Blank Red Program

```java
package org.firstinspires.ftc.teamcode;

import com.qualcomm.hardware.bosch.BNO055IMU;
import com.qualcomm.robotcore.eventloop.opmode.Autonomous;
import com.qualcomm.robotcore.eventloop.opmode.Disabled;
import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
import com.qualcomm.robotcore.hardware.CRServo;
import com.qualcomm.robotcore.hardware.DcMotor;
import com.qualcomm.robotcore.hardware.DcMotorSimple;
import com.qualcomm.robotcore.hardware.Servo;
import com.qualcomm.robotcore.util.ElapsedTime;

import org.firstinspires.ftc.robotcore.external.ClassFactory;
import org.firstinspires.ftc.robotcore.external.hardware.camera.WebcamName;
import org.firstinspires.ftc.robotcore.external.navigation.VuforiaLocalizer;
import org.firstinspires.ftc.robotcore.external.tfod.Recognition;
import org.firstinspires.ftc.robotcore.external.tfod.TFObjectDetector;

import java.util.List;

@Autonomous (name="Enter Name Here", group="Red")
@Disabled
public class BlankRed extends LinearOpMode
{

    private static final String TFOD_MODEL_ASSET = "Skystone.tflite";
    private static final String LABEL_FIRST_ELEMENT = "Stone";
    private static final String LABEL_SECOND_ELEMENT = "Skystone";

    /*
     * IMPORTANT: You need to obtain your own license key to use Vuforia. The string below with which
     * 'parameters.vuforiaLicenseKey' is initialized is for illustration only, and will not function.
     * A Vuforia 'Development' license key, can be obtained free of charge from the Vuforia developer
```

```
34       * web site at https://developer.vuforia.com/license-
   manager.
35       *
36       * Vuforia license keys are always 380 characters long,
   and look as if they contain mostly
37       * random data. As an example, here is a example of a
   fragment of a valid key:
38       *      ...
   yIgIzTqZ4mWjk9wd3cZO9T1axEqzuhxoGlfOOI2dRzKS4T0hQ8kT ...
39       * Once you've obtained a license key, copy the string
   from the Vuforia web site
40       * and paste it in to your code on the next line, between
   the double quotes.
41       */
42      private static final String VUFORIA_KEY = "AQUWr4X/////
   AAABme38EPssRkvls9+q/
   BGPYgxKXBXELWHMdkTcCqUqHeyDpyXGWFLCTABgDXEMGe1EmsnDQxmJ7WQ069J
   3YSv+kOcfq3g2EnwZr2O3DujsIU1nT0aXgLlAtQU2r7wWAgHvR9ADO5pe/
   q7MzCyhjSTQLCgizGFLgmqfre0A9rjYcXYbYw11R3P7VRHnL3QHn3QH2oFVQfM
   b+dIzmZkfv0cd5qWvdhjovYF8hpZ/
   HT7veIa8ZQ9CIQ0541pxplXVud80z1xWpjFGJPaoQGO+xKWZ8E+
   Zlu7z5umiaV1+
   ChGeJ9pPyIJn0LsnoIHumZoYb4di4tFygMPVmH8ChsTlGJjaPBSCRBFjxzBqsX
   mBZY7eCa6S";
43
44       /**
45        * {@link #vuforia} is the variable we will use to store
   our instance of the Vuforia
46        * localization engine.
47        */
48       private VuforiaLocalizer vuforia;
49
50       /**
51        * {@link #tfod} is the variable we will use to store our
   instance of the TensorFlow Object
52        * Detection engine.
53        */
54       private TFObjectDetector tfod;
55
56       //Wheel Motors
57       DcMotor leftFront;
58       DcMotor rightFront;
```

```
59      DcMotor leftBack;
60      DcMotor rightBack;
61
62      //Foundation Servos
63      Servo leftFoundation;
64      Servo rightFoundation;
65
66      //Turret Motor
67      DcMotor turret;
68
69      //Lift Motor
70      DcMotor lift;
71
72      //Arm Motor
73      DcMotor arm;
74
75      //Claw Continuous Rotation Servo
76      CRServo leftClaw;
77      CRServo rightClaw;
78
79      Servo capstone;
80
81      //Gyroscope
82      BNO055IMU imu;
83
84      //Robot Classes
85      DriveTrain dT;
86      Armstrong a;
87
88      //State Machine Class
89      RedStates rS;
90
91      //Timer
92      ElapsedTime runtime;
93
94      //Stone counter
95      int count = 1;
96      //Extra distance for each stone
97      int extraTick;
98
99      //Boolean for if the skystone has been found
100     boolean skystone = false;
```

```
101
102     //The label of the object the program sees
103     String label;
104
105     @Override
106     public void runOpMode()
107     {
108
109         // The TFObjectDetector uses the camera frames from
    the VuforiaLocalizer, so we create that
110         // first.
