Chapter 1

Robot Code

Robot.java

```
package frc.robot;
3 import edu.wpi.first.wpilibj.TimedRobot;
4 import frc.robot.Mode.Autonomous;
5 import frc.robot.Mode.Disabled;
{\small 6}\>\>\> \mathbf{import}\>\>\> \mathbf{frc.robot.Mode.Simulation}\;;
7 import frc.robot.Mode.Teleop;
 8 import frc.robot.Mode.Test;
9 import frc.robot.Mode.Onabot;
10
  public class Robot extends TimedRobot {
11
     @Override public void robotInit
                                                                   .Initialize(); }
                                                 () { Onabot
12
13
     @Override public void robotPeriodic
                                                 () { Onabot
                                                                   . Periodic ();
14
15
     @Override public void autonomousInit
                                                 () { Autonomous .Initialize(); }
     @Override public void autonomousPeriodic () { Autonomous . Periodic ();
16
17
     @Override public void disabledInit
                                                 () { Disabled
                                                                   .Initialize(); }
18
     @Override public void disabledPeriodic
                                                 () { Disabled
                                                                   . Periodic ();
19
20
     @Override public void teleopInit
                                                 () { Teleop
                                                                   .Initialize(); }
21
     @Override public void teleopPeriodic
                                                 () { Teleop
                                                                   . Periodic ();
22
^{23}
                                                                   .Initialize(); }
     @Override public void testInit
                                                      Test
24
25
     @Override public void testPeriodic
                                                 () { Test
                                                                   . Periodic ();
26
27
     @Override public void simulationInit
                                                 () { Simulation .Initialize(); }
28
     @Override public void simulationPeriodic () { Simulation .Periodic(); }
29 }
```

Onabot.java

```
1 package frc.robot.Mode;
3 import frc.robot.Hardware.Driver;
4 import frc.robot.Hardware.Elevator;
5 import frc.robot.Hardware.Navigation;
6 import frc.robot.Hardware.Swerve;
8 public class Onabot {
9
       public static void Initialize () {
10
                       .Initialize();
           Driver
11
           Elevator
                       .Initialize();
12
           Navigation . Initialize ();
13
                       . Initialize ();
           Swerve
14
15
           // Lidar
// Sonar
                           . Initialize();
16
                           . Initialize ();
17
           // Vision
                           . Initialize ();
18
       }
19
20
       public static void Periodic () {
21
22
           Driver
                        . Display();
           Elevator
                        . Display();
^{23}
           Navigation . Display();
24
25
           Swerve
                        . Display();
26
           // Lidar
27
                           . \, Display();
           // Sonar
28
                           . Display();
           // Vision
                           . Display();
29
30
31
32 }
```

Autonomous.java

```
1 package frc.robot.Mode;
3 import edu.wpi.first.wpilibj.smartdashboard.SendableChooser;
4 import edu.wpi.first.wpilibj.smartdashboard.SmartDashboard;
5 import frc.robot.Hardware.Autopilot;
 6 import frc.robot.Hardware.Elevator;
 7 import frc.robot.Hardware.Stage;
{\small 8}\>\> \mathbf{import}\>\>\> \mathbf{frc.robot.Hardware.Swerve};\\
9 import frc.robot.Hardware.Track;
10
11 public class Autonomous {
12
       public static final String kDefault = "Nothing";
13
       public static final String kPath01 = "Path-01";
14
       public static final String kPath02 = "Path-02";
15
       public static final String kPath03 = "Path-03";
16
        {\bf public \ static \ final \ String \ kPath04 \ = "Path-04"}; \\
17
       public static final SendableChooser<String> chooser = new SendableChooser<>();
18
19
       public static void Initialize () {
20
21
            chooser.setDefaultOption("Nothing", kDefault
            chooser.setDefaultOption ("Path\_01"
22
                                                   , kPath01
            chooser.setDefaultOption("Path ∪02", kPath02
23
            chooser.setDefaultOption ("Path\_03", kPath03")
            chooser.setDefaultOption (\,"Path\_04"\,,~kPath04
25
                                                               );
            chooser.setDefaultOption("Path_05", kPath04
26
                                                               );
            SmartDashboard.putData ("PATH",
27
                                                     chooser
28
29
            Stage. Initialize ();
       }
30
31
       public static void Periodic () {
32
33
            Stage.Begin();
35
36
            switch ( chooser.getSelected() ) {
                case "Default" : Track.Track_00(); break;
37
                case "Path-01" : Track.Track_01(); break;
38
                case "Path-02" : Track.Track_02(); break;
39
                \mathbf{case} \ "Path-03" \ : \ \mathrm{Track}.\mathrm{Track}\_03\,(\,)\,; \ \mathbf{break}\,;
40
                case "Path-04" : Track.Track_04(); break;
41
                case "Path-05" : Track.Track_05(); break;
42
            }
43
44
            Stage. Next();
45
46
            if ( Stage.Number <= 100 ) { Stage.Display(); }</pre>
47
48
49
50
            SmartDashboard.putString("CURRENT_PATH", chooser.getSelected());
51
            // EXECUTE COMMANDS
52
            Elevator. Periodic();
53
            Swerve. UpdateRobotRelative( Autopilot.vx, Autopilot.vy, Autopilot.vt );
54
55
            // Example chassis speeds: 1 meter per second forward, 3 meters
56
            /\!/ per second to the left, and rotation at 1.5 radians per second
57
            // counterclockwise.
58
            // ChassisSpeeds speeds = new ChassisSpeeds(1.0, 3.0, 1.5);
59
60
            // THESE NEED TO BE SET BY THE AUTONOMOUS MODE
61
62
               double \ curPitch = Navigation. GetPitch();
63
            // SmartDashboard.putNumber("Robot-Pitch", curPitch);
// SmartDashboard.putNumber("Robot-Stage", stage);
64
65
```

```
66
             // Navigation. Periodic ();
67
             // double curAng = Navigation. GetDirection();
68
69
             // double target = 0;
70
71
             //\  \, \textit{double}\  \, \textit{diff}\,=\,\textit{curAng}\,-\,\textit{target}\,;
72
             // // SMALLEST ANGLE TO SWIVEL: -180 to 180
73
             // double minTurn = ( diff + 180 ) % 360 - 180;
74
                      double\ turnMag\ =\ Math.\, abs \qquad (\ minTurn\ );
75
76
                      double turnDir = Math.signum(minTurn);
77
             // // MINIMIZE WHEEL SWIVEL: +120 becomes -60
78
             // if ( turnMag > 0 ) {
                     // turnMag = 180 - minTurn; // Turn smaller angle turnDir <math>*= -1; // and reverse swivel
80
81
82
83
             // // DETERMINE POWER USING PSEUDO PID CONTROLLER
             // if ( turnMag > 20 ) { vt = 0.15; } // else if ( turnMag > 10 ) { vt = 0.10; }
85
86
             // else if ( turnMag > 3 ) { vt = 0.06; }
87
             // else if ( turnMag > 2 ) { vt = 0.00; }
88
             // else if ( turnMag > 1 ) { vt = 0.00; }
89
                                               \{ vt = 0.00; \}
90
             // else
91
             // vt *= turnDir;
92
93
             // SET MOTOR CONTROLLERS
94
             // SteerMotor.set( TalonFXControlMode.PercentOutput, PID.calculate( SP, PV ) );
95
96
             // \ double \ diff = (cur) \% 360 - 180;
97
        }
99
100
101 }
```

