Lesson 3
The Robot Package

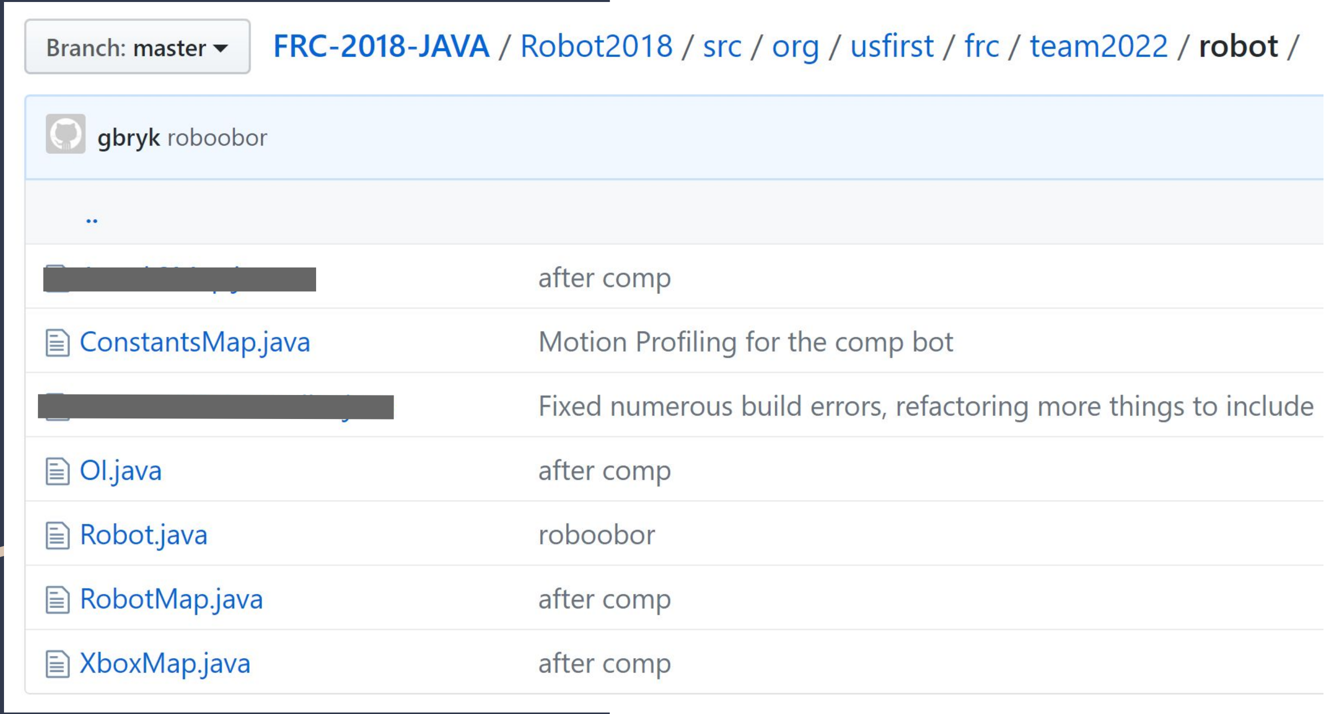
Titan Robotics Team 2022

A quick refresher

- What does the command package/command classes do for the entire FRC project

- What does the subsystem package/command classes do for the entire FRC project

The Robot Package



Branch: master

FRC-2018-JAVA / Robot2018 / src / org / usfirst / frc / team2022 / robot /

gbryk roboobor

..

