



FRC Programming

with Matthew Neal (Mentor, Team 342)



Welcome!

Overview

- Introduction
- Algorithms and Abstractions
- Getting Competition Ready
- Programming Basics (Java)
- Robot Code
- Resources



bit.ly/FRC_PROGRAMMING_2024



FRC Programming

with Matthew Neal (Mentor, Team 342)



About Me

- **SC Department of Education**
 - K-12 Computer Science Coach
- **Fort Dorchester High School (Former)**
 - AP CS Principles
 - AP CSA
 - Algebra
 - Statistics
 - FIRST Robotics



mhneal@ed.sc.gov



FRC Programming

with Matthew Neal (Mentor, Team 342)



About Team 342

- Sub-team devoted primarily to programming.
- Fairly strong CS program in school, community, and district.
- Use Java
- Utilize GitHub for collaboration and work-flow.
 - Team GitHub: <https://github.com/frc-team-342>



Algorithms and Abstractions



Algorithms and Abstractions



List 5 steps explaining how you got ready to come here today.



Algorithms and Abstractions



Let's go one step deeper. Pick one of your 5 steps and split it into 5 smaller steps you need to complete the larger task. Write them on the same sheet of paper with an arrow to the step you broke up.



Algorithms and Abstractions



Let's go even deeper. Pick one of your 5 smaller steps and split it into 5 even smaller steps you need to complete the larger task. Write them on the same sheet of paper.



Algorithms and Abstractions



Share with a partner what you came up with.



Algorithms and Abstractions



In the previous activity, you used two important CS concepts - **algorithms** and **abstractions**.



The steps you listed to complete a task was an **algorithm**
Naming a group of instructions is called creating an **abstraction**



