**Programming with “while” Loops**

Here are two programs that use ‘while’ loops and if statements. The first program has a pretty simple structure, but the second one includes if statements inside while loops, while loops inside other while loops, and more. To try a programs:

* Create a “robot” that supports the features of the respective program as described in the comments.
* Copy the program to the clipboard
* Use Eclipse to create a new class and make sure that the class name is the same as in the program you copied. For the first program, for example, create a new class named “SoundProbe”.
* Replace the entire code by pasting the copied program.

Your file should not have any errors and you can execute it as usual on your brick (of course only after you build your robot according to the directions in the comments).

**Program 1: SoundProbe**

**import** lejos.nxt.Button;

**import** lejos.nxt.LCD;

**import** lejos.nxt.SensorPort;

**import** lejos.nxt.Sound;

**import** lejos.nxt.UltrasonicSensor;

/\* Program to create a "sound probe" that makes a sound proportional

 \* to the distance to an obstacle. If the distance is small, the sound

 \* is high, if the distance is large, the sound is low.

 \*

 \* @author Bert G. Wachsmuth

 \* @version 1.0 (03/04/2014)

 \*/

**public** **class** SoundProbe

{

 // State variables: Our 'robot' really only consists of the distance sensor.

 // NOTE: the sensor MUST be plugged in to sensor port S4!!

 **static** UltrasonicSensor *echo* = **new** UltrasonicSensor(SensorPort.*S4*);

 // Standard main method (where the program begins)

 **public** **static** **void** main(String[] args)

 {

 // Setting the speaker volume to 50% (to be less annoying :-)

 Sound.*setVolume*(50);

 // We want to continuously probe the distance, so we need a loop. It will

 // execute as long as the ESCAPE button is NOT pressed (NOT is !).

 **while** (!Button.*ESCAPE*.isDown())

 {

 // Reading the distance from the sensor and storing it in a variable

 **int** distance = *echo*.getDistance();

 // Showing the distance on the LCD screen

 LCD.*drawString*("Dist = " + distance + " ", 0, 0);

 // Converting distance to freq (an "inverse proportional relation").

 // We create the conversion so that:

 // \* minimum frequency is 500 (distance is very large)

 // \* max frequency is 10,000 / 10 + 500 = 1,500 (distance is small)

 **int** freq = 10000 / (distance + 10) + 500;

 // Now using that fequency to create a tone for 0.5 seconds

 Sound.*playTone*(freq, 500);

 }

 // End of loop (if the ESCAPE button is pressed); program will exit

 }

}

**Program 2: AvoidWall**

**import** lejos.nxt.Button;

**import** lejos.nxt.LCD;

**import** lejos.nxt.Motor;

**import** lejos.nxt.NXTRegulatedMotor;

**import** lejos.nxt.SensorPort;

**import** lejos.nxt.Sound;

**import** lejos.nxt.UltrasonicSensor;

/\*

 \* Program to help a differential drive robot to avoid a wall of unknown width

 \* and to escape through a gap in the wall if it finds one.

 \*

 \* NOTE: The program uses the "turn" and "drive" functions we created previously

 \* to turn the robot in place by a given degree and to drive it forward by

 \* a given distance, respectively. Verify that these functions work properly;

 \* they depend on how your particular robot is constructed.

 \*

 \* SETUP: The robot should be placed perpendicular to a wall, and the wall should

 \* have an opening (a door) somewhere to the left of the robot. The robot will then

 \* avoid the wall, find the door, and drive though the door, hopefully.

 \*

 \* @author Bert Wachsmuth

 \* @version 1.0 (03/04/2014)

 \*/

**public** **class** AvoidWall

{

 // Constants

 **final** **static** **double** *PI* = 3.1415;

 // State variables defining the robot. Our differential drive robot has:

 //

 // \* wheels of radius 2.8 cm

 // \* the left motor plugged in to motor port C

 // \* the right motor plugged in to motor port A

 // \* a distance sensor facing forward, plugged in to sensor port 4

 //

 // Verify: you need to adjust it to match this setup, or adjust the program accordingly.

 **static** **double** *radius* = 2.8;

 **static** NXTRegulatedMotor *leftMotor* = Motor.*C*;

 **static** NXTRegulatedMotor *rightMotor* = Motor.*A*;

 **static** UltrasonicSensor *sensor* = **new** UltrasonicSensor(SensorPort.*S4*);

 // Turns robot by 'angle' degrees. If angle > 0, turns right, otherwise turns left.

 **public** **static** **void** turn(**int** angle)

 {

 // You need to calibrate (i.e. adjust if necessary) this number to make sure

 // that an angle of 90 degrees really yields a 90 degree turn. Use trial and

 // error to find the best possible multiplier.

 **double** multiplier = 3.0;

 *leftMotor*.rotate((**int**)(angle\*multiplier), **true**);

 *rightMotor*.rotate((**int**)(-angle\*multiplier));

 }

 // Drives forward by 'distance' cm.

 **public** **static** **void** drive(**int** distance)

 {

 // This time we are computing the correct multiplier mathematically, but you

 // should still verify that the robot really drives for the given distance.

 **double** angle = distance / *radius* \* 180/ *PI*;

 leftMotor.rotate((**int**)angle, **true**);

 rightMotor.rotate((**int**)angle, **false**);

 }

 // Function to drive forward until robot is 30 cm in front of the wall, then stops.

 **public** **static** **void** proceed()

 {

 **while** (*sensor*.getDistance() > 30)

 {

 *leftMotor*.forward();

 *rightMotor*.forward();

 }

 *leftMotor*.stop();

 *rightMotor*.stop();

 }

 // Function to avoid the wall. Simply turns left, drives 20 cm, then turns right

 // again to face in the original direction.

 **public** **static** **void** avoid()

 {

 *turn*(-90);

 *drive*(20);

 *turn*(90);

 }

 // Standard main function (where the program starts executing)

 **public** **static** **void** main(String[] args)

 {

 // Setting up volume and speed of the motors.

 Sound.*setVolume*(50);

 *leftMotor*.setAcceleration(2000);

 *rightMotor*.setAcceleration(2000);

 *leftMotor*.setSpeed(200);

 *rightMotor*.setSpeed(200);

 // We don't know how far away the door is, so we use a loop

 **while** (!Button.*ESCAPE*.isDown())

 {

 // proceed: drive forward until you are 30 cm in front of the wall

 *proceed*();

 // avoid: Avoid wall by turning left, drive a distance, then turn right

 *avoid*();

 }

 }

}