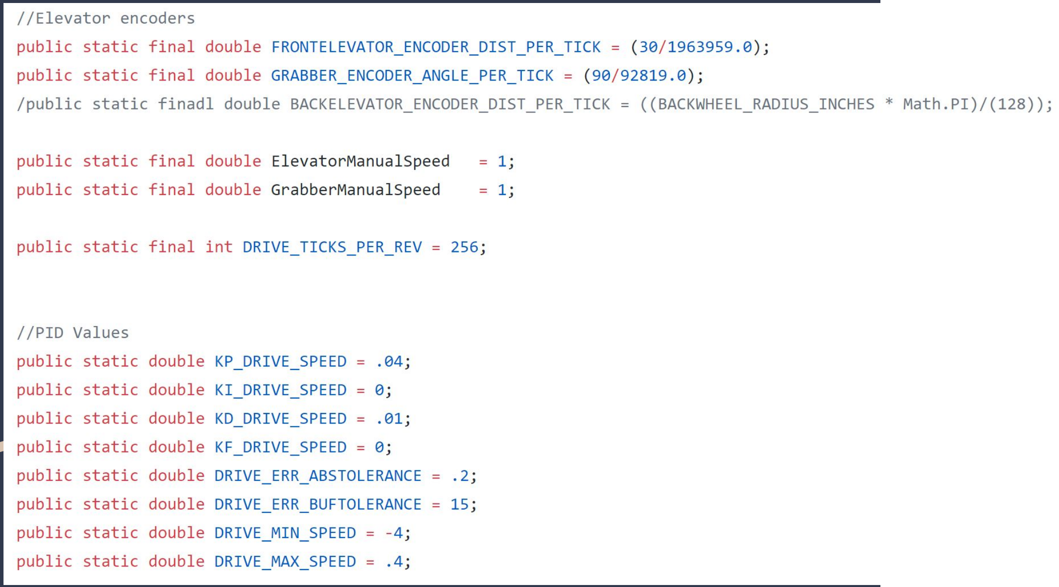
[redacted]	after comp
ConstantsMap.java	Motion Profiling for the comp bot
[redacted]	Fixed numerous build errors, refactoring more things to include
OI.java	after comp
Robot.java	roboobor
RobotMap.java	after comp
XboxMap.java	after comp

DESTINATION:
DEEP SPACE
Presented By BOEING

1. ConstantsMap.java
2. OI.java
3. Robot.java
4. RobotMap.java
5. XboxMap.java

ConstantsMap.java

- Stores <u>EVERY</u> constant in the program
- Some of the variables include:
  - Robot width and length
  - Elevator height
  - Elevator speed
  - Drive base speeds
  - And much more…
- Usually all constants have a specific type either
  - Public static final double
  - Public static double

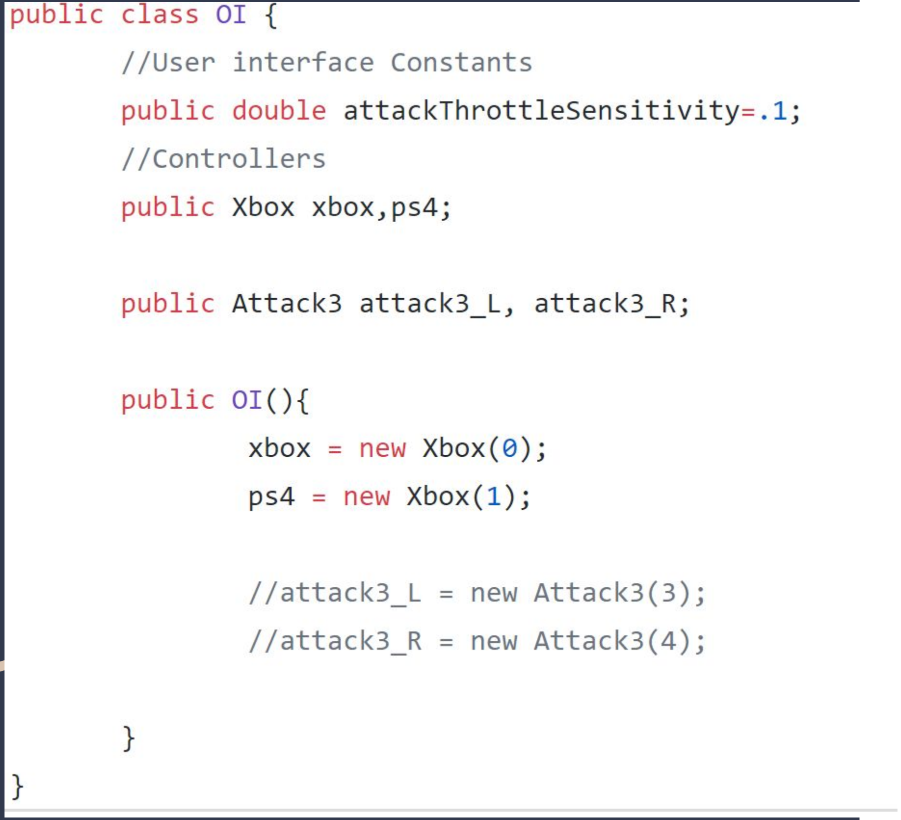


```
//Elevator encoders
public static final double FRONTELEVATOR_ENCODER_DIST_PER_TICK = (30/1963959.0);
public static final double GRABBER_ENCODER_ANGLE_PER_TICK = (90/92819.0);
/public static finadl double BACKELEVATOR_ENCODER_DIST_PER_TICK = ((BACKWHEEL_RADIUS_INCHES * Math.PI)/(128));

public static final double ElevatorManualSpeed   = 1;
public static final double GrabberManualSpeed    = 1;

public static final int DRIVE_TICKS_PER_REV = 256;



//PID Values
public static double KP_DRIVE_SPEED = .04;
public static double KI_DRIVE_SPEED = 0;
public static double KD_DRIVE_SPEED = .01;
public static double KF_DRIVE_SPEED = 0;
public static double DRIVE_ERR_ABSTOLERANCE = .2;
public static double DRIVE_ERR_BUFTOLERANCE = 15;
public static double DRIVE_MIN_SPEED = -4;
public static double DRIVE_MAX_SPEED = .4;
```

