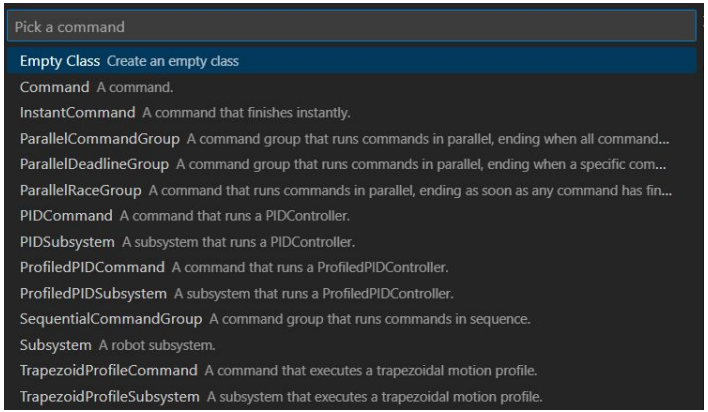
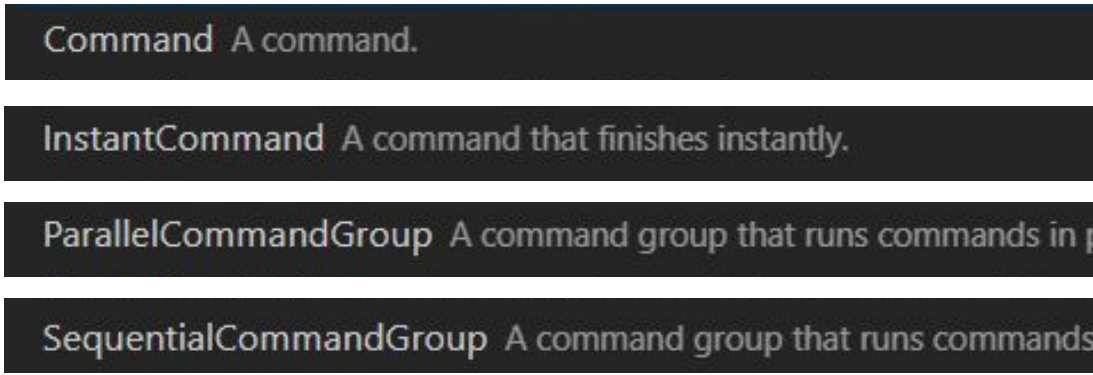
Understanding a Command Class

589 Falkon Robotics

What does a Command exactly do?

A command is the final level of code we deal with, the final level and execution we get to send to the robot. There are many types of commands, but the command types we usually work with are just just the “Command” command, the “Instant Command”, the “ParrallelCommandGroup”, and the “SequentialCommandGroup”





The Default Command

The Default Command usually consists of private variables, a constructor, and 4 Overridden methods.

```
public ExampleCommand(DriveSubsystem subsystem) {
  m_subsystem = subsystem;
  // Use addRequirements() here to declare subsystem dependencies.
  addRequirements(subsystem);
}
```

```
// Called when the command is initially scheduled.
@Override
public void initialize() {

}
// Called every time the scheduler runs while the command is scheduled.
@Override
public void execute() {

}
// Called once the command ends or is interrupted.
```

```
// Called once the command ends or is interrupted.
@Override
public void end(boolean interrupted) {

}
// Returns true when the command should end.
@Override
public boolean isFinished() {
  return false;
}
```

The Constructor

The Constructor of a default command takes in the subsystems that it will utilize to call methods on. The constructor also takes in parameters that it would need to call the necessary methods. Here is DriveDistance, an example command that makes the robot drive forward at a certain speed for a certain distance.

```
public DriveDistance(double inches, double speed, DriveSubsystem drive) {
  m_distance = inches;
  m_speed = speed;
  m_drive = drive;
  m_drive.resetEncoders();
  addRequirements(m_drive);
}
```

Method #1: `initialize()`

This method is called whenever the command starts, so anything that the command would need to have set up as soon after the objects are constructed, this is useful for resetting values or starting counters at zero.

This is an example for DriveDistance:

```
@Override
public void initialize() {
  m_drive.resetEncoders();
  m_drive.setMaxOutput(maxOutput: 1);
  m_drive.arcadeDrive(m_speed, rot: 0);
}
```

Method #2: `execute()`

This method is repeatedly called every time the command has not finished, sort of like a periodic loop until the command scheduler is either forced to stop it, or until a condition is met.

This is an example for DriveDistance:

```
@Override
public void execute() {
  m_drive.arcadeDrive(m_speed, rot: 0);
}
```

Method #3: `end()`

This method is called whenever the command is completed or forced to end with the command scheduler.

This is an example for DriveDistance:

```
@Override
public void end(boolean interrupted) {
  m_drive.arcadeDrive(fwd: 0, rot: 0);
  m_drive.resetEncoders();
}
```

Method #4: `isfinished()`

This method is called everytime execute is called aswell, if this condition is true the command finishes. This can be similar to a base case in recursion, where if it is met the method completes.

This is an example for DriveDistance:

Lab

Using this knowledge, create a command that uses the tank drive method created to turn the robot 90 degrees either left or right.