Java Tutorials

Timeline/Assignments

Class	Vocabulary (classes overview, Java terms, most used FTC Java classes)
Assignment	Review the terms you are unsure of as every meeting we will have an oral quiz
Class	Video Tutorial: Hardware class, lesson time, questions
Assignment	Write your own Hardware class: 1. Use at least 3 “robot parts” classes 2. Use 1 “sensor” class 3. Use 1 mode, 1 position, and 1 power
Class	Video Tutorial: Teleop class, lesson time, questions
Assignment	Write your own Teleop class: 1. Use at least 3 if statements 2. Use 2 variables 3. Use the Telemetry class

	4. Use the Gamepad class
Class	Video Tutorial: Autonomous class, lesson time, questions
Assignment	Write your own Autonomous class: 1. Have the robot drive forward 2. Have the robot turn left 3. Have the robot go forward 4. Have the robot lift the arm

Intro

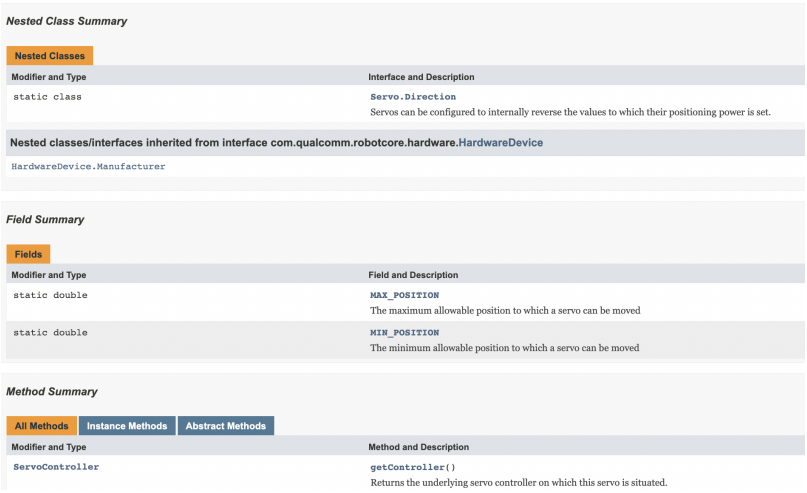
How to Read a Class

Interface - tells you the name of the class

com.qualcomm.robotcore.hardware

Interface Servo

Summaries - “Nested Classes” is not important, “Fields” tells you the attributes, and “All Methods” tells you every method in the class, including setters and getters (don’t worry about the different types of methods)



Nested Class Summary

Nested Classes

Modifier and Type	Interface and Description
static class	Servo.Direction Servos can be configured to internally reverse the values to which their positioning power is set.

Nested classes/interfaces inherited from interface com.qualcomm.robotcore.hardware.HardwareDevice

HardwareDevice.Manufacturer

Field Summary

Fields

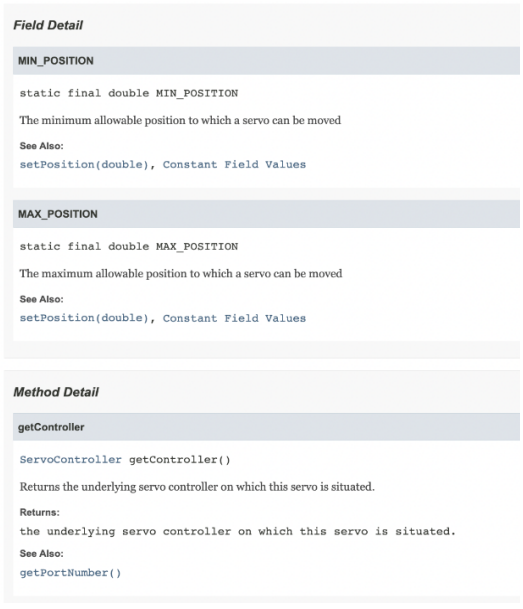
Modifier and Type	Field and Description
static double	MAX_POSITION The maximum allowable position to which a servo can be moved
static double	MIN_POSITION The minimum allowable position to which a servo can be moved

Method Summary

All Methods | Instance Methods | Abstract Methods

Modifier and Type	Method and Description
ServoController	getController() Returns the underlying servo controller on which this servo is situated.