Teleop.java

```
1 package frc.robot.Mode;
3 import edu.wpi.first.wpilibj.Joystick;
4 import frc.robot.Hardware.Driver;
5 import frc.robot.Hardware.Elevator;
6 import frc.robot.Hardware.Settings;
8 public class Teleop {
9
10
       public static Joystick DriveStick;
       public static Joystick ManipStick;
11
12
13
       public static double Xratio;
       public static double Yratio;
14
15
       public static double Tratio;
16
       public static void Initialize () {
17
           {\tt DriveStick} = {\tt new} \ {\tt Joystick} ( \ {\tt Settings.DriveStickID} \ );
18
           ManipStick = new Joystick ( Settings.ManipStickID );
19
20
21
       public static void Periodic () {
22
           Xratio = -DriveStick.getY();
^{23}
           Yratio = -DriveStick.getX();
24
           Tratio = -DriveStick.getTwist();
25
26
27
           Driver
                   . Periodic ();
28
           Elevator . Periodic();
29
       }
30
       public static void Display () {
31
32
33 }
```

Autopilot.java

The Autopilot methods are used in Autonomous mode to set the chassis speed variable found in this class. Values are sent to motor controllers in Autonomous.Periofic().

```
package frc.robot.Hardware;
3 public class Autopilot {
4
5
     public static double LastHeading = 0;
6
     public static double vx = 0;
7
     public static double vy = 0;
     public static double vt = 0;
9
10
11 /
12 // HeadingDiff is a simple method that calculates the angle difference
13 /\!/ between the current and desired heading. This can be used anywhere.
14 //
     public static double HeadingDiff ( double SP ) {
15
16
         // CALCULATE TURN VALUES
17
           double PV = Navigation.GetDirection(); // Current state (Initial)
18
                                                   // Ensure SP is between 0 and 360
19
               SP = (SP + 360) \% 360;
                                                   // Why is this negated? Should setInverted have been used?
20
               double diff = -(SP - PV);
21
           // SMALLEST ANGLE TO SWIVEL: -180 to 180
           double minTurn = (diff + 180) \% 360 - 180;
23
24
         return minTurn;
^{25}
26
27 //
^{28} // This is a simple method for driving somewhat straight without using
29 // a gyroscope. There may be situations where it is good enough.
30 //
     public static void DriveSortaStraight (double Vx, double Vy) {
31
32
         vx = Vx; vy = Vy; vt = 0;
33
34
     // Consider turning this into a pseudo tank drive for purposes of
35
     // driving in a straigh line using the gyroscope.
36
37
     public static void DriveStraight (double Vx, double Vy) {
         vx = Vx; vy = Vy; vt = 0;
38
39
40
41 //
42 //
43 //
     public static void DriveNorth ( double Speed ) {
44
         vx = +Speed; vy = 0; vt = 0;
45
46
47
48
     public static void DriveSouth ( double Speed ) {
49
         vx = -Speed; vy = 0; vt = 0;
50
51
     public static void DriveWest ( double Speed ) {
52
53
         vx = 0; vy = +Speed; vt = 0;
54
55
     public static void DriveEast ( double Speed ) {
56
         vx = Speed; vy = -Speed; vt = 0;
57
58
59
60 //
61 //
62 //
     public static void DriveNorthWest ( double Speed ) {
```

```
double radians = Math.toRadians(45);
64
65
         double speed = Speed * Math.cos( radians );
         vx = +speed; vy = +speed; vt = 0;
66
67
68
      public static void DriveNorthEast ( double Speed ) {
69
70
         double radians = Math.toRadians(45);
         double speed = Speed * Math.cos( radians );
71
         vx = +speed; vy = -speed; vt = 0;
72
73
74
      public static void DriveSouthWest ( double Speed ) {
75
         double radians = Math.toRadians(45);
76
         double speed = Speed * Math.cos( radians );
77
         vx = -speed; vy = +speed; vt = 0;
78
79
80
      public static void DriveSouthEast ( double Speed ) {
81
         double radians = Math.toRadians(45);
82
         double speed = Speed * Math.cos( radians );
83
84
         vx = -speed; vy = -speed; vt = 0;
      }
85
86
87 /
88 // TurnToHeading sets the turn power variable in Autonomous mode to reach
   // the desired heading using the shortest wheek swivel.
89
90
      public static void TurnToHeading ( double NewHeading ) {
91
         double minTurn = HeadingDiff( NewHeading );
92
93
         double turnMag = Math.abs
                                      ( minTurn );
94
         double turnDir = Math.signum( minTurn );
95
            // MINIMIZE WHEEL SWIVEL: +120 becomes -60
            \mathbf{if} ( \operatorname{turnMag} > 0 ) {
97
                turnMag \ = 180 - turnMag; \ /\!/ \ \textit{Turn smaller angle}
98
                turnDir *= -1;
                                           // and reverse swivel
99
            }
100
101
            // DETERMINE POWER USING PSEUDO PID CONTROLLER
102
103
                   (turnMag > 20) \{ vt = 0.20;
            else if (turnMag > 10) \{vt = 0.10;
104
            else if ( turnMag > 1 ) { vt = 0.08; }
105
                                      \{ vt = 0.00; \}
107
         LastHeading = NewHeading;
108
         vx = 0; vy = 0; vt *= turnDir;
109
110
111
112 /
113 // Stop sets the robot speed vector to zero. This is useful only in Autonomous
114 // mode. It should not be used elsewhere.
115 //
      public static void Stop () {
116
117
         vx = 0; vy = 0; vt = 0;
118
119
121 //
      These methods rotate the robot at a constant counter-clockwise speed and
      clockwise speed respectively. This is only useful in Autonomous mode.
122 //
123 //
      public static void TurnLeftAtSpeed ( double Speed ) {
124
         vx = 0; vy = 0; vt = +Speed;
125
126
127
128
      public static void TurnRightAtSpeed ( double Speed ) {
         vx = 0; vy = 0; vt = -Speed;
129
130
131
```

