FTC SDK Sample Code Analysis

This document provides an overview of the sample programs found in `FtcRobotController/src/main/java/org/firstinspires/ftc/robotcontroller/external/samples`. These samples are excellent resources for learning how to program your robot.

Categories

The samples are generally categorized by their prefix:

- **Basic**: Simple templates to get started.
- **Concept**: Demonstrates specific features or concepts (e.g., AprilTags, Gamepad, Telemetry).
- **Robot**: Complete examples of robot behaviors (e.g., Auto Drive, TeleOp).
- **Sensor**: Examples of how to use specific sensors (e.g., Color, IMU, Touch).
- **Utility**: Helper classes or tools.

Detailed File Explanations

1. Basic OpModes (Templates)

- `BasicOmniOpMode_Linear.java`: TeleOp for omni-directional robots (Mecanum/X-Drive).
- `BasicOpMode_Iterative.java`: Template for Iterative OpMode (loop-based).
- `BasicOpMode_Linear.java`: Template for Linear OpMode (sequential).

2. Concept Samples (Features & Tutorials)

- `ConceptAprilTag.java`: Basic AprilTag detection.
- `ConceptAprilTagEasy.java`: Simplified AprilTag detection.
- `ConceptAprilTagLocalization.java`: AprilTag for field positioning.
- `ConceptAprilTagMultiPortal.java`: Using multiple cameras for AprilTags.
- `ConceptAprilTagOptimizeExposure.java`: Tuning camera exposure for tags.
- `ConceptAprilTagSwitchableCameras.java`: Switching between cameras.
- `ConceptBlackboard.java`: Using the Blackboard feature.
- `ConceptExploringIMUOrientation.java`: Understanding IMU axes.
- `ConceptGamepadEdgeDetection.java`: Detecting button presses (rising edge).
- `ConceptGamepadRumble.java`: Making the controller vibrate.
- `ConceptGamepadTouchpad.java`: Using the PS4/PS5 touchpad.
- `ConceptLEDStick.java`: Controlling LED strips.

- `ConceptMotorBulkRead.java` : Optimizing code speed by reading all motors at once.
- `ConceptNullOp.java` : An empty OpMode.
- `ConceptRampMotorSpeed.java` : Smoothly accelerating motors.
- `ConceptRevLED.java` : Controlling REV Blinkin LED modules.
- `ConceptRevSPARKMini.java` : Using the REV SPARK Mini controller.
- `ConceptScanServo.java` : Moving a servo back and forth.
- `ConceptSoundsASJava.java` : Playing sounds (Android Studio).
- `ConceptSoundsOnBotJava.java` : Playing sounds (OnBot Java).
- `ConceptSoundsSKYSTONE.java` : Playing legacy Skystone sounds.
- `ConceptTelemetry.java` : Advanced telemetry usage.
- `ConceptVisionColorLocator_Circle.java` : Finding circular colored objects.
- `ConceptVisionColorLocator_Rectangle.java` : Finding rectangular colored objects.
- `ConceptVisionColorSensor.java` : Advanced color sensor usage.

3. Robot Samples (Complete Behaviors)

- `RobotAutoDriveByEncoder_Linear.java` : Auto drive using encoders.
- `RobotAutoDriveByGyro_Linear.java` : Auto drive using Gyro/IMU.
- `RobotAutoDriveByTime_Linear.java` : Auto drive using time (simple).
- `RobotAutoDriveToAprilTagOmni.java` : Auto drive to tag (Omni drive).
- `RobotAutoDriveToAprilTagTank.java` : Auto drive to tag (Tank drive).
- `RobotAutoDriveToLine_Linear.java` : Auto drive until a line is seen.
- `RobotTeleopMecanumFieldRelativeDrive.java` : Field-centric TeleOp.
- `RobotTeleopPOV_Linear.java` : Simple POV TeleOp.
- `RobotTeleopTank_Iterative.java` : Tank drive TeleOp.

4. Sensor Samples (Hardware Drivers)

- `SensorAndyMarkIMUNonOrthogonal.java` : AndyMark IMU (tilted).
- `SensorAndyMarkIMUOrthogonal.java` : AndyMark IMU (flat).
- `SensorAndyMarkTOF.java` : Time-of-Flight distance sensor.
- `SensorBNO055IMU.java` : Older BNO055 IMU.
- `SensorBNO055IMUCalibration.java` : Calibrating BNO055.
- `SensorColor.java` : Standard Color Sensor.
- `SensorDigitalTouch.java` : Digital Touch Sensor.
- `SensorGoBildaPinpoint.java` : GoBilda Pinpoint Odometry Computer.
- `SensorHuskyLens.java` : AI Vision Sensor (HuskyLens).
- `SensorIMUNonOrthogonal.java` : Generic IMU (tilted).
- `SensorIMUOrthogonal.java` : Generic IMU (flat).
- `SensorKLNavxMicro.java` : Kauai Labs NavX Micro.
- `SensorLimelight3A.java` : Limelight 3A Vision Camera.

- `SensorMRColor.java`: Modern Robotics Color Sensor.
- `SensorMRGyro.java`: Modern Robotics Gyro.
- `SensorMROpticalDistance.java`: Modern Robotics ODS.
- `SensorMRRangeSensor.java`: Modern Robotics Range Sensor.
- `SensorOctoQuad.java`: OctoQuad Encoder Interface.
- `SensorOctoQuadAdv.java`: Advanced OctoQuad usage.
- `SensorOctoQuadLocalization.java`: OctoQuad for localization.
- `SensorREV2mDistance.java`: REV 2m Distance Sensor.
- `SensorSparkFunOTOS.java`: SparkFun Optical Tracking Odometry Sensor.
- `SensorTouch.java`: Standard Touch Sensor.

5. Utility & Hardware

- `SampleRevBlinkinLedDriver.java`: Driver for REV Blinkin.
- `UtilityCameraFrameCapture.java`: Capturing camera frames.
- `UtilityOctoQuadConfigMenu.java`: Menu for configuring OctoQuad.

How to Use These Samples

1. **Don't edit them directly.**
2. **Copy and Paste:** Open the file you want to use in Android Studio. Select all, Copy.
3. **Create New Class:** Go to your `TeamCode` folder, right-click -> New -> Java Class.
4. **Paste:** Paste the code into your new file.
5. **Rename:** Change the class name to match your new file name.
6. **Enable:** Find the `@Disabled` line near the top and remove it (or comment it out `// @Disabled`).
7. **Configure:** Update the hardware names (e.g., "left_drive") to match your Robot Configuration.