Getting Competition Ready

FIRST Robotics Competition



ZERO TO ROBOT

Introduction

Step 1: Building your Robot

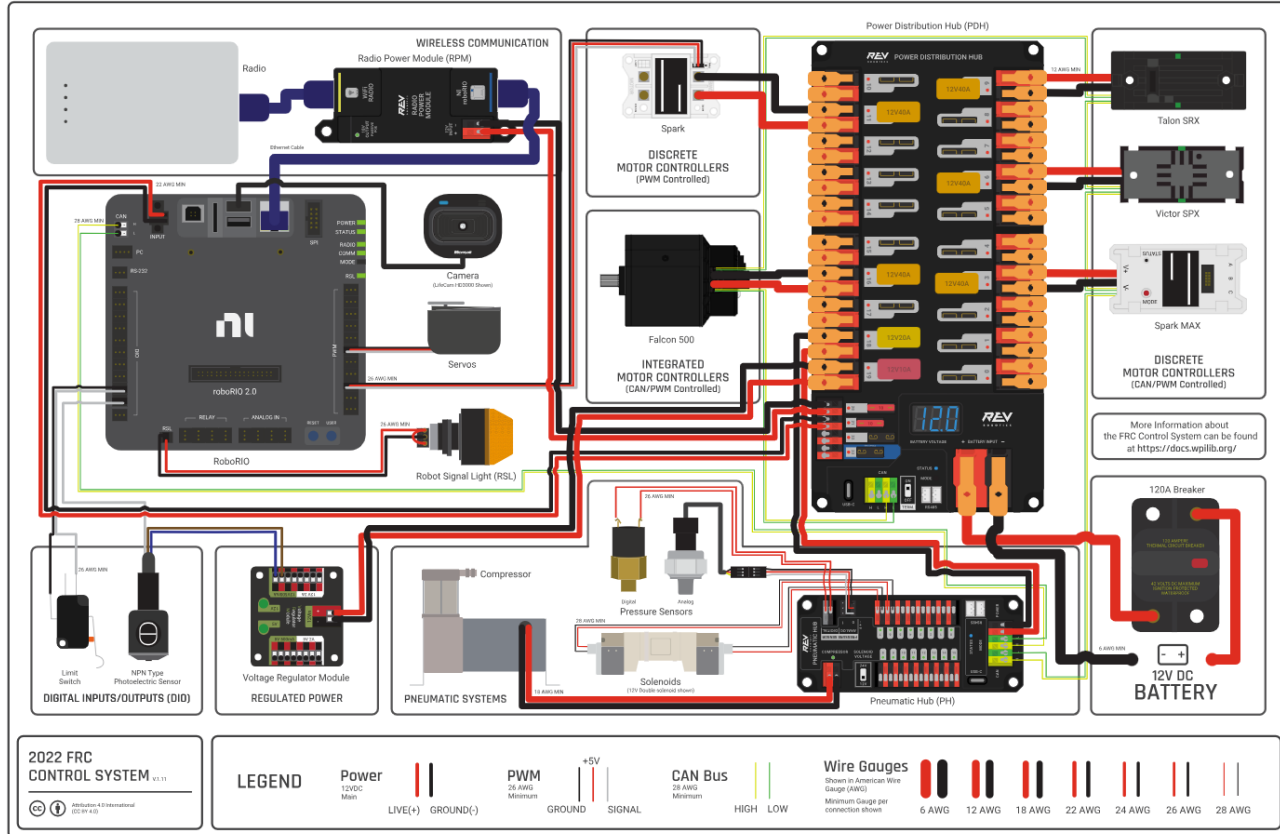
Step 2: Installing Software

Step 3: Preparing Your Robot

Step 4: Programming your Robot

<https://docs.wpilib.org/en/stable/docs/zero-to-robot/introduction.html>

Control System



Software Install



- [Install FRC Game Tools](#)
 - Driver Station (Requires Windows Device)
 - FRC roboRIO Imaging Tool and Images



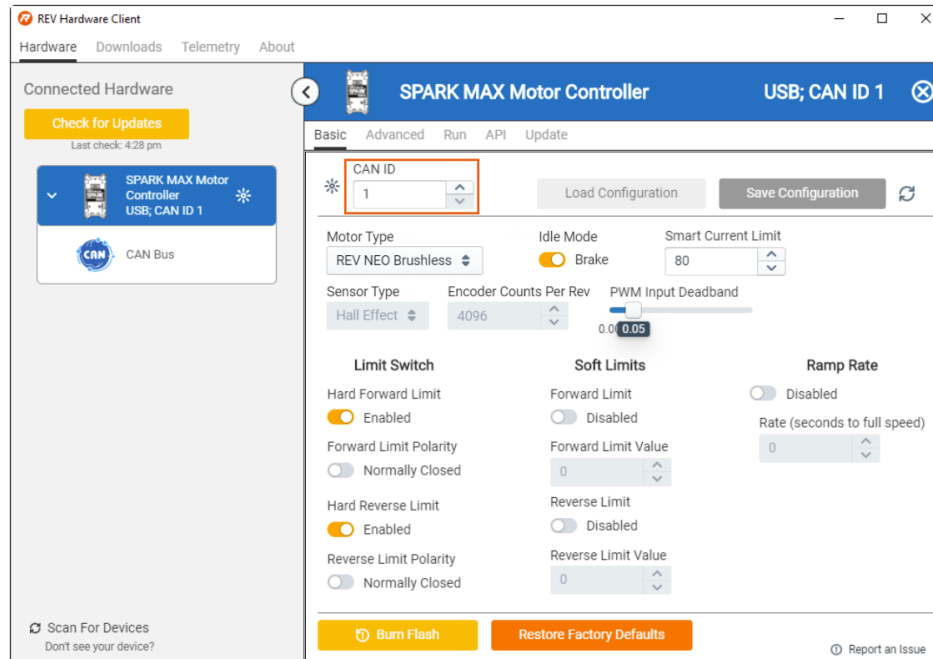
- [Install WPILib](#)
 - Allows us to code the robot



Software Install



- Install Hardware Clients ([Rev/Phoenix Tuner](#))
 - Allows us to set motor Ids and manually test hardware






Programming Basics



Basic Java Program




 Main.java

```
1 class Main {
2     public static void main(String[] args) {
3
4         System.out.println("Does anyone smell up dog?");
5
6     }
7 }
```

Basic Java Program



 Main.java

```
1 class Main {
2     public static void main(String[] args) {
3
4         //Calling the setup method
5         setup();
6         //Calling the punchline method
7         punchline();
8     }
9
10    public static void setup(){
11        System.out.println("What do you call a nun that sleep walks?");
12    }
13
14    public static void punchline(){
15        System.out.println("A Roman Catholic");
16    }
17 }
```