Driver.java

```
package frc.robot.Hardware;
2
3 import edu.wpi.first.wpilibj.smartdashboard.SendableChooser;
4 import edu.wpi.first.wpilibj.smartdashboard.SmartDashboard;
5 import frc.robot.Driver.Aubrey;
{\small 6}\>\> \mathbf{import}\>\>\> \mathbf{frc.robot.Driver.Default}\;;\\
7 import frc.robot.Driver.Nate;
8 import frc.robot.Driver.Steensma;
10 public class Driver {
11
       public static final String kDefault = "Default";
12
                                                = "Aubrey";
13
       public static final String kAubrey
                                                = "Nate";
       public static final String kNate
14
       public static final String kSteensma = "Steensma";
15
       public static final SendableChooser<String> chooser = new SendableChooser<>();
16
17
       public static void Initialize () {
18
           {\tt chooser.setDefaultOption("Default"}\,,
                                                     kDefault
19
                                                                );
20
            chooser\,.\,addOption
                                      ("Aubrey"
                                                     kAubrey
                                                                 );
                                       ("Nate",
21
           chooser.addOption
                                                     kNate
           {\tt chooser.addOption}
                                       ("Steensma", kSteensma
22
                                                                );
           SmartDashboard.putData \quad \left( \text{"DRIVER"} \right.,
^{23}
                                                     chooser
       }
24
25
       public static void Periodic () {
26
27
           switch ( chooser.getSelected() ) {
                case "Default" : Default . Periodic();
28
                                             .Periodic();
29
                case "Aubrey"
                                : Aubrey
                case "Nate"
30
                                 : Nate
                                              . Periodic ();
                case "Steensma": Steensma . Periodic ();
31
32
           }
33
       }
34
35
       public static void Display () {
           SmartDashboard.putString("Driver", chooser.getSelected());
36
37
38
39 }
```

Elevator.java

```
package frc.robot.Hardware;
3 public class Elevator {
       public static void Initialize () {
5
6
            ElevArm
                          .Initialize();
            ElevClaw
                          .Initialize();
7
            ElevLift
                         .Initialize();
            {\bf ElevWrist}
                         .Initialize();
9
10
11
       public static void Periodic () {
12
                          . Periodic ();
13
            {\bf ElevArm}
                          .Periodic ();
            ElevClaw
14
15
            ElevLift
                          . Periodic ();
            ElevWrist
                         . Periodic ();
16
17
       }
18
       public static void Display () {
19
20
            {\bf ElevArm}
                          . Display();
            ElevClaw
                          .\, {\rm Display}\, (\,)\, ;
21
            ElevLift
                          . Display();
22
            {\bf ElevWrist}
                         . Display();
^{23}
24
       }
25
        public static void Reset () {
26
27
            {\bf ElevArm}
                          . Reset();
                          .Reset();
            ElevClaw
28
                          . Reset ();
            ElevLift
29
            {\bf ElevWrist}
30
                          . Reset();
31
32
        public static void Set ( double A, double C, double L, double W ) \{
33
                         .SetPosition( A );
.SetPosition( C );
            ElevArm
34
35
            ElevClaw
                          . SetPosition (L);
            ElevLift
36
37
            ElevWrist
                         .SetPosition(W);
       }
38
39
40 }
```

