**The Robot Class**

**Prerequisites:**

* **Java Classes**
* **Java Methods**
* **Java Print Statement**

**States of the Robot**

In order to understand the Robot Class, you first need to understand the states of the robot.

When the robot is turned on, there are two possible states for it to be in. The robot can either be enabled or disabled. When the robot is enabled, the Robot Status Light, or RSL (Shown below), will be on and blinking. In this state, the robot is able to move its motors and pneumatics, making it dangerous to be around. Whenever someone enables the robot, they will first yell, “Enabling!” to warn everyone around them. When the robot is disabled, the RSL will be solid orange. The robot can still run code, however, it can’t move. It’s important to remember that enabling the robot and disabling the robot is very different from turning it on and off.



The Robot’s RSL

Within the enabled state of the robot, there are three more possible states: Teleoperated (Teleop), Autonomous (Auton), and Test.

’

The [Driverstation](https://docs.wpilib.org/en/stable/docs/software/driverstation/driver-station.html) is used to change the state of the robot.

When the robot is in teleoperated mode, it is under the control of the driver. This means that it will accept input from a human driver, who controls the robot using a handheld controller and possibly a button box.



When the robot is in autonomous mode, it is controlled by code. No input from the human driver will be accepted. This mode can be very dangerous, so before enabling, make sure that people around you are aware and that the proper safety precautions have been put into place.

Test mode is very similar to teleop. It can still be controlled by a human driver. The one main difference is that test mode is meant to be a place for programmers to put code to verify that all of the robot’s systems are functioning properly.

**What is the Robot Class?**

Now that we know all about the states of the robot, we can finally get into the good stuff. The robot class is where everything in the robot starts. Code is run here when the robot is turned on, and also when one of the states of the robot is enabled or disabled.

If you look at the code below, you’ll notice a few different types of methods. First we have our ‘Init’ methods, such as robotInit(), autonomousInit(), and teleopInit(). ‘Init’ is short for ‘Initialize’, and that’s exactly what these methods do. When the robot is first powered on, the method robotInit() is run. When we enable the robot in teleop mode, teleopInit() is run. These methods will contain any initialization we need to do at the beginning of each state.

Next we have our Periodic methods. Periodic methods are called repeatedly at an extremely fast rate, which is 50 times per second by default. The robotPeriodic() method is called repeatedly as long as the robot is powered on, similarly the teleopPeriodic() method is called repeatedly as long as the robot is enabled in teleop mode. An example of something we would put in one of these methods is code to constantly update the robot’s desired movement based upon the input from the joysticks.

| public class Robot extends TimedRobot {    @Override  public void robotInit() {    }   @Override  public void robotPeriodic() {  }    @Override  public void autonomousInit() {    }   @Override  public void autonomousPeriodic() {  CommandScheduler.getInstance().run();  }   @Override  public void teleopInit() {  }   @Override  public void teleopPeriodic() {  CommandScheduler.getInstance().run();  }    @Override  public void disabledInit() {}   @Override  public void disabledPeriodic() {}     @Override  public void testInit() {  }    @Override  public void testPeriodic() {  } } |
| --- |

**Practice**

Now that we know what the robot class is and what its methods do, let's put it into practice. Make sure to test your code by simulating after each step!

1. Clone [this](#) empty robot project onto your computer from github
2. Add some print statements so that we print to the console when the robot is turned on
3. Add a few more print statements so that we print the name of the mode we enter. Ex: When we enable teleop, “Teleop mode enabled” is printed to the console.
4. Try putting print statements into the periodic methods and see what happens.