**Java - WPILib: 01 - Robot**

**Lesson:**

There are two template types when creating a WPILib project: Iterative and Command. We will be using Command (If you want to learn how the Iterative model works there will be a separate course on it). The advantages of the Command based model is that it has built in easy separation of subsystems and controls for the robot, known as Commands.

The main class for a WPILib project is wthe Robot Class. Inside the Robot class is where all your calls to the rest of the program come from. You can think of it like the Main method, where your code will start to run from (This is not really correct but it is a good analogy to help you understand what is happening).

The Robot class is split into a few different sections: Robot, Autonomous, TeleOp, Disabled, and Test (Section is a term I am using to describe this, it is not an actual term). Each of these sections has two methods: Init and Periodic. The Init method is used to initialize any variables you may need for that section and runs once (there is more detail than this but that will be explained later as you add your own code). The Periodic method is where you put your code you want to run in that section. The Periodic method will run multiple times, usually once every 50ms, while that section is active (we will explain this a little later).

The difference between each section is when its methods are run. Robot is the first section to run. The very first part of your code that will run is RobotInit. After that RobotPeriodic will run approximately every 50ms until the program is told to switch to a different section. After that, it will run after the periodic of the selected section.

Disabled is where the code you want to run when the Robot is in Disabled Mode is placed. This can often be prints to the console about data from the robot or a default state of motors (usually off). Just like Robot, the DisabledInit runs once when Disabled Mode is started and then the DisabledPeriodic runs every 50ms.

Autonomous is where the code to run in Autonomous mode goes. Autonomous mode is when the Robot controls itself without the use of user input from a controller or some other means. In a Command Based robot you put the define what Commands to run in Autonomous inside AutonomousInit and then they are run by the scheduler in AutonomousPeriodic (we will go over the scheduler in more detail later on).

TeleOp is where the code for the TeleOp period is located. The TeleOp period is when the users control the robot through the use of a controller or other means. TeleOpInit stops the Autonomous command on the scheduler. TeleOpPeriodic runs the scheduler, so when commands are called (we will go over this in the Commands assignment) they are run.

Test is a mode designed for testing code. It runs when the Robot is in Test Mode and essentially acts like TeleOp.

**Notes:**

A few quick notes about Robot programming. First, never use loops in your code. NEVER. No matter how tempting it is, they will cause you more problems than they solve. The code all ready has loops built into the background that make the Periodic Methods run as they do. Don’t get in the way of the background code. It will always win in the end so do not try to interrupt it.

**Examples:**

**MyRobot – Robot.java**

/\*----------------------------------------------------------------------------\*/

/\* Copyright (c) 2017-2018 FIRST. All Rights Reserved.                        \*/

/\* Open Source Software - may be modified and shared by FRC teams. The code   \*/

/\* must be accompanied by the FIRST BSD license file in the root directory of \*/

/\* the project.                                                               \*/

/\*----------------------------------------------------------------------------\*/

package frc.robot;

import edu.wpi.first.wpilibj.TimedRobot;

import edu.wpi.first.wpilibj.command.Command;

import edu.wpi.first.wpilibj.command.Scheduler;

import edu.wpi.first.wpilibj.smartdashboard.SendableChooser;

import edu.wpi.first.wpilibj.smartdashboard.SmartDashboard;

import frc.robot.commands.ExampleCommand;

import frc.robot.subsystems.ExampleSubsystem;

/\*\*

 \* The VM is configured to automatically run this class, and to call the

 \* functions corresponding to each mode, as described in the TimedRobot

 \* documentation. If you change the name of this class or the package after

 \* creating this project, you must also update the build.gradle file in the

 \* project.

 \*/

public class Robot extends TimedRobot {

  public static ExampleSubsystem m\_subsystem = new ExampleSubsystem();

  public static OI m\_oi;

  Command m\_autonomousCommand;

  SendableChooser<Command> m\_chooser = new SendableChooser<>();

  /\*\*

   \* This function is run when the robot is first started up and should be

   \* used for any initialization code.

   \*/

  @Override

  public void robotInit() {

    m\_oi = new OI();

    m\_chooser.setDefaultOption("Default Auto", new ExampleCommand());

    // chooser.addOption("My Auto", new MyAutoCommand());

    SmartDashboard.putData("Auto mode", m\_chooser);

  }

  /\*\*

   \* This function is called every robot packet, no matter the mode. Use

   \* this for items like diagnostics that you want ran during disabled,

   \* autonomous, teleoperated and test.

   \*

   \* <p>This runs after the mode specific periodic functions, but before

   \* LiveWindow and SmartDashboard integrated updating.

   \*/

  @Override

  public void robotPeriodic() {

  }

  /\*\*

   \* This function is called once each time the robot enters Disabled mode.

   \* You can use it to reset any subsystem information you want to clear when

   \* the robot is disabled.

   \*/

  @Override

  public void disabledInit() {

  }

  @Override

  public void disabledPeriodic() {

    Scheduler.getInstance().run();

  }

  /\*\*

   \* This autonomous (along with the chooser code above) shows how to select

   \* between different autonomous modes using the dashboard. The sendable

   \* chooser code works with the Java SmartDashboard. If you prefer the

   \* LabVIEW Dashboard, remove all of the chooser code and uncomment the

   \* getString code to get the auto name from the text box below the Gyro

   \*

   \* <p>You can add additional auto modes by adding additional commands to the

   \* chooser code above (like the commented example) or additional comparisons

   \* to the switch structure below with additional strings & commands.

   \*/

  @Override

  public void autonomousInit() {

    m\_autonomousCommand = m\_chooser.getSelected();

    /\*

     \* String autoSelected = SmartDashboard.getString("Auto Selector",

     \* "Default"); switch(autoSelected) { case "My Auto": autonomousCommand

     \* = new MyAutoCommand(); break; case "Default Auto": default:

     \* autonomousCommand = new ExampleCommand(); break; }

     \*/

    // schedule the autonomous command (example)

    if (m\_autonomousCommand != null) {

      m\_autonomousCommand.start();

    }

  }

  /\*\*

   \* This function is called periodically during autonomous.

   \*/

  @Override

  public void autonomousPeriodic() {

    Scheduler.getInstance().run();

  }

  @Override

  public void teleopInit() {

    // This makes sure that the autonomous stops running when

    // teleop starts running. If you want the autonomous to

    // continue until interrupted by another command, remove

    // this line or comment it out.

    if (m\_autonomousCommand != null) {

      m\_autonomousCommand.cancel();

    }

  }

  /\*\*

   \* This function is called periodically during operator control.

   \*/

  @Override

  public void teleopPeriodic() {

    Scheduler.getInstance().run();

  }

  /\*\*

   \* This function is called periodically during test mode.

   \*/

  @Override

  public void testPeriodic() {

  }

}

**Exercises:**

1. Create a new Robot Project.

**Next Assignment Link:** <https://classroom.github.com/a/6qQxQqxj>