Basic Java Program



```
class Main {  
    public static void main(String[] args) {  
  
        Thing.doSomething();  
  
        Thing thing = new Thing();  
        thing.doSomethingElse();  
    }  
}
```


 Main.java


 Thing.java

```
1 public class Thing{  
2  
3     public static void doSomething(){  
4         System.out.println("What do you call a depressed robot?");  
5     }  
6  
7     public void doSomethingElse(){  
8         System.out.println("A sigh borg");  
9     }  
10 }
```

Basic Robot Program



 Main.java

 Robot.java

```
public final class Main {
    private Main() {}

    /**
     * Main initialization function. Do not perform any initialization here.
     *
     * <p>If you change your main robot class, change the parameter type.
     */
    public static void main(String... args) {
        RobotBase.startRobot(Robot::new);
    }
}
```

Basic Robot Program



```
public class Robot extends TimedRobot {  
    /**  
     * This function is run when the robot is first started up and should be used for any  
     * initialization code.  
     */  
    @Override  
    public void robotInit() {}  
  
    @Override  
    public void robotPeriodic() {}  
  
    @Override  
    public void autonomousInit() {}  
  
    @Override  
    public void autonomousPeriodic() {}  
  
    @Override  
    public void teleopInit() {}  
  
    @Override  
    public void teleopPeriodic() {}  
  
    @Override  
    public void disabledInit() {}  
  
    @Override  
    public void disabledPeriodic() {}  
  
    @Override  
    public void testInit() {}  
  
    @Override  
    public void testPeriodic() {}  
}
```



Variables and Data Types

CS
FOR
SC



Data Types



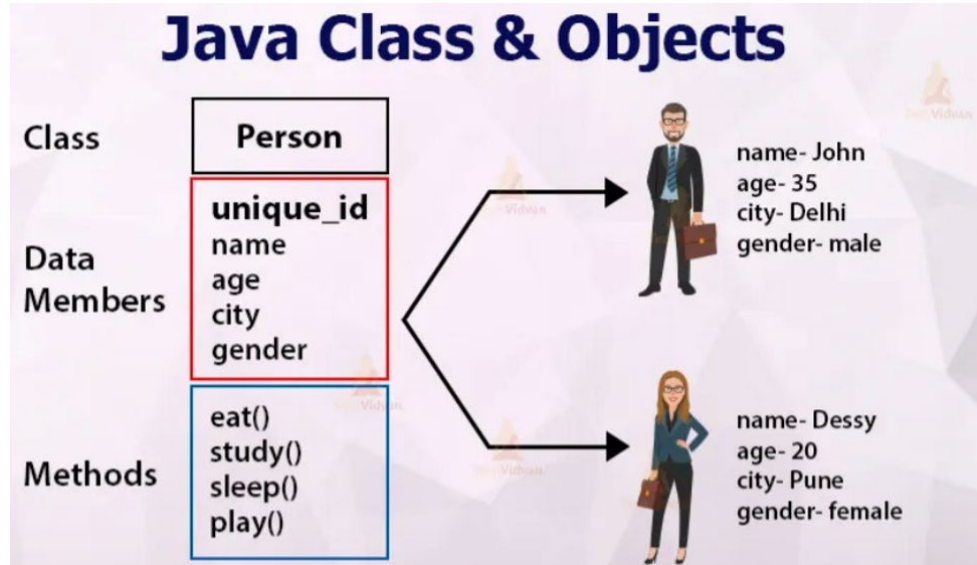
A **variable** is a named memory location that can store a value. Each variable has a **name** and a **data type**.

```
//Variables  
int year = 2024;  
String game = "Crescendo";  
boolean crushingIt = true;  
double hourSlept = 6.5;
```

Note: Good variable names help debugging!

