Agreement

By working on and completing this laboratory assignment, you agree to use the hardware given to you in a responsible manner. Each group is responsible for the safety and security of the robot while it is in your possession. You are not allowed to take the robots outside of Alden Hall, without instructor’s permission. If you need to complete the assignment outside of the laboratory session time, you can obtain your robot from room 107 between 8am and 5pm, and you must return it to room 107 when you are finished working with it that day. If you need to work with the robot beyond 5pm or would like to take it outside Alden Hall ask the instructor or a teaching assistant for access.

Objectives

To develop a program for patrolling robot using a while loop, if/else statements, boolean operators, arrays, built-in methods in lejos (Java for Lego robots), and to continue to develop your problem solving skills. Additionally, to enhance your experience with the Java development environment called “Eclipse” that is widely used in practice, and to develop practical and hands-on skills of programming using Lego robots.

General Guidelines for Labs

- **Work on the Alden Hall computers.** If you want to work on a different machine, be sure to transfer your programs to the Alden machines and re-run them before submitting.

- **Update your repository often!** You should add, commit, and push your updated files each time you work on them. I will not grade your programs until the due date has passed.

- **Review the Honor Code policy.** You may discuss programs with others, but programs that are nearly identical to others’ will be taken as evidence of violating the Honor Code.

Reading Assignment

To learn more about while loops, if/else statements, boolean and comparison expressions, review Sections 5.1–5.4 in your textbook. Since this assignment will also require you to continue to use Java methods, you should once again review Section 4.4. To best prepare for the new content in this laboratory assignment, you should study Sections 8.1-8.2, and you should also study leJOS API documentation at [http://www.lejos.org/ev3/docs/] (http://www.lejos.org/ev3/docs/).
Set Up lejos and Start the Project

leJOS is a small Java Virtual Machine, which allows us to write Java programs for Lego robots (http://www.lejos.org). You will be using lejos to write Java program(s) for a Lego EV3 robot using a plugin (add-on software) in a development environment called “Eclipse”. Eclipse is already installed on the computers in all of the labs in Alden Hall. However, you need to complete the following instructions to configure Eclipse to work on leJOS and to recognize the Lego Robots.

1. In your own “cs111f2016-<your user name>” repository inside the “labs/” directory, create a directory called “lab9”.

2. Search for “Eclipse” using icon and launch Eclipse by clicking on its icon .

3. When the “Workspace Launcher” comes up, navigate your workspace to the directory (“lab9”) you created in step 1.

4. Connect your robot using a given USB cable to the computer and turn on your robot by pushing the middle square button on the robot.

5. Using the top menu in Eclipse, click on “Window”, select ”Preferences”, you will see a small window pop up as in the snapshot shown below. Click on “leJOS EV3” on the left-side of the menu, check boxes for “Run Tools in separate JVM”, “Connect to named brick”, and “Run program after upload”. Verify that “EV3_HOME” has “/opt/lejos-ev3” field populated next to it. Then, next to the “Connect to named brick” you have to specify the IP address of your robot, it should be 10.0.1.1, but you may want to verify this by cross-checking it with the second line displayed on your robot’s LCD screen. Click “OK”.

6. Now you are ready to start developing Java programs for a Lego robot!

Lego EV3 robot

Lego EV3 robot contains a programmable EV3 controller, called a “brick”, user interface with an LCD screen, Lego building blocks, motors and various sensors such as light, ultrasonic, color, touch, gyro. The brick contains four outputs (motors), four inputs (sensors), USB, Bluetooth, or Wi-Fi connection, LCD screen, 16 MB flash memory, 64 MB RAM, SD Card Port, EV3 Brick Button lights, sound. EV3 Brick buttons are explained in the figure below:
Sample Program

First of all, you will create and run a sample program. The first part of the program will simply output “Hello Word!” to the LCD screen of the robot, and the second part of the program will make the robot move forward, backward, rotate left and rotate right. Please complete the following steps to execute this program on the Lego robot.

1. The first thing you need to do is to create a leJOS EV3 project. Using the top menu in Eclipse, click on “File”, “New”, “Project”. This will pop up a small window as shown below. Click on “LeJOS EV3 Project”, click “Next”. In the new small window specify the name for your project as “lab9”, change the “Use an execution environment JRE” field’s value to ‘JavaSE-1.7’. Then click “Finish”. This will add “lab9” directory in your “Package Explorer” view on the left-side panel in Eclipse.

2. Expand your new project in the Package Explorer in Eclipse, click on “src”, then right-click and select “New”, “Package” to create a Java package. Name your package “lab9” and click “Finish”.

3. Right-click on “lab9” package in the “Package Explorer” and select “New”, “Class” to create
a main Java class. It is a good idea to check the box that causes the main method to be created for you.

4. Copy and paste the following Java statements into your Sample class’s main method:

```java
// print "Hello Word!" to the LCD screen
System.out.println("Hello World!");
// Wait for 5000 milliseconds
Delay.msDelay(5000);
```

Add the following import statement:
```java
import lejos.utility.Delay;
```
You can also easily do this by clicking on little yellow light bulb with a red square ‘x’ by the line numbers and selecting appropriate “import”.

5. Make sure your EV3 robot is turned on, is connected to the computer via a USB cable and its menu is running. Now run your sample program by right-clicking of the class name from the “Package Explorer”, and selecting “Run As”, “leJOS EV3 Program”. You should then see the message on the EV3 robot’s LCD screen. You will also see some messages in the leJOS EV3 Eclipse console view, this is ok. Was this “Hello World!” message displayed on your robot’s screen?

6. Now add another print statement and run the program again. Did you see both messages being displayed on the screen? Try to change the value inside msDelay method. What happens if the value of the delay is very minimal?