ElevArm.java

```
package frc.robot.Hardware;
3 import edu.wpi.first.wpilibj.smartdashboard.SmartDashboard;
5 public class ElevArm {
       public static double target_position = 0;
9 //
10 //
11 //
       public static void Initialize () {
12
13
14
15
       public static void Periodic () {
16
           // Some sort of controller to find the position
17
           // difference and set the motor ratio. Might need
18
           // a PID controller to hold position.
19
20
21
       public static void Display () {
22
           SmartDashboard.putNumber("Elevator-Arm\_Pos", \ GetPosition()
^{23}
           SmartDashboard.putNumber("Elevator-Arm_{\sqcup}Tar", target\_position");\\
24
25
26
27 //
28 //
29 //
       public static double GetPosition () {
30
31
           return 0;
32
33
       public static void SetPosition ( double pos ) {
34
35
           target\_position = pos;
36
37
38 //
39 //
40 //
       public static void Reset () {
41
42
           Retract();
43
44
       public static void Extend () {
45
46
47
48
49
       public static void Retract () {
50
51
52
53 }
```

ElevClaw.java

```
package frc.robot.Hardware;
3 import edu.wpi.first.wpilibj.smartdashboard.SmartDashboard;
5 public class ElevClaw {
       public static double target_position;
9 //
10 //
11 //
       public static void Initialize () {
12
13
14
15
       public static void Periodic () {
16
           // Some sort of controller to find the position
17
           // difference and set the motor ratio. Might need
           // a PID controller to hold position.
19
20
21
       public static void Display () {
22
           SmartDashboard.putNumber("Elevator-Claw\_Pos", \ GetPosition()
^{23}
           SmartDashboard.putNumber("Elevator-Claw_{\sqcup}Tar"\,,\ target\_position\ );
24
25
26
27 //
28 //
29 //
       public static void Reset () {
30
31
           Open();
32
33
       public static void Grab () {
34
35
36
37
       public static void Open () {
38
39
40
41
42 //
43 //
44 //
       {\bf public\ static\ double\ GetPosition\ ()\ \{}
45
           return 0;
46
47
48
49
       public static void SetPosition ( double pos ) {
           target\_position = pos;
50
51
52
53 }
```

ElevLift.java

```
package frc.robot.Hardware;
3 import edu.wpi.first.wpilibj.smartdashboard.SmartDashboard;
5 public class ElevWrist {
       public static double target_position;
9 //
10 //
11 //
       public static void Initialize () {
12
13
14
15
       public static void Periodic () {
16
            // Some sort of controller to find the position
17
            // difference and set the motor ratio. Might need
18
            // a PID controller to hold position.
19
20
21
       public static void Display () {
22
            SmartDashboard.putNumber("Elevator-Wrist_{\sqcup}Pos"\,,\ GetPosition\,()
^{23}
            SmartDashboard.putNumber("Elevator-Wrist_{\sqcup}Tar", target\_position");
24
25
26
27 //
28 //
29 //
       public static double GetPosition () {
30
31
            return 0;
32
33
       public static void SetPosition ( double pos ) {
34
35
            target\_position = pos;
36
37
38 //
39 //
40 //
       public static void Reset () {
41
42
           Bend();
43
44
       {\bf public\ static\ void\ }{\rm Bend\ }(\ )\ \{
45
46
47
48
49
       public static void Straighten () {
50
51
52
53 }
```

ElevWrist.java

```
package frc.robot.Hardware;
3 import edu.wpi.first.wpilibj.smartdashboard.SmartDashboard;
5 public class ElevWrist {
       public static double target_position;
9 //
10 //
11 //
       public static void Initialize () {
12
13
14
15
       public static void Periodic () {
16
           // Some sort of controller to find the position
17
           // difference and set the motor ratio. Might need
18
           // a PID controller to hold position.
19
20
21
       public static void Display () {
22
           SmartDashboard.putNumber("Elevator-Wrist_{\sqcup}Pos"\,,\ GetPosition\,()
^{23}
           SmartDashboard.putNumber("Elevator-Wrist_{\sqcup} Tar", target\_position");
24
25
26
27 //
28 //
29 //
       public static double GetPosition () {
30
31
           return 0;
32
33
       public static void SetPosition ( double pos ) {
34
35
           target\_position = pos;
36
37
38 //
39 //
40 //
       public static void Reset () {
41
42
           Bend();
43
44
       public static void Bend () {
45
46
47
48
49
       public static void Straighten () {
50
51
52
53 }
```

${\bf EncTalon FX. java}$

```
package frc.robot.Hardware;
import com.ctre.phoenix.sensors.CANCoder;

public class EncTalonFX {

public CANCoder FalconEncoder;
public final static int kUnitsPerRevolution = 2048;

public EncTalonFX ( int CanBusID ) {
    FalconEncoder = new CANCoder( CanBusID );
}
```