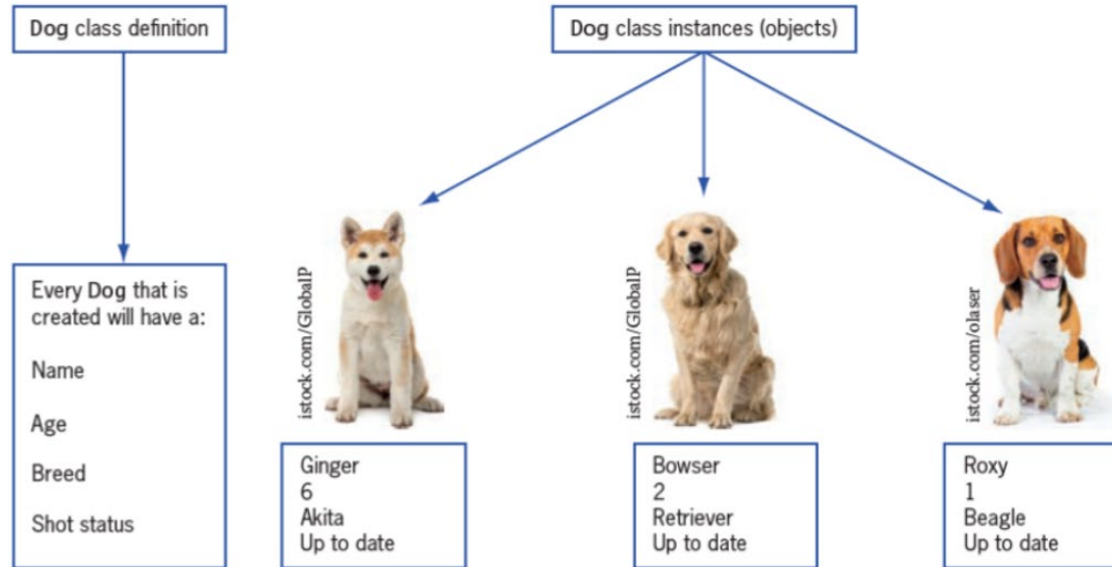
Classes and Objects

Suppose we want to represent something bigger in a variable, like a person.



Classes and Objects

A **class** is the formal implementation, or blueprint, for an **object**.



An **object** is a specific instance of a **class** with defined attributes.



Anatomy of a Class

Classes typically have three parts. (1) instance variables (2) constructor (3) methods;

```
public class Countdown{  
  
    private int start;  
  
    public Countdown(int startVal) {  
        start = startVal;  
    }  
  
    public void print() {  
        for(int i = start; i > 0; i--){  
            System.out.print(i + " ");  
        }  
    }  
  
} //End of Countdown
```

The defining attributes of the class

Always the same name as the class.

Initializes instance variables, if not initialized, the variables take on their default states.

Methods define what actions objects of this type can perform

Anatomy of a Class



To create a Countdown object, we invoke the constructor and the keyword `new`.

```
CountDown newYears = new Countdown(10)
```

This calls the constructor to give values to the start instance variable.

```
public Countdown(int startVal) {  
    start = startVal;  
}
```

Anatomy of a Class



In order to use a class, it's important to know how the constructor works and what methods you have access to.

```
1 class Main {
2     public static void main(String[] args) {
3
4         Countdown newYears = new Countdown(10);
5
6         newYears.print();
7
8     }
9 }
```

```
1 public class Countdown{
2
3     private int start;
4
5     public Countdown(int startVal){
6         start = startVal;
7     }
8
9     public void print(){
10        for(int i = start; i > 0; i--){
11            System.out.println(i);
12        }
13    }
14 }
```



Robot Code

CS
FOR
SC



Example Robot Program



```
public class Robot extends TimedRobot{
    //Creates two Motors in PWM ports 0 and 1
    private final PWMSparkMax m_leftMotor = new PWMSparkMax(0);
    private final PWMSparkMax m_rightMotor = new PWMSparkMax(1);

    //Creates an xbox controller
    private final XboxController xbox = new XboxController(0);

    public void robotInit(){
        //set motor inverted on right side
        m_rightMotor.setInverted(true);
    }

    public void teleopPeriodic(){
        double leftSpeed = xbox.getLeftY();
        double rightSpeed = xbox.getRightY();
        m_leftMotor.set(leftSpeed);
        m_rightMotor.set(rightSpeed);
    }
}
```



Imports (appear at top of file)

```
import edu.wpi.first.wpilibj.TimedRobot;  
import edu.wpi.first.wpilibj.XboxController;  
import edu.wpi.first.wpilibj.motorcontrol.PWMSparkMax;
```

<https://docs.wpilib.org/en/stable/docs/software/vscode-overview/3rd-party-libraries.html>

Vendor Libraries



To use commands that exist outside of wpilibj, you will need to install the vendor libraries.

```
> WPILib m
WPILib: Manage Vendor Libraries recently used
WPILib: Change Run Commands in Online Mode Settings
Select an option
Manage current libraries
Check for updates (offline)
Check for updates (online)
Install new libraries (offline)
Install new library (online)
```

You can view common vendor libraries on the documentation website.



