FTC Relic
John Marchand – jhmarchand@gmail.com
Team 4381 – Twisted Devils
https://github.com/jhmarchand/java4381

Week 1

Updating or Configuring Your Phones

Your Driver Station phone (9087-DS) and Robot Controller phone (9087-RC) need to be updated to have the latest FTC apps. They likely have version 2.6 on them. You need to update them to have version 3.4. I think we just need to get the google account password, and should be able to update them in the google play store.

If your phones have not been set up, or you want to check, or you need more detailed instructions, they are available here.

https://github.com/ftctechnh/ftc_app/wiki/Configuring-Your-Android-Devices

Configuring Your Hardware

Before we begin programming, it's best to configure your robot with some of the motors and servos you will be using on your real robot. Please note that to do this configuration, the motors and servos only need to be connected to the REV expansion hub, and don't really need to be mounted properly. Please follow the instructions here.

https://github.com/ftctechnh/ftc_app/wiki/Configuring-Your-Hardware

Important Links

FTC Programming Home
https://www.firstinspires.org/resource-library/ftc/technology-information-and-resources

FTC Control System Wiki (I recommend reading all of it)
https://github.com/ftctechnh/ftc_app/wiki

FTC Control System Forums (Mainly Programing Questions and Answers here. Also sub forums.)
https://ftcforum.usfirst.org/forum/ftc-technology

FTC SDK Documentation
http://ftctechnh.github.io/ftc_app/doc/javadoc/index.html

Choosing a Programming Tool

https://github.com/ftctechnh/ftc_app/wiki/Choosing-a-Programming-Tool

The FTC Blocks Programming Tool - A visual, programming tool that lets programmers use a web browser to create, edit and save their op modes. This tool is recommended for novice programmers and for users who prefer to design their op modes visually, using a drag-and-drop interface.

The FTC OnBot Java Programming Tool - A text-based programming tool that lets programmers use a web browser to create, edit and save their Java op modes. This tool is recommended for programmers who have basic to advanced Java skills and who would like to write text-based op modes.

Android Studio - An advanced integrated development environment for creating Android apps. This tool is the same tool that professional Android app developers use. Android Studio is only recommended for advanced users who have extensive Java programming experience.

I recommend using the OnBot Java.

- Testing changes to your code is quick and easy
- Builds on your Java learning this summer.
- FTC seems to be the recommending this over Android Studio for Java.
- No install needed on your laptop, you only need a javascript enabled browser.

NOTE - Java is not related to javascript at all.

Optional Android Studio Setup

Even though I recommend OnBot Java, having Android Studio is nice for several reasons.

- You can write code without the phones
- It is easy to browse the sample projects
- It is a good place to save your OnBot Java files. These files can be uploaded and downloaded from the phone.
- It is likely easier to edit and examine code, including the FTC libraries.

1. Download the FTC Robot Controller Project. Click the green button on the right and choose Download zip. https://github.com/ftctechnh/ftc_app
2. Upzip the project and remember the location.
3. Download Android Studio (You can use Eclipse if you want, but for this Android Studio is cleaner) https://developer.android.com/studio/index.html
4. Install and Open Android Studio
5. Choose Import project (Eclipse ADT, Gradle, etc.) and choose the folder you unzipped to from step 2.

OpModes

OpModes define what the robot does when it is running. There are 2 types of OpModes Autonomous and TeleOp. Autonomous OpModes are when your code drives the robot. TeleOp OpModes are for when the gamepads are controlling the robot.

For now, let's focus on TeleOp.

Each OpMode you write is a Java Class. Each OpMode has to extend (or inherit) from a class called OpMode.