DESTINATION:
DEEP SPACE
Presented By BOEING

1. ConstantsMap.java
2. OI.java
3. Robot.java
4. RobotMap.java
5. XboxMap.java

OI.java



```
public class OI {
        //User interface Constants
        public double attackThrottleSensitivity=.1;
        //Controllers
        public Xbox xbox,ps4;

        public Attack3 attack3_L, attack3_R;

        public OI(){
                xbox = new Xbox(0);
                ps4 = new Xbox(1);

                //attack3_L = new Attack3(3);
                //attack3_R = new Attack3(4);


        }
}
```

- Should be called IO.java, but FRC is weird

- the glue that binds the controls on the physical operator interface to the commands and command groups that allow control of the robot.
- Creates the object for each controller we use during Teleop
- A VERY small class. Don’t overthink it

DESTINATION: DEEP SPACE

Presented By BOEING

1. ConstantsMap.java
2. OI.java
3. Robot.java
4. RobotMap.java
5. XboxMap.java

Robot.java

- This is the main class of the entire robot

- This is how at competitions the robot communicates with the game.

- Everything that is worked on in other classes is done for a reason and all of that gets declared/used in this class
  - Teleop runs from here (Robot.teleopInit())
  - Autoruns from here (Robot.autonomousInit())
  - What happens when you turn on the Robot?
  -

Steps for the perfect Robot.java class

I don’t think this is possible, but you guys (and girl) can prove me wrong

1. **Instantiate** all the subsystems and **declare** all the commands
2. Create an OI object for the controllers you are using
3. Robot.robotInit()
    a. Instantiate the commands and OI object
4. Robot.autonomousInit()
    a. get the game data (if the game requires it)
    b. Using the game data, start the correct auto group that was previously created in the commands package
5. Robot.teleopInit()
    a. Start all of the commands
    b. Watch the robot move with controllers!

This should be all that you have to do in the Robot.java class, however, there are a few more methods at the bottom that do not need to be edited (usually).