Details - “Field Detail” explains the attributes from the summary and “Method Detail” explains the methods from the summary. Details are super useful and the main thing to refer to when reading a class dictionary.



Field Detail

MIN_POSITION

static final double MIN_POSITION

The minimum allowable position to which a servo can be moved

See Also:
setPosition(double), Constant Field Values

MAX_POSITION

static final double MAX_POSITION

The maximum allowable position to which a servo can be moved

See Also:
setPosition(double), Constant Field Values

Method Detail

getController

ServoController getController()

Returns the underlying servo controller on which this servo is situated.

Returns:
the underlying servo controller on which this servo is situated.

See Also:
getPortNumber()

Most Common (to us) FTC Java Classes

All classes:
https://ftctechnh.github.io/ftc_app/doc/javadoc/index.html?com

Robot Parts:

Servo:
https://ftctechnh.github.io/ftc_app/doc/javadoc/com/qualcomm/robotcore/hardware/Servo.html

- Parent Servo class

CRServo:
https://ftctechnh.github.io/ftc_app/doc/javadoc/com/qualcomm/robotcore/hardware/CRServo.html

- Inherits some methods/attributes of Servo
- Difference: Continuous rotation rather than to a specific position

DcMotor:
https://ftctechnh.github.io/ftc_app/doc/javadoc/com/qualcomm/robotcore/hardware/DcMotor.html

- Parent DcMotor class

DcMotorEx:
https://ftctechnh.github.io/ftc_app/doc/javadoc/com/qualcomm/robotcore/hardware/DcMotorEx.html

- Inherits some methods/attributes of DcMotor
- Difference: More precise because uses encoders

Sensors:

- Some sensors uses classes specific to the company (i.e. REV has its own sensor class)
- Sensors always have pre written templates so you can test them when programming

DistanceSensor:
https://ftctechnh.github.io/ftc_app/doc/javadoc/com/qualcomm/robotcore/hardware/DistanceSensor.html

ColorSensor:
https://ftctechnh.github.io/ftc_app/doc/javadoc/com/qualcomm/robotcore/hardware/ColorSensor.html

LightSensor:
https://ftctechnh.github.io/ftc_app/doc/javadoc/com/qualcomm/robotcore/hardware/LightSensor.html

Always Used:

Gamepad:
https://ftctechnh.github.io/ftc_app/doc/javadoc/com/qualcomm/robotcore/hardware/Gamepad.html

- This is super important as the Gamepad class has all the methods that actually program our controller to control the robot

HardwareMap:
https://ftctechnh.github.io/ftc_app/doc/javadoc/com/qualcomm/robotcore/hardware/HardwareMap.html

- This class will be prewritten so don't worry about it, just acknowledge it

Vocabulary List

Object-oriented programming - a programming paradigm based on the concept of "objects", which can contain data and code.

Java - a high-level, class-based, object-oriented programming language.

Class - a blueprint for creating objects (a particular data structure), providing initial values for state (member variables or attributes), and implementations of behavior (member functions or methods). A blueprint that defines the nature of a future object.

Java object - a member (also called an instance) of a Java class. Each object has an identity, a behavior and a state. The state of an object is stored in fields (variables), while methods (functions) display the object's behavior.

Attributes - a changeable property or characteristic of some component of a program that can be set to different values.

Method - a set of instructions that can be called for execution using the method name. A Java method can take in data or parameters and return a value - both parameters and return values are optional.

Constructor - a special method that is used to initialize objects. The constructor is called when an object of a class is created.

Getters and Setters - methods used to protect your data and make your code more secure. Getter returns the value (accessors), it returns the value of data type int, String, double, float, etc. Setter sets or updates the value (mutators). It sets the value for any variable which is used in the programs of a class.