Example Robot Program 2



```
public class Robot extends TimedRobot{
    //Creates two Can Motors with IDs 0 and 1
    private final int leftID = 0;
    private final int rightID = 1;
    private CANSparkMax m_leftMotor = new CanSparkMax(leftID, MotorType.kBrushless));
    private CANSparkMax m_rightMotor = new = new CanSparkMax(rightID, MotorType.kBrushless));

    //Creates an xbox controller
    private final XboxController xbox = new XboxController(0);

    public void robotInit(){
        //set motor inverted on right side
        m_rightMotor.setInverted(true);
    }

    public void teleopPeriodic(){
        if(xbox.getAButtonPressed()){
            m_rightMotor.set(0.5);
            m_leftMotor.set(-0.5);
        }
    }
}
```



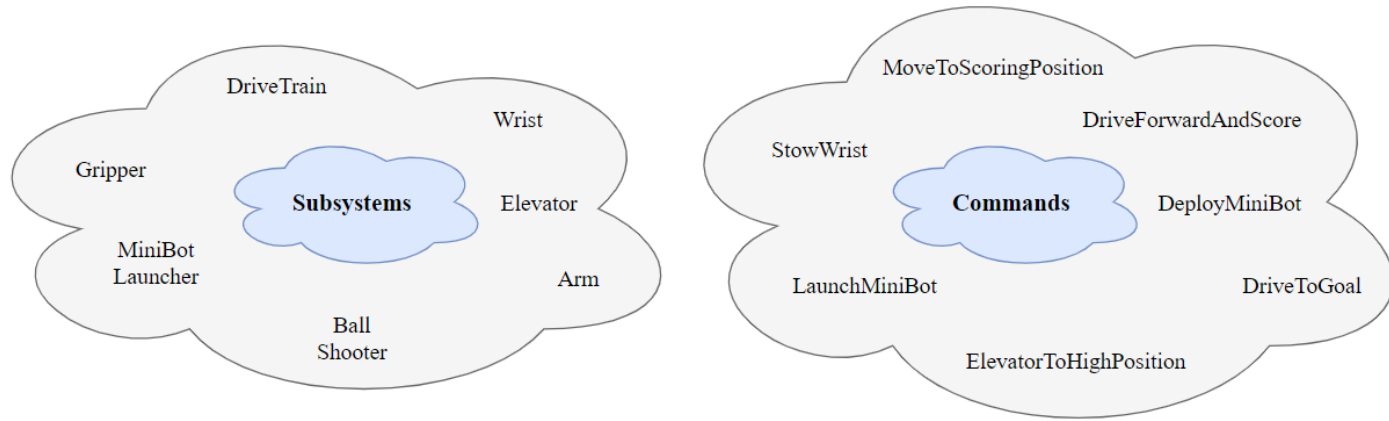
Imports (appear at top of file)

```
import edu.wpi.first.wpilibj.Joystick;  
import edu.wpi.first.wpilibj.TimedRobot;  
import com.revrobotics.CANSparkMax;  
import com.revrobotics.CANSparkMaxLowLevel.MotorType;
```





Command-Based Programming



Command-Based Programming



The command-based pattern is based around two core abstractions:
commands, and **subsystems**.

Commands represent actions the robot can take. Commands run when scheduled, until they are interrupted or their end condition is met.

Subsystems represent independently-controlled collections of robot hardware (such as motor controllers, sensors, pneumatic actuators, etc.) that operate together. Subsystems back the resource-management system of command-based: only one command can use a given subsystem at the same time.

Command-based programming does require significant advantages in writing manageable code, but does come with a learning curve for teams.

