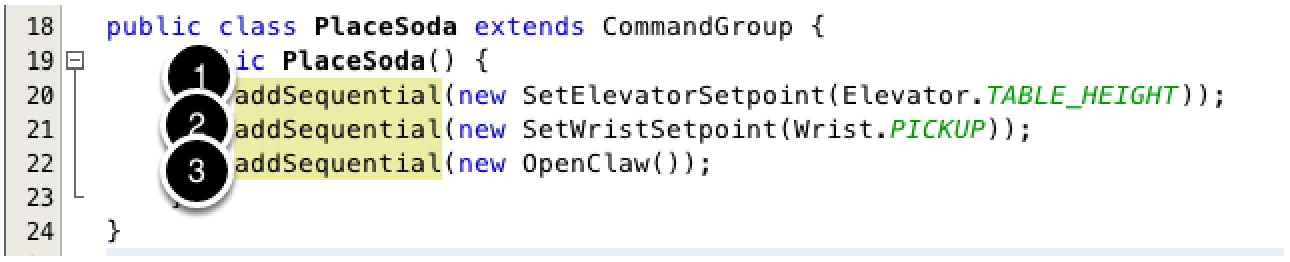
Creating groups of commands

Once you have created commands to operate the mechanisms in your robot, they can be grouped together to get more complex operations. These groupings of commands are called CommandGroups and are easily defined as shown in this article.

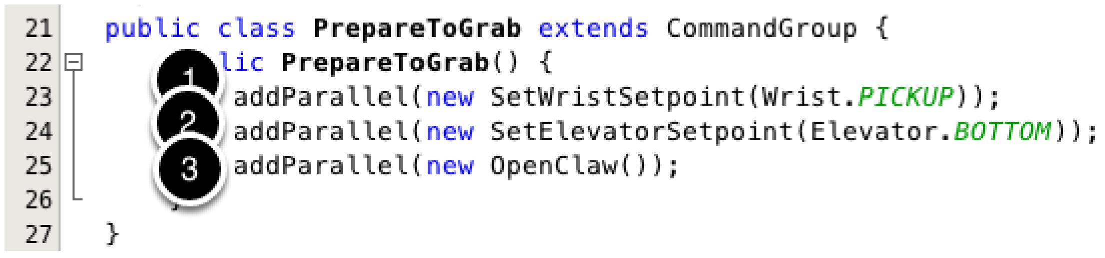
Creating a command to do a complex operation



```
public class PlaceSoda extends CommandGroup {
    public PlaceSoda() {
        addSequential(new SetElevatorSetpoint(Elevator.TABLE_HEIGHT));
        addSequential(new SetWristSetpoint(Wrist.PICKUP));
        addSequential(new OpenClaw());
    }
}
```

This is an example of a command group that places a soda can on a table. To accomplish this, (1) the robot elevator must move to the "TABLE_HEIGHT", then (2) set the wrist angle, then (3) open the claw. All of these tasks must run sequentially to make sure that the soda can isn't dropped. The addSequential() method takes a command (or a command group) as a parameter and will execute them one after another when this command is scheduled.

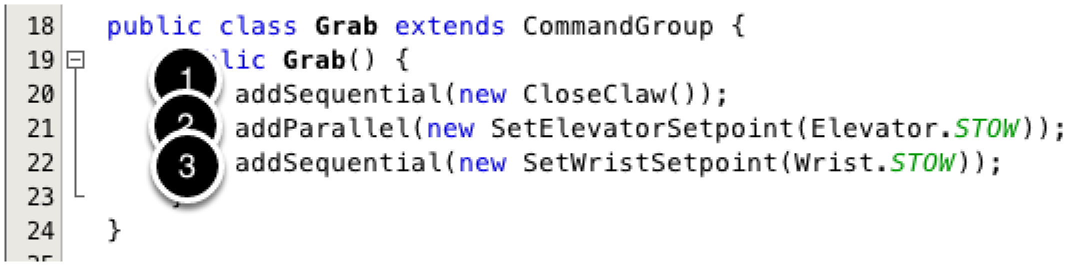
Running commands in parallel



```
public class PrepareToGrab extends CommandGroup {
    public PrepareToGrab() {
        addParallel(new SetWristSetpoint(Wrist.PICKUP));
        addParallel(new SetElevatorSetpoint(Elevator.BOTTOM));
        addParallel(new OpenClaw());
    }
}
```

To make the program more efficient, often it is desirable to run multiple commands at the same time. In this example, the robot is getting ready to grab a soda can. Since the robot isn't holding anything, all the joints can move at the same time without worrying about dropping anything. Here all the commands are run in parallel so all the motors are running at the same time and each completes whenever the isFinished() method is called. The commands may complete out of order. The steps are: (1) move the wrist to the pickup setpoint, then (2) move the elevator to the floor pickup position, and (3) open the claw.

Mixing parallel and sequential commands



```java
public class Grab extends CommandGroup {
        lic Grab() {
            addSequential(new CloseClaw());
            addParallel(new SetElevatorSetpoint(Elevator.STOW));
            addSequential(new SetWristSetpoint(Wrist.STOW));
    }
}
```

Often there are some parts of a command group that must complete before other parts run. In this example, a soda can is grabbed, then the elevator and wrist can move to their stowed positions. In this case, the wrist and elevator have to wait until the can is grabbed, then they can operate independently. The first command (1) CloseClaw grabs the soda and nothing else runs until it is finished since it is sequential, then the (2) elevator and (3) wrist move at the same time.