Module.java

```
package frc.robot.Hardware;
3 import com. ctre.phoenix.motorcontrol.NeutralMode;
4 import com. ctre.phoenix.motorcontrol.can.WPI TalonFX;
7 import edu.wpi.first.math.kinematics.SwerveModuleState;
8 import edu.wpi.first.wpilibj.smartdashboard.SmartDashboard;
10 public class Module {
11
12
      EncTalonFX
                     SteerEncoder;
      int
                     ModuleNumber:
13
       String
                     ModuleName;
14
      WPI\_TalonFX
15
                     DriveMotor;
      WPI_TalonFX
                     SteerMotor;
16
      PIDController
                     SteerPID;
17
      double
                     SpeedPlus
18
      double
                     LastPosition = 0;
19
      boolean
                     still_turning_flag = true;
20
21
22
       public Module ( String ModuleName, int ModuleNumber ) {
23
           // REMEMBER VALUES
           this. ModuleName = ModuleName;
25
           this. ModuleNumber = ModuleNumber;
26
27
           // ID 'S FOLLOW A PATTERN BASED ON MODULE NUMBERS
28
29
          int DriveMotorID = ModuleNumber *2 -1;
          int SteerMotorID = ModuleNumber *2 -0;
30
31
           // DEFINE AND CONFIGURE DRIVE MOTOR
32
           DriveMotor = new WPI TalonFX ( DriveMotorID );
33
           DriveMotor.setNeutralMode( NeutralMode.Brake );
35
36
           // DEFINE AND CONFIGURE STEER MOTOR
           SteerMotor = new WPI_TalonFX ( SteerMotorID );
37
           SteerMotor.setNeutralMode( NeutralMode.Brake );
38
39
           // DEFINE STEER ENCODER
40
41
           SteerEncoder = new EncTalonFX ( ModuleNumber );
42
43
      public void Display () {
44
45
           // STEER ENCODER
46
           SmartDashboard.putNumber(ModuleName + "\_PV", this.SteerEncoder.FalconEncoder.getAbsolutePosition());
47
48
49
50
      public void ResetDriveEncoder () {
           // DriveMotor
51
52
53
      public double GetDirection () {
54
          return SteerEncoder.FalconEncoder.getAbsolutePosition();
55
56
57
      public void Update ( SwerveModuleState state ) {
58
59
60
           // CALCULATE DRIVE VALUES
          double DriveRatio = state.speedMetersPerSecond;
61
          double reverse
                             = 1;
62
63
           // CALCULATE TURN VALUES
64
          double SP = state.angle.getDegrees(); // Desired state (Final)
65
```

```
// Current state (Initial)
             double PV = GetDirection();
66
67
                  SP = (SP + 360) \% 360;
                                                           // Ensure SP is between 0 and 360
68
69
             // SMALLEST ANGLE TO SWIVEL: -180 to 180
             double minTurn = (PV - SP + 180) \% 360 - 180;
70
                  double turnMag = Math.abs ( minTurn );
71
                  double turnDir = Math.signum( minTurn );
72
73
             // MINIMIZE WHEEL SWIVEL: +120 becomes -60
74
             if ( turnMag > 90 ) {
75
                  turnMag = 180 - turnMag; // Turn smaller angle
turnDir *= -1; // and reverse swivel
76
77
                                                 // and reverse drive
                  reverse *= -1;
78
             }
80
             // DETERMINE POWER USING PSEUDO PID CONTROLLER
81
             double SteerRatio = 0;
82
                      (turnMag > 20) \{ SteerRatio = 0.20; \}
83
             else if ( turnMag > 10 ) { SteerRatio = 0.08; }
             else if ( turnMag > 1 ) { SteerRatio = 0.07; }
85
86
                                           { SteerRatio = 0.00; }
87
             // If any the heading difference of any wheel is more than one degree and the
88
             // module has not turned in the last 20 ms then increase the turning speed.
89
             double CurrentPosition = SteerEncoder.FalconEncoder.getAbsolutePosition();
90
             boolean is_moving = CurrentPosition == LastPosition ? false : true;
91
             \begin{array}{lll} \textbf{if} & (& turnMag < 1 \\ \textbf{else} & \textbf{if} & (& ! & is\_moving \\ \end{array}) \; \left\{ \begin{array}{ll} SpeedPlus \; = \; 0.000; \; \right\} \\ SpeedPlus \; += \; 0.001; \; \end{array} \right\}
92
93
             else
94
95
             // SET TURNING FLAG, CHECKED BY DRIVETRAIN
96
             still_turning_flag = turnMag >= 5 ? true : false;
97
             // RESET LAST POSITION TO SEE IF MOVEMENT OCCURED
99
             LastPosition = CurrentPosition;
100
101
             SteerRatio += SpeedPlus;
102
             // SET MOTOR CONTROLLERS
103
             DriveMotor.setVoltage( DriveRatio * 10 * reverse );
104
105
             SteerMotor.setVoltage( SteerRatio * 10 * turnDir );
106
107
108 }
```

Navigation.java

```
package frc.robot.Hardware;
2
3 import com. kauailabs.navx.frc.AHRS;
4 import edu.wpi.first.wpilibj.SPI;
5 import edu.wpi.first.wpilibj.smartdashboard.SmartDashboard;
7 public class Navigation {
        public static AHRS NavX;
9
10
        {\bf public\ static\ void\ Initialize\ ()\ \{}
11
            NavX = new AHRS(SPI.Port.kMXP);
12
13
             NavX.calibrate();
             Reset();
14
15
16
        public static void Periodic () {}
17
18
        \mathbf{public} \ \mathbf{static} \ \mathbf{void} \ \mathrm{Display}() \ \{
19
             SmartDashboard.putNumber( "Nav—Yaw", SmartDashboard.putNumber( "Nav—Pitch",
20
                                                              GetYaw()
                                                                            );
                                                              GetPitch() );
21
             SmartDashboard.putNumber("Nav-Roll",
                                                              GetRoll() );
22
^{23}
24
25 /
26 // SUPPORT METHODS
27 //
                                  Reset
28
        public static void
                                             () { NavX.reset(); }
29
         \textbf{public static double} \ \ \textbf{GetPitch} \ \ () \ \ \{ \ \ \textbf{return} \ \ \ \textbf{NavX.getPitch} \ (); \ \} \ \ /\!/ \ \textit{Forward tilt } : - \ \textit{is up} 
30
        public static double GetRoll () { return NavX.getRoll(); } // Side-to-side : + is ?
31
                                             () { return -NavX.getYaw();
                                                                                  } // Twist
32
        public static double GetYaw
33
        public static double GetDirection () { return GetYaw(); }
34
35 }
```