Code



commands



subsystems



Constants.java



Limelight.java



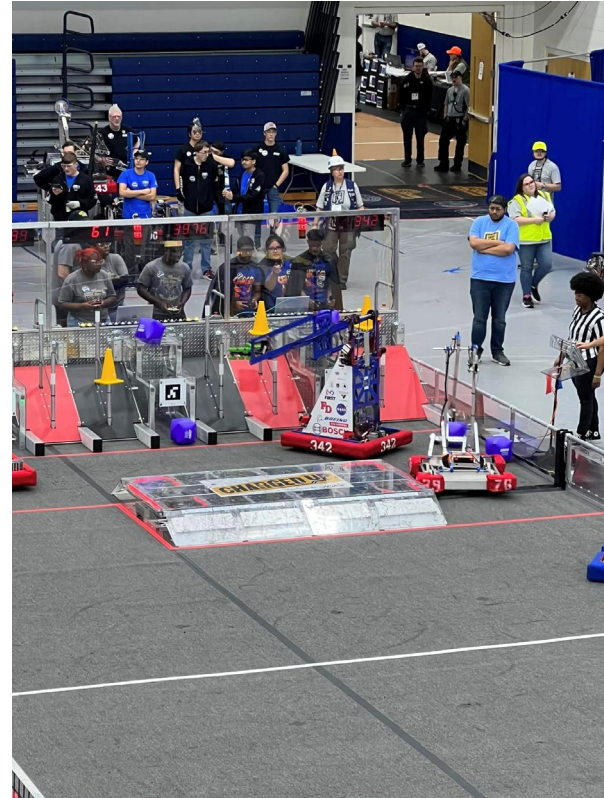
Main.java



Robot.java





RobotContainer.java





Subsystems



 AddressableLEDSubsystem.java

 DriveSystem.java

 GripperSystem.java

 LiftSystem.java

 Testable.java



Subsystems (Example Code Part 1)



```
public DriveSystem() {  
    // motors  
    frontLeft = new CANSparkMax(FRONT_LEFT_MOTOR, MotorType.kBrushless);  
    frontRight = new CANSparkMax(FRONT_RIGHT_MOTOR, MotorType.kBrushless);  
    backLeft = new CANSparkMax(BACK_LEFT_MOTOR, MotorType.kBrushless);  
    backRight = new CANSparkMax(BACK_RIGHT_MOTOR, MotorType.kBrushless);  
  
    frontLeft.setIdleMode(IdleMode.kBrake);  
    frontRight.setIdleMode(IdleMode.kBrake);  
    backLeft.setIdleMode(IdleMode.kBrake);  
    backRight.setIdleMode(IdleMode.kBrake);  
}
```

Subsystems (Example Code Part 2)



```
public void drivePercent(double leftSpeed, double rightSpeed) {  
    // left side  
    double leftVelocity = leftSpeed * currentMode.speedMultiplier;  
    leftController.setReference(leftVelocity, ControlType.kDutyCycle);  
  
    // right side  
    double rightVelocity = rightSpeed * currentMode.speedMultiplier;  
    rightController.setReference(rightVelocity, ControlType.kDutyCycle);  
}
```

Subsystems (Example Code Part 3)



```
public CommandBase driveWithJoystick(Joystick joyLeft, Joystick joyRight) {
    return runEnd(
        // Runs drive repeatedly until command is stopped
        () -> {
            double left = MathUtil.applyDeadband(joyLeft.getY(), 0.15);
            double right = MathUtil.applyDeadband(joyRight.getY(), 0.15);

            drivePercent(left, right);
        },
        // Stops robot after command is stopped
        () -> {
            drivePercent(0, 0);
        }
    );
}
```



FRC Programming

with Matthew Neal (Mentor, Team 342)



Resources

- [WPILib Competition Control System Documentation](#)
- [Chief Delphi \(Forum\)](#)
- [Swerve Drive Coding Video](#)



mhneal@ed.sc.gov



FRC Programming

with Matthew Neal (Mentor, Team 342)



Questions

- **SC Department of Education**
 - K-12 Computer Science Coach



mhneal@ed.sc.gov



FRC Programming

with Matthew Neal (Mentor, Team 342)



Welcome!

Overview

- Introduction
- Algorithms and Abstractions
- Getting Competition Ready
- Programming Basics (Java)
- Robot Code
- Resources



bit.ly/FRC_PROGRAMMING_2024