111         initVuforia();
112
113         if (ClassFactory.getInstance().
    canCreateTFObjectDetector()) {
114             initTfod();
115         } else {
116             telemetry.addData("Sorry!", "This device is not
    compatible with TFOD");
117         }
118
119         /**
120          * Activate TensorFlow Object Detection before we
    wait for the start command.
121          * Do it here so that the Camera Stream window will
    have the TensorFlow annotations visible.
122          **/
123         if (tfod != null) {
124             tfod.activate();
125         }
126
127         //Setting variables to the real life components using
     the configuration on the phone.
128         leftFront = hardwareMap.dcMotor.get("leftFront");
129         rightFront = hardwareMap.dcMotor.get("rightFront");
130         leftBack = hardwareMap.dcMotor.get("leftBack");
131         rightBack = hardwareMap.dcMotor.get("rightBack");
132
133         leftFoundation = hardwareMap.servo.get("
    leftFoundation");
134         rightFoundation = hardwareMap.servo.get("
    rightFoundation");
```

```
135
136         turret = hardwareMap.dcMotor.get("turret");
137
138         lift = hardwareMap.dcMotor.get("lift");
139
140         arm = hardwareMap.dcMotor.get("arm");
141
142         leftClaw = hardwareMap.crservo.get("leftClaw");
143         rightClaw = hardwareMap.crservo.get("rightClaw");
144
145         capstone = hardwareMap.servo.get("capstone");
146
147         imu = hardwareMap.get(BNO055IMU.class, "imu");
148
149         //Setting up the classes to run using the variables
    above
150         dT = new DriveTrain(gamepad1, gamepad2 ,leftFront,
    rightFront, leftBack, rightBack, leftFoundation,
    rightFoundation);
151         a = new Armstrong(gamepad1, gamepad2,  turret, lift,
    arm, leftClaw, rightClaw, capstone, imu);
152         runtime = new ElapsedTime();
153
154         //Set the motors to stop and stay still instead of
    removing all power and coasting
155         leftFront.setZeroPowerBehavior(DcMotor.
    ZeroPowerBehavior.BRAKE);
156         leftFront.setDirection(DcMotorSimple.Direction.
    REVERSE);
157
158         rightFront.setZeroPowerBehavior(DcMotor.
    ZeroPowerBehavior.BRAKE);
159
160         leftBack.setZeroPowerBehavior(DcMotor.
    ZeroPowerBehavior.BRAKE);
161         leftBack.setDirection(DcMotorSimple.Direction.REVERSE
    );
162
163         rightBack.setZeroPowerBehavior(DcMotor.
    ZeroPowerBehavior.BRAKE);
164
165         turret.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior
```

```
165 .BRAKE);
166
167         lift.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior.
    BRAKE);
168
169         arm.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior.
    BRAKE);
170
171         //Stops Robot
172         dT.kill();
173         dT.release();
174         a.kill();
175
176         //Setting up Red States
177         rS = new RedStates(leftFront, rightFront, leftBack,
    rightBack, leftFoundation, rightFoundation, turret, lift, arm
    , leftClaw, rightClaw, dT, a, runtime);
178
179         /** Wait for the game to begin */
180         telemetry.addData(">", "Press Play to start op mode")
    ;
181         telemetry.update();
182
183         waitForStart();
184
185         //Resets Timer
186         runtime.reset();
187
188         //If the play button is pressed
189         //Uncomment to activate skystone detection
190         /*if (opModeIsActive())
191         {
192
193             //While the state machine is still running
194             while (rS.getState() != "finished")
195             {
196
197                 //Run the state machine method
198                 rS.runFull();
199
200                 //Telemetry that displays the current state
201                 telemetry.addData("State>", rS.getState());
```

```
202
203                 //Display Telemetry
204                 telemetry.update();
205
206             }
207
208             //Loops the program
209             while (opModeIsActive())
210             {
211
212                 if (tfod != null)
213                 {
214
215                     // getUpdatedRecognitions() will return
    null if no new information is available since
216                     // the last time that call was made.