Settings.java

```
package frc.robot.Hardware;
3 public class Settings {
4
          // CONTROLLER PORTS
5
          public static int
 6
                DriveStickID = 0,
                ManipStickID = 1;
9
          // MAXIMUM MODULE SPEEDS
10
          public static double
11
               \label{eq:max_drift} \begin{aligned} \text{MAX\_DRIVE\_RATIO} &= \ 0.20 \,; \end{aligned}
12
13
          // MODULE ASSIGNMENTS
14
15
          public static int
                {\rm FL\_moduleNumber} \, = \, 1 \, ,
16
                FR_{moduleNumber} = 4,
17
                RL_{moduleNumber} = 5,
                RR_{moduleNumber} = 3;
19
20
          // MODULE LOCATIONS
21
          public static double
22
               FLx = 1, FLy = 1, FRx = 1, FRy = -1, RLx = -1, RLy = 1, RRx = -1, RRy = -1;
^{23}
24
25
^{26}
27
          // CLICKS PER FOOT
28
          public static double
29
               \label{eq:normalized_energy} {\tt IN\_PER\_CLICK} = \left( \begin{array}{ccc} {\tt Math.PI} \ * \ 4 \end{array} \right) \ / \ {\tt EncTalonFX.kUnitsPerRevolution} \, ;
30
31
32 }
```

Stage.java

```
package frc.robot.Hardware;
3 import edu.wpi.first.wpilibj.smartdashboard.SmartDashboard;
  public class Stage {
6
      public static double
                              AutonStartTime;
      public static double
                              AutonFinalTime;
8
                              StageStartTime;
      public static double
9
10
      public static int
11
                              Number:
12
      public static boolean
                              ReadyToAdvance;
                              StageDistance = new double [10];
      public static double[]
13
14
      public static double[] StageTime
                                             = new double [10];
15
                              NegTilt = 0;
16
      public static double
17
      public static double
                              PosTilt = 0;
18
19
      public static void Initialize () {
20
21
         AutonStartTime = System.currentTimeMillis();
22
         StageStartTime = AutonStartTime;
         Number
                        = 0;
23
24
25
      public static void Display () {
26
         SmartDashboard.putNumber("Robot-Stage \_Number"
^{27}
                                                             Number
         SmartDashboard.putNumber("Robot-Stage_Distance", GetDistance()
                                                                                 ):
28
         SmartDashboard.putNumber("Robot-Stage_Time",
29
                                                             GetStageTime()
         SmartDashboard.putNumber("Robot-Auton_Time",
                                                             GetAutonDuration() );
30
31
32
33 //
34 // The Next method advances to the next stage after storing Stage
     information. The Last method stops
35
36
      public static void Begin () {
37
         Autopilot.Stop();
38
39
         ReadyToAdvance = true;
40
41
      public static void Next () {
42
43
         if ( ReadyToAdvance == true ) {
            StageDistance[Number] = GetDistance();
44
            StageTime
                          [Number] = GetStageTime();
45
46
            ResetOdometer();
            StageStartTime = System.currentTimeMillis();
47
            Number++;
48
49
50
51
      public static void Last () {
52
         AutonFinalTime = System.currentTimeMillis();
53
         ReadyToAdvance = false;
54
55
56
         Autopilot.Stop();
      }
57
58
      public static void Fail () {
59
60
         AutonFinalTime = System.currentTimeMillis();
         ReadyToAdvance = false;
61
         Number
62
63
64
65 //
```

```
66 // Get... Time methods are is useful in auton mode to determine the amount
67 // of time that the current stage or the entire auton process has been
68 // executing.
69 //
      public static double GetAutonDuration () {
70
         return ( System.currentTimeMillis() - AutonStartTime ) / 1000.0;
71
72
73
      public static double GetStageTime () {
74
         return ( System.currentTimeMillis() - StageStartTime ) / 1000.0;
75
76
77
      public static void WaitForDuration ( double Duration ) {
78
          if ( GetStageTime() < Duration ) {</pre>
80
             ReadyToAdvance = false;
81
      }
82
83
84
   //
85
86
      public static double GetDistance () {
87
         double FL = Swerve.FL module.DriveMotor.getSelectedSensorPosition();
88
         double FR = Swerve.FL_module.DriveMotor.getSelectedSensorPosition();
89
         double RL = Swerve.FL_module.DriveMotor.getSelectedSensorPosition();
90
         double RR = Swerve.FL_module.DriveMotor.getSelectedSensorPosition();
91
92
          // ABS SINCE SOME WHEELS GOING BACKWARD
93
         FL = Math.abs(FL);
94
         FR = Math.abs(FR);
95
         RL = Math.abs(RL);
96
         RR = Math.abs(RR);
97
          // TAKE AN AVERAGE FOR SIMPLICITY
99
         return (FL + FR + RL + RR) * Settings.IN_PER_CLICK / 4;
100
101
102
103
      public static void ResetOdometer () {
         Swerve.FL_module.DriveMotor.setSelectedSensorPosition(0);
104
105
         Swerve.FR_module.DriveMotor.setSelectedSensorPosition(0);
106
         Swerve.RL_module.DriveMotor.setSelectedSensorPosition(0);
107
         Swerve.RR_module.DriveMotor.setSelectedSensorPosition(0);
108
109
      public static void WaitForDistance ( double Distance ) {
110
          if \ (\ \mathrm{GetDistance}\,(\,)\,<\,\mathrm{Distance}\,\,)\ \{
111
             ReadyToAdvance = false;
112
113
114
115
116 //
117 //
118
      public static void WaitForHeading ( double Heading, double Tolerance ) {
119
120
         double diff = Autopilot. Heading Diff( Heading );
          if ( Math.abs( diff ) < Tolerance ) {</pre>
121
             ReadyToAdvance = false;
122
123
124
      }
125
126 /
127 // Second draft of code to be used in auton. Drive forward until we notice an incline.
128 // At that point we advance stages and continue to drive forward until we notice a
129 // balanced condition. It would be good to also have a maximum distance travelled for
   // each stage and fail if the condition is not met.
130
131 //
      public static void WaitForBalance ( double Tolerance ) {
132
         double pitch = Navigation.GetPitch();
133
```

```
if ( Math.abs( pitch ) > Tolerance ) {
134
135
              ReadyToAdvance = false;
136
137
       }
138
139
       public static void WaitForIncline ( double Angle ) {
           double pitch = Navigation.GetPitch();
140
           if (Math.abs(pitch) < Angle) {
141
142
              ReadyToAdvance = false;
143
144
       }
145
146 //
147 //
148 //
       //\ public\ static\ void\ WaitForWheelAlignment\ (\ double\ Angle\ )\ \{
149
150
151
       // public static void WaitForHeading ( double Heading, double Tolerance ) {
152
153
154
155
156 // public static boolean WaitForHeading (double targetHeading, double tolerance ) {
157 //
           if \ (\textit{Math.abs}(\textit{Navigation}. \textit{GetDelta}(\textit{targetHeading})) < \textit{tolerance} \ ) \ \{ \textit{ return true}; \ \}
158 // }
159 //
160 //
161 // public static boolean WaitForTarget( double tolerance ) {
162 //
           if \ (\ Drivetrain.\ TargetMin <\!\!-tolerance\ //\ Drivetrain.\ TargetMax \!\!>\! tolerance\ )\ \{
              StillWorking = true;
163 //
164 //
              return\ true;
165 //
166 //
           else {
167 //
              return false;
168 //
169 //
170 // }
171
172
173 }
```