Iterative OpMode - Preferred

Steps to write your own OpMode. Don't worry, you can start from samples!
https://github.com/ftctechnh/ftc_app/tree/master/FtcRobotController/src/main/java/org/firstinspires/ftc/robotcontroller/external/samples

Create a class that extends OpMode. It has some funny stuff on top, the name is what will get listed on your phone. Also, the @TeleOp says its teleOp as opposed to autonomous.

```
@TeleOp(name="Basic: POV Iterative", group="Iterative Opmode")
public class BasicPOVIterativeTeleOpMode extends OpMode
```

Add members for your controls (Motors and servos)

```
private DcMotor leftDrive = null;
private DcMotor rightDrive = null;
```

Override the init function. You won't really write a Constructor for your class, but can do most constructor like functionality in here. Including making sure your member variables point to the correct motors.

```
public void init() {
   // Set up Motors
   leftDrive  = hardwareMap.get(DcMotor.class, "left_drive");
   rightDrive = hardwareMap.get(DcMotor.class, "right_drive");

   // For Motors that are mounted backwards, we have to set the direction to be reverse
   leftDrive.setDirection(DcMotor.Direction.FORWARD);
   rightDrive.setDirection(DcMotor.Direction.REVERSE);
}
```

Override the loop function. This is a function that will get called every 20ms. Here you want to get information from the gamepad and give the motors and servos the proper power. Do not use any wait, delay or loops in this function. Get your information from the gamepad, do any calculations needed, and give information to the controls.

```java
public void loop() {

    // calculate Power for motors
    double drive = -gamepad1.left_stick_y;
    double turn  =  gamepad1.right_stick_x;
    double leftPower    = Range.clip(drive + turn, -1.0, 1.0) ;
    double rightPower   = Range.clip(drive - turn, -1.0, 1.0) ;

    // Send calculated power to wheels
    leftDrive.setPower(leftPower);
    rightDrive.setPower(rightPower);

    // Show the elapsed game time and wheel power.
    telemetry.addData("Status", "Run Time: " + runtime.toString());
    telemetry.addData("Motors", "left (%.2f), right (%.2f)", leftPower, rightPower);
}
```

GamePads

Member variables gamepad1 and gamepad2 are defined in the superclass OpMode, but you can use them in your subclass. Here are most of the values available. The sticks give values from -1 to 1, so half way forward would be .5. As luck would have it, thats the values most motors want.

```java
double gamepadDouble;
boolean gamepadBoolean;

gamepadDouble = gamepad1.left_stick_x;
gamepadDouble = gamepad1.left_stick_y;

gamepadDouble = gamepad1.right_stick_x;
gamepadDouble = gamepad1.right_stick_y;

gamepadDouble = gamepad1.right_trigger;
gamepadDouble = gamepad1.left_trigger;

gamepadBoolean = gamepad1.a;
gamepadBoolean = gamepad1.b;
gamepadBoolean = gamepad1.x;
gamepadBoolean = gamepad1.y;

gamepadBoolean = gamepad1.dpad_down;
gamepadBoolean = gamepad1.dpad_up;
gamepadBoolean = gamepad1.dpad_left;
gamepadBoolean = gamepad1.dpad_right;

gamepadBoolean = gamepad1.left_bumper;
gamepadBoolean = gamepad1.right_bumper;

gamepadBoolean = gamepad1.left_stick_button;
gamepadBoolean = gamepad1.right_stick_button;
```

Output

System.out.print() doesn't work on the phones. However, you can send information from the robot controller to the driver station with the telemetry member variable. Again, this is defined in the OpMode class.

```
   telemetry.addData("Status", "Run Time: " + runtime.toString());
   telemetry.addData("Motors", "left (%.2f), right (%.2f)", leftPower, rightPower);
```

LinerOpMode

This is different than a regular OpMode. Instead of having an init and a loop function automatically get called every 20ms, you have a function that gets called once. You can look at the samples on how to write this, but in general, it allows for more errors and should only be used if needed for some reason.

```
public class BasicOpMode_Linear extends LinearOpMode
```

OnBot Java

FTC WIki Tutorial (if your phone is setup, start with step 3 part 2)
https://github.com/ftctechnh/ftc_app/wiki/OnBot-Java-Tutorial

GearBots Video Tutorial
https://www.youtube.com/watch?v=tRDSj2AZJuI

Things to Consider

As your team talks about the robot, how are you going to build a good control system. Even simple things like POV (point of view) drive (left stick speed, right stick turn) versus tank drive (left stick - left motors, right stick - right motors). Configuring a controllable arm will be much more difficult. Try to visualize it as you are building the robot.

What do you want to do in Autonomous. Having a good idea of your goal will help you write your first Autonomous OpMode.

Look at the sample codes.

Read the forums.