Example Java Program

Program

ExampleClass.java > ...

```java
// Class Declaration
public class ExampleClass {
    // Variable
    String example_string_variable;

    // Constructor Declaration of Class
    public ExampleClass(String example_string_variable) {
        this.example_string_variable = example_string_variable;
    }

    // Getter
    public String getExampleString() {
        return example_string_variable;
    }

    // Setter
    // Uses "void" because it is a keyword used to specify that a
    // Method doesn't return anything
    public void setExampleString(String new_string) {
        this.example_string_variable = new_string;
    }

    // Method
    public String ExampleMethod() {
        return ("Hi I am a method and here is what my example string is: " + this.getExampleString());
    }

    // Main method
    // Always need one to run the program
    // The keyword static means that the method belongs to this class itself
    Run | Debug
    public static void main(String[] args) {
        ExampleClass obj = new ExampleClass("Hi lol");
        System.out.println(obj.getExampleString());
        obj.setExampleString("PYTHON IS BETTER.");
        System.out.println(obj.getExampleString());
        System.out.println(obj.ExampleMethod());
    }
}
```

Output

```
Hi lol
PYTHON IS BETTER.
Hi I am a method and here is what my example string is: PYTHON IS BETTER.
```

Classes

Links

Set up to code video:
https://www.youtube.com/watch?v=08x4qU9n0BM&list=PLPvmeKJRojE0qyfm5RnsyD7P4gqKzuwyr

FTC Team Code & templates folders:
https://github.com/ftctechnh/ftc_app

Android Studio:
https://developer.android.com/studio

Set Up to Code

1. Download Android Studio if you don't already have it
2. Start a new project and navigate to the FTC Team Code and template folder you should have dragged onto your desktop from the links section
3. Open this folder and follow the instructions!

How to Navigate

- Templates: found under FtcRobotController/java/external.samples
- Your Code: found under TeamCode/java
- Copy and paste the template into your code and make the edits

Hardware Classes

Template you should follow: HardwarePushbot

1. Change file name and match class & constructor name to said name
2. Declare your OpMode Members
    - "public DcMotor leftDrive = null;"
    - public [CLASS NAME] [name] = null;
3. Initialize OpMode Members
    - "leftDrive = hwMap.get(DcMotor.class, "left_drive");"
    - [name] = hwMap.get([CLASS NAME].class, "[NAME ON PHONE]");
4. Set Modes/Powers/Positions etc. (depends on what you are coding, I'm using a DcMotor class here as an example)
    - "leftDrive.setPower(0);"
    - [name].setPower(0);
    - "leftDrive.setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);"
    - [name].setMode([CLASS NAME].RunMode.RUN_WITHOUT_ENCODER);
5. Delete what you don't need and adjust the comments

TeleOp Classes

Template you should follow: it depends on what you are coding and you should use a combo of different templates for this

1. Delete the “@Disabled”
2. Ensure that your “@TeleOp” code matches the name of your hardware class (the “group=” should be followed by your hardware class name
3. Code only within “public void runOpMode()”
4. Declare variables
    - “double [name] = [value];”
5. Set your powers/positions/modes etc. to what they should be just like in Hardware class if it is different
6. Now code within the while loop
7. Refer to the classes above to code (from the previous lesson) as well as templates to code what you wish
8. Use the telemetry class to ensure that the robot is reading the right values

Example code: *if the button is pressed, set the arm to the lowest position and have the motor go to that position at a low power*

```
if (gamepad1.dpad_left) {
    robot.arm.setTargetPosition(220);
    robot.arm.setMode(DcMotor.RunMode.RUN_TO_POSITION);
    robot.arm.setPower(0.2);
}
```

Autonomous Classes

Template you should follow: it depends on what you are coding and you should use a combo of different templates for this

1. Delete the “@Disabled”
2. Ensure that your “@Autonomous” code matches the name of your hardware class (the “group=” should be followed by your hardware class name
3. Code only within “public void runOpMode()”
4. Coded the same way as the Teleop classes (refer back) but instead NO GAMEPAD class is used
5. You can use sensors and/or hard coded code
6. Use “sleep(#);” for the robot to do the code above it for as long as # (in milliseconds) is

Example Code: *drives the robot forward at a speed of 0.4 for 350 milliseconds*

```
// Drive forward
robot.frontLeft.setPower(0.4);
robot.frontRight.setPower(0.4);
robot.backLeft.setPower(0.4);
robot.backRight.setPower(0.4);
sleep( milliseconds: 350);
```