Swerve.java

```
package frc.robot.Hardware;
3 import edu.wpi.first.math.geometry.Rotation2d;
4 import edu.wpi.first.math.geometry.Translation2d;
 5 import edu.wpi.first.math.kinematics.ChassisSpeeds;
 6 import edu.wpi.first.math.kinematics.SwerveDriveKinematics;
 7 import edu.wpi.first.math.kinematics.SwerveModuleState;
8 import edu.wpi.first.wpilibj.smartdashboard.SmartDashboard;
10 public class Swerve {
11
       // CHASSIS SPEEDS
12
       public static ChassisSpeeds
13
14
            RobotSpeed;
15
       // MODULE DEFINITIONS
16
       public static Module
17
           FL module.
18
           FR_module,
19
           RL_module,
20
21
           RR_module;
22
       // TRANSLATION OBJECTS
23
       static Translation2d
24
           FL_Trans2d,
25
           FR Trans2d,
26
           RL_Trans2d,
27
           RR Trans2d;
28
29
       // KINEMATICS OBJECT
30
       public static SwerveDriveKinematics
31
32
            Kinematics:
33
       // INITIALIZE
34
       public static void Initialize () {
35
36
            // CHASSIS SPEEDS
37
            RobotSpeed = new ChassisSpeeds(0, 0, 0);
38
39
            // MODULE DEFINITIONS
40
           FL_module = new Module( "FL", Settings.FL_moduleNumber );
41
           FR_module = new Module( "FR", Settings.FR_moduleNumber);
42
           \label{eq:RL_module} RL\_module = \textbf{new} \ \operatorname{Module} ( \ "RL" \, , \ \operatorname{Settings.RL\_moduleNumber} \ ) \, ;
43
           RR_module = new Module("RR", Settings.RR_moduleNumber);
44
45
            // TRANSLATION OBJECT
46
           FL Trans2d = new Translation2d( Settings.FLx, Settings.FLy );
47
           FR_Trans2d = new Translation2d( Settings.FRx, Settings.FRy );
48
           RL\_Trans2d = \textbf{new} \ Translation2d ( \ Settings.RLx, \ Settings.RLy \ );
49
50
           RR_Trans2d = new Translation2d( Settings.RRx, Settings.RRy);
51
            // KINEMATICS OBJECT
52
            Kinematics = new SwerveDriveKinematics( FL_Trans2d, FR_Trans2d, RL_Trans2d, RR_Trans2d);
53
54
       }
55
56
       public static void Display () {
            SmartDashboard.putNumber ("\r{R}obot-vx", RobotSpeed.vxMetersPerSecond") \\
                                                                                          ):
57
            SmartDashboard.putNumber (\,"Robot-vy"\;,\;\;RobotSpeed.vyMetersPerSecond
58
            SmartDashboard.putNumber("Robot-vt", RobotSpeed.omegaRadiansPerSecond);
59
60
61
           FL_module. Display();
           FR_module.Display();
62
63
           RL_module. Display();
           RR_module. Display();
64
65
```

```
66
67
       public static void UpdateFieldRelative ( double vx, double vy, double vt ) {
                          Rot2d = Rotation2d.fromDegrees( Navigation.GetYaw() );
           Rotation2d
68
69
           ChassisSpeeds Speeds = ChassisSpeeds.fromFieldRelativeSpeeds(vx, vy, vt, Rot2d);
           Update(Speeds);
70
       }
71
72
       public static void UpdateRobotRelative ( double vx, double vy, double vt ) {
73
74
           ChassisSpeeds Speeds = new ChassisSpeeds (vx, vy, vt);
           Update(Speeds);
75
76
77
       private static void Update ( ChassisSpeeds Speeds ) {
78
79
            // WAIT FOR WHEEL TO ADJUST TO HEADING
80
           boolean ok_to_drive = true;
81
           if (FL_module.still_turning_flag ) { ok_to_drive = false; }
82
           if (FR_module.still_turning_flag ) { ok_to_drive = false;
83
           if ( RL_module.still_turning_flag ) { ok_to_drive = false;
           if ( RR_module.still_turning_flag ) { ok_to_drive = false; }
85
86
            // ALIGN WHEELS BEFORE TRANSLATION
87
           if (! ok to drive) {
88
                Speeds.vxMetersPerSecond = 0;
89
                Speeds.vyMetersPerSecond = 0;
90
           }
91
92
            // CALCULATE INDIVIDUAL MODULE STATES
93
           SwerveModuleState[] ModuleStates = Kinematics.toSwerveModuleStates( Speeds );
94
95
            // NORMALIZE WHEEL RATIOS IF ANY SPEED IS ABOVE SPECIFIED MAXIMUM
96
           SwerveDriveKinematics.desaturateWheelSpeeds(ModuleStates, Settings.MAX_DRIVE_RATIO);
97
98
            // UPDATE ROBOT SPEEDS
99
           RobotSpeed = Kinematics.toChassisSpeeds( ModuleStates );
100
101
            // UPDATE EACH MODULE
102
103
           FL_module.Update( ModuleStates[0] );
           FR\_module.\,Update(\ ModuleStates\,[\,1\,]
104
105
           RL_module.Update( ModuleStates [2]
           RR_module. Update( ModuleStates [3] );
106
107
       }
108
109 }
```