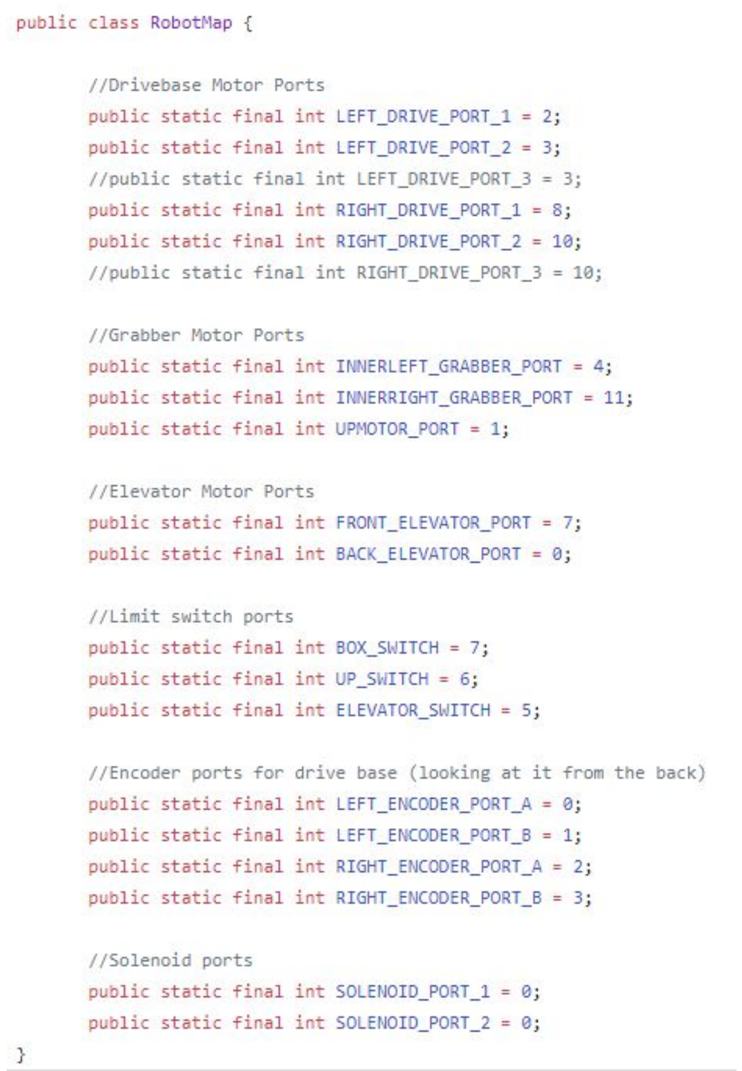
DESTINATION: DEEP SPACE

Presented By BOEING

1. ConstantsMap.java
2. OI.java
3. Robot.java
4. RobotMap.java
5. XboxMap.java

RobotMap.java

- Initialized in the Robot class
- Holds all the variables to port numbers for motors and sensors on the robot
- They always have the type
    - Public static final int
- Always comment the mechanism the variable is for.



```java
public class RobotMap {

    //Drivebase Motor Ports
    public static final int LEFT_DRIVE_PORT_1 = 2;
    public static final int LEFT_DRIVE_PORT_2 = 3;
    //public static final int LEFT_DRIVE_PORT_3 = 3;
    public static final int RIGHT_DRIVE_PORT_1 = 8;
    public static final int RIGHT_DRIVE_PORT_2 = 10;
    //public static final int RIGHT_DRIVE_PORT_3 = 10;

    //Grabber Motor Ports
    public static final int INNERLEFT_GRABBER_PORT = 4;
    public static final int INNERRIGHT_GRABBER_PORT = 11;
    public static final int UPMOTOR_PORT = 1;

    //Elevator Motor Ports
    public static final int FRONT_ELEVATOR_PORT = 7;
    public static final int BACK_ELEVATOR_PORT = 0;

    //Limit switch ports
    public static final int BOX_SWITCH = 7;
    public static final int UP_SWITCH = 6;
    public static final int ELEVATOR_SWITCH = 5;

    //Encoder ports for drive base (looking at it from the back)
    public static final int LEFT_ENCODER_PORT_A = 0;
    public static final int LEFT_ENCODER_PORT_B = 1;
    public static final int RIGHT_ENCODER_PORT_A = 2;
    public static final int RIGHT_ENCODER_PORT_B = 3;

    //Solenoid ports
    public static final int SOLENOID_PORT_1 = 0;
    public static final int SOLENOID_PORT_2 = 0;
}
```

1. ConstantsMap.java
2. OI.java
3. Robot.java
4. RobotMap.java
5. XboxMap.java

XboxMap.java

XboxMap.java

- Where the controller interacts with the robot
- Each button/bumper/joystick has a method
- Using its method we can find out when the button is moved from original state and map it to a specific function for the robot itself
- Using the OI class and the controllers that are in that class, we program every mechanism our robot has

```java
public boolean piston() {
        return oi.xbox.getLeftBumperValue();
}

public boolean inTake() {
        return oi.xbox.getAValue();
}
public boolean override() {
        return oi.xbox.getXValue();
}

public boolean outTake() {
        return oi.xbox.getBValue();
}
public boolean shiftLow() {
        return oi.ps4.getBValue();
}
public boolean shiftHigh() {
        return oi.ps4.getAValue();
}
```

```java
public double actuate() {
        return oi.xbox.getLeftY();
}

public double right() {
        return oi.ps4.getRightY();
}

public double left() {
        return oi.ps4.getLeftY();
}
```

Partners for this week

1. Matt H & Travis
2. Liana & Archan
3. Jake & Teodor
4. Brett & Sachin
5. Matt S & Aaron

Assignment

Write robot package for the car that everyone created together.

Each group from last week worked on a specific mechanisms commands and subsystems, I will compile the list for you.

You should be able to write each class in the robot package.

You do not need to do anything with Robot.autonomousInit() and you do not need to implement the RobotMap class.

DUE: Monday, October 15th at 7pm