Java API - Team 08

**Navigation class:**

Contains all the methods concerning the movements of the two wheels of the robot.

*Method summary*

public void goForward(Odometer odo, double distance)  
Make the robot travel forward for a given distance.

Takes as parameters the odometer of the robot and the distance.

public void turnTo(Odometer odo, double angle)

Make the robot turn to an absolute angle.

Takes as parameters the odometer of the robot and the absolute angle.

public void turn(Odometer odo, double angle)

Make the robot turn to a relative angle.

Takes as parameters the odometer of the robot and the relative angle.

**Odometer class implements TimerListener:**

Contains all the methods concerning the position of the center of rotation of the robot.

*Method summary*

public Odometer(Robot robot, boolean start)

Constructor of the odometer.

Takes as parameters the robot on which the odometer will be implemented and a boolean to start it.

public void setPosition (double [] pos, boolean [] update)

Set the position of the odometer, according to the array of position, if the matching element in the array of booleans is true.

Takes as parameters the array of position (x,y and angle) as doubles and an array of booleans to indicate the updated position.

public static double fixAngle(double angle)

Set the angle to a value between 0 and 360 degrees.

Takes as parameter the angle which needs to be fixed.

public static double minimumAngleFromTo(double a, double b)

Indicate which angle is minimal to get from an angle to another.

Takes as parameters the beginning angle "a" and the wanted angle "b", as doubles.

**Main class**

Initialize the robot and its components.

*Method summary*

public static void main(String[] args)

Initiate the robot, the light sensors, the odometer, the ultrasonic sensor(s), the LCD display and starts the program, according to the Bluetooth class.

**Flag localizer**

Localize the flag on the field and get to it.

*Method summary*

public FlagLocalizer(Odometer odo, Lightsensor ls1, Lightsensor ls2, UltrasonicSensor us)

Construct the flag localizer and initiate its sensors.

Takes as parameters the robot's odometor, the two light sensors and the ultrasonic sensor.

public int turnAndScanFloor(int numOfDegrees)

Method that makes the robot turns a certain number of degrees and returns the value read from the light sensor at the floor level.

Takes as parameters the number of degrees that the robot needs to turn.

public int turnAndScanTop(int numOfDegrees)

Method that makes the robot turns a certain number of degrees and returns the value read from the light sensor at the top level (when the arm is fully extended).

Takes as parameters the number of degrees that the robot needs to turn.

public double getBrightestOrientation()

Method that makes the robot find the orientation at which the light is the brightest. The robot first rotates 360 degrees while the light sensor is at the floor level then rotates 360 degrees while the light sensor is at the top level. It then returns the orientation at which the maximum light value was read.

Takes no parameter.

public void getToFlag()

Method that gets the robot to the flag, using the getBrightestOrientation method along with the obstacleAvoidance method.

Takes no parameter.

public void obstacleAvoidance(double orientation)

Method which uses the ultrasonic sensor in order to avoid the obstacle and get around them.

Takes as parameters the orientation that the robot needs to have once the obstacle has been avoided.

**RobotLocalizer**

Localize the robot using the ultrasonic sensor as well as the light sensor.

Method Summary

public RobotLocalizer(Odometer odo, UltrasonicSensor us, LightSensor ls)

Construct the RobotLocalizer and initiate its sensors.

Takes as parameters the robot’s odometer, an ultrasonic sensor and a light sensor.

public void doUSLocalization()

Method has the robot turn and scan for walls so the odometer’s heading can be updated.

Takes no parameters.

public void doLSLocalization()

Method has the robot turn and scan for gridlines so the odometer’s position and heading can be updated.

Takes no parameters.

public void getFilteredData()

Method applies simple filter to spurious 255 values.

Takes no parameters.

**ArmMovement**

Controls arm movement and grabbing.

Method Summary

public void pickUp()

Method moves arm motor to object and grabs it.

Takes no parameters.

public void putDown()

Method moves arm motor and drops object at a location.

Takes no parameters.

public void grab()

Method controls claw movement to grab something.

Takes no parameters.

public void release()

Method controls claw movements to release something.

Takes no parameters.

**Robot**

Controls movement of robot and takes care of calculations of robot displacement and heading.

Method Summary

public Robot (NXTRegulatedMotor leftMotor, NXTRegulatedMotor rightMotor)

Construct the Robot.

Takes motors responsible for movement as parameters.

public double getDisplacement()

Method calculates displacement of robot and returns the result as a double.

Takes no parameters.

public double getHeading()

Method calculates heading of robot and returns the result as a double.

Takes no parameters.

public void getDisplacementAndHeading(double [] data)

Method calculates both heading and displacement and stores values in data[].

Takes a double array as a parameter.

public void setForwardSpeed(double speed)

Method sets forward speed to the double speed passed to the method and updates the motors’ speeds.

Takes a double as a parameter.

public void setRotationSpeed(double speed)

Method sets rotational speed to the double speed passed to the method and updates the motors’ speeds.

Takes a double as a parameter.

public void setSpeeds(double forwardSpeed, double rotationalSpeed)

Method sets forward and rotation speed to the doubles forwardSpeed and rotationSpeed respectively. It then calculates the speeds to be passed to the motors and gets them moving.

Takes the doubles which store the values of the forward and rotation speeds to set as parameters.

**LCDInfo implements TimerListener**

Displays values from the odometer.

Method Summary

Public LCDInfo(Odometer odo)

Construct the LCDInfo and give it access to the odometer’s readings.

Takes the robot’s odometer as input.

public void timedOut()

Method draws the position and heading values from odometer to the LCD screen.

Takes no parameters.