Track.java

```
1 package frc.robot.Hardware;
3 public class Track {
       public static void Track_00 () {
           switch (Stage.Number) {
6
                case 0:
                    Autopilot.Stop();
8
                    Stage.WaitForDuration(2.00);
9
10
                    break;
11
12
                default:
                    Stage.Last();
13
14
                    break;
           }
15
16
       }
17
       public static void Track_01 () {
18
           switch ( Stage.Number ) {
19
                case 0:
20
21
                     Stage. WaitForDuration (1.00);
^{22}
                    break;
23
                \mathbf{default}:
                    Stage.Last();
25
                    break;
26
           }
^{27}
       }
28
29
       public static void Track_02 () {
30
           switch ( Stage.Number ) {
31
                \mathbf{case} \ \ 0\colon
32
                    Stage. WaitForDuration(2.00);
33
                    break;
35
36
                default:
                    Stage.Last();
37
                    break;
38
39
           }
40
41
       public static void Track\_03 () {
42
43
           switch ( Stage.Number ) {
44
                case 0:
                     Autopilot.DriveNorth(0.20);
45
                     Stage.WaitForDistance( 20 * 100 );
46
                    break;
47
48
                case 1:
49
50
                     Autopilot.TurnToHeading(0);
51
                     Stage.WaitForDuration(3.00);
                    break;
52
53
                // case 2:
54
                        Autopilot. DriveSouth (0.20);
55
56
                        Stage.\ WaitFor Distance (\ 136\ );
                        break;
57
                {\bf default}:
59
60
                     Stage.Last();
                    break;
61
62
           }
63
64
       public static void Track_04 () {
```

```
\mathbf{switch} ( \mathbf{Stage.Number} ) {
66
67
                  case 0:
                       {\bf Autopilot.TurnToHeading(~90~);;}
68
69
                       Stage. WaitForDuration(2.00);
                       break;
70
71
                  case 1:
72
                       {\tt Autopilot.TurnToHeading(\ 180\ );;}
73
                       Stage. WaitForDuration(2.00);
74
                       {\bf break}\,;
75
76
                  {\bf default}:
77
                       {\tt Stage.Last();}
78
79
                       break;
             }
80
        }
81
82
        public static void Track\_05 () {
83
84
             switch ( Stage.Number ) {
                  case 0:
85
                       Stage.WaitForDuration( 1.00 );
86
                       break;
87
88
                  {\bf default:}
89
90
                       Stage.Last();
91
                       break;
             }
92
93
94
        public static void Track_06 () {
95
             switch ( Stage.Number ) {
96
                  case 0:
97
                       Stage. WaitForDuration(1.00);
98
                       break;
99
100
                  {\bf default}:
101
                       {\tt Stage.Last();}
102
103
                       break;
             }
104
105
106
107 }
```

Default.java (Driver)

```
package frc.robot.Driver;
3 import frc.robot.Hardware.ElevLift;
4 import frc.robot.Hardware.Swerve;
5 import frc.robot.Mode.Teleop;
7
   public class Default {
        public static void Periodic () {
10
             // GET VALUES
11
            double Xratio = Teleop.Xratio;
12
            double Yratio = Teleop. Yratio;
13
            double Tratio = Teleop.Tratio;
15
             // JOYSTICK COMPONENTS
16
            double Xmag = Math.abs( Xratio ); double Xsig = Math.signum( Xratio );
17
            double Ymag = Math.abs( Yratio ); double Ysig = Math.signum( Yratio );
18
19
            double Tmag = Math.abs( Tratio ); double Tsig = Math.signum( Tratio );
20
            // APPLY DEADZONE AND SCALE SPEEDS: e.g., 0.20 is a 20% dead zone if ( Xmag < 0.10 ) { Xmag = 0; } else { Xmag = Math.pow( Xmag-0.10, 2 ) / 2; } if ( Ymag < 0.10 ) { Ymag = 0; } else { Ymag = Math.pow( Ymag-0.10, 2 ) / 2; }
21
22
23
             if ( Tmag < 0.20 ) { Tmag = 0; } else { Tmag = Math.pow( Tmag-0.20, 2) / 2; }
24
25
             // TESTING COMMANDS
26
            if
                      (\ \ Teleop.DriveStick.getRawButton(\ 7\ )\ )\ \{\ ElevLift.SetHigh\ ();\ \}
27
                                                                      { ElevLift.SetLow (); }
            else
28
29
             // SEND SPEEDS TO SWERVE CLASS
30
            Swerve. UpdateRobotRelative(\ Xmag*Xsig\,,\ Ymag*Ysig\,,\ Tmag*Tsig\ );
31
32
33 }
```