217
218                     List<Recognition> updatedRecognitions =
    tfod.getUpdatedRecognitions();
219
220                     if (updatedRecognitions != null)
221                     {
222
223                         telemetry.addData("# Object Detected
    ", updatedRecognitions.size());
224
225                         // step through the list of
    recognitions and display boundary info.
226
227                         int i = 0;
228
229                         for (Recognition recognition :
    updatedRecognitions)
230                         {
231
232                             telemetry.addData(String.format("
    label (%d)", i), recognition.getLabel());
233                             //Sets the label to the name of
    the recognition
234                             label = recognition.getLabel();
235
236                             telemetry.addData(String.format
```

```
236 ("  left,top (%d)", i), "%.03f , %.03f",
237                                     recognition.getLeft(),
    recognition.getTop());
238
239                                 telemetry.addData(String.format
    ("  right,bottom (%d)", i), "%.03f , %.03f",
240                                     recognition.getRight(),
    recognition.getBottom());
241
242                                 telemetry.update();
243                             }
244
245                             //If label equals Skystone or if two
    stones have been detected
246                             if(label == "Skystone" || count == 3)
247                             {
248
249                                 //Telemetry for detecting the
    Skystone
250                                 telemetry.addData("Guess What", "
    Skystone Baby");
251
252                                 //Displays telemetry
253                                 telemetry.update();
254
255                                 //Sets the Skystone flag to true
256                                 skystone = true;
257
258                                 //Stops robot
259                                 dT.kill();
260
261                                 //Exits while loop
262                                 break;
263
264                                 //If the label detects Stone and
    the timer has been going for one second
265                             } else if (label == "Stone" &&
    runtime.milliseconds() >= 1000)
266                             {
267
268                                 //Telemetry for not seeing a
    Skystone
```

```
269                             telemetry.addData("Guess What", "
    Nothing!");
270
271                             //Displays Telemetry
272                             telemetry.update();
273
274                             //Adds one to the count number
275                             count++;
276
277                             //Adds to the extra tick distance
278                             extraTick = extraTick + 380;
279
280                             //Moves left to next stone
281                             dT.left(0.25, 380);
282
283                             //Resets timer
284                             runtime.reset();
285
286                             //If the robot times out
287                         } else if (runtime.milliseconds() >=
    1500)
288                         {
289
290                             //Rotate arm up to create
    movement to help detection
291                             a.rUp(0.25, 50);
292
293                             //Resets timer
294                             runtime.reset();
295
296                         }
297
298                         //Displays Telemetry
299                         telemetry.update();
300
301                     }
302                 }
303             }
304         }*/
305
306     }
307
```

```
308     /**
309      * Initialize the Vuforia localization engine.
310      */
311     private void initVuforia() {
312         /*
313          * Configure Vuforia by creating a Parameter object,
    and passing it to the Vuforia engine.
314          */
315         VuforiaLocalizer.Parameters parameters = new
    VuforiaLocalizer.Parameters();
316 
317         parameters.vuforiaLicenseKey = VUFORIA_KEY;
318         parameters.cameraName = hardwareMap.get(WebcamName.
    class, "Webcam 1");
319 
320         //  Instantiate the Vuforia engine
321         vuforia = ClassFactory.getInstance().createVuforia(
    parameters);
322 
323         // Loading trackables is not necessary for the
    TensorFlow Object Detection engine.
324     }
325 
326     /**
327      * Initialize the TensorFlow Object Detection engine.
328      */
329     private void initTfod() {
330         int tfodMonitorViewId = hardwareMap.appContext.
    getResources().getIdentifier(
331                 "tfodMonitorViewId", "id", hardwareMap.
    appContext.getPackageName());
332         TFObjectDetector.Parameters tfodParameters = new
    TFObjectDetector.Parameters(tfodMonitorViewId);
333         tfodParameters.minimumConfidence = 0.8;
334         tfod = ClassFactory.getInstance().
    createTFObjectDetector(tfodParameters, vuforia);
335         tfod.loadModelFromAsset(TFOD_MODEL_ASSET,
    LABEL_FIRST_ELEMENT, LABEL_SECOND_ELEMENT);
336     }
337 
338 }
```