7. Now return to your Sample class in Eclipse, and copy and paste the lines of code from the provided sampleprogram.txt inside your main method at the end of existing Java statements.

8. Now add the following “import” statements:

```java
import java.io.IOException;
import lejos.hardware.Button;
import lejos.robotics.RegulatedMotor;
import lejos.robotics.navigation.DifferentialPilot;
import lejos.utility.Delay;
import lejos.utility.PilotProps;
```

9. Finally add throws IOException to the end of the main method declaration. Your main method will then be: public static void main(String[] args) throws IOException

IOException is a class of Exception that was created due to all Input/Output contingencies and it is designed to identify any input or output problems that might occur.

10. Study this more complicated code and the comments included with it. This part of the program allows the robot to travel forward, backward, then rotate left and right. You may also want to study the leJOS API documentation at [http://www.lejos.org/ev3/docs/](http://www.lejos.org/ev3/docs/) for the usage of the classes and methods in this sample code.

11. Following the step 5, run the completed program on your robot. After you see the following lines appear in your Eclipse “Console” window at the button, you can unplug the USB cable.
Jar file has been created successfully
Using the EV3 menu for upload and to execute program
Uploading to 10.0.1.1 ...
Program has been uploaded
Running program ...
leJOS EV3 plugin launch complete

Also, after you upload the program from Eclipse, if you would like to run the same program more than once, you do not have to upload it again - you can just use EV3 robot’s menu, go to “Programs” and click on the program you wish to run. However, if you make changes to the program, then you have to upload the new version to the robot using a USB cable.

12. Study the DifferentialPilot class in the leJOS API. What other methods do you see that could be useful when writing programs that control the movements of the robot?

13. Study the Button class in the leJOS API. What other method in the Button class could you use instead of waitForAnyPress();?

Patrol Robot
For this laboratory assignment, your task is to create a robot that patrols a specified area a certain number of times (at least three times). Using the program you created above as a reference program, you will develop two Java programs that will complete the following tasks:

1. First, you need to repeat step 3 from Section “Sample Program” to create a new Java class in your “lab9” package, name it “Lab9”, or something similar.

2. In Lab9 class, write Java statements to display an appropriate welcome message. Wait for the user to push the enter button. Once the button is pushed, play a sound (beep, two beeps, or buzz, etc.), and proceed to step 2. In order to play the sound, you can use the methods in the Sound class in lejos API [http://www.lejos.org/ev3/docs/]. For example, in order to make the robot buzz, you can call Sound class method “buzz” as follows: Sound.buzz();

3. In Lab9 class, add Java statements (loop) that will allow the robot to patrol some area a specified number of times (at least three times). The actual patrol behavior (how the robot patrols and which area it patrols) is to be implemented separately (described below).

4. Create a new Java class that will have a constructor and three methods, called patrol, detectObject and checkVoltage. To implement the patrol method, you need to think about what your area will be. You can define it to be a small square, rectangle, circle, etc. It has to be some predefined area, your robot should not be driving around randomly. patrol method should be called from the loop inside Lab9 class. Think about what parameter(s) you need to pass to this method, to allow your robot to move.

You may find it useful to explore several other methods in the DifferentialPilot class, in addition to the ones used in the sample program. Methods such as travel, steer, rotate can be helpful in making your robot move appropriately. Please see leJOS API documentation [http://www.lejos.org/ev3/docs/] under DifferentialPilot class for more useful methods.
// Moves the robot 10cm forward in a straight line
robot.travel(10)
// Moves the robot 10cm backward in a straight line
robot.travel(-10)
// Moves the robot along a curved path given by the turn rate = 50 and the turn angle = 180
// If angle is positive, the robot will move in the direction that increases its heading (it turns left).
// If angle is negative, the robot will move in the direction that decreases its heading (turns right).
// If angle is zero, the robot will not move and the method returns immediately.
// Turn rate specifies the sharpness of the turn.
robot.steer(50, 180);
// Rotates the robot through 90 degree angle (clockwise)
robot.rotate(90);

5. Ensure your robot has an ultrasonic sensor correctly connected to it. Then add `detectObject` method to your new program to detect the object in front of the robot. This method should be called from `Lab9` class. While the robot is patrolling, it should stop and check for objects in front of it. To detect an object, you first need to create an instance of the ultrasonic sensor as `EV3UltrasonicSensor lightSensor = new EV3UltrasonicSensor(SensorPort.S3);` (this should be created in the `Lab9` class and passed as a parameter to `detectObject` method), assuming the ultrasonic sensor is connected to port 3. Then, you should use `getDistanceMode()` method, to set up the distance detection. Finally you can use a method `fetchSample` in the `SampleProvider` class in lejos and an array to actually take the distance sample and store it in the array. For example, to complete these tasks you can do the following:

```java
EV3UltrasonicSensor sonicSensor = new EV3UltrasonicSensor(SensorPort.S2);
SampleProvider sonicSamplePr = sonicSensor.getDistanceMode();
float[] sonicSample = new float[sonicSamplePr.sampleSize()];
sonicSamplePr.fetchSample(sonicSample, 0);
```

Your robot needs to have an object detection built-in, which means that it should detect the object and instead of hitting it, it should try to avoid it. You can use an `if` statement, to check if the distance between the robot and the object is less than some specified threshold, if it is (meaning it is too close to the object), then the robot would try to avoid the object. If your robot detects an object in front of it, it should play a sound and then try to go around it in some way as it proceeds with its patrol.

6. In addition to checking for an object, your robot should also check its voltage, and it should stop its patrol if its voltage is getting low (below some threshold). This behavior should be implemented in the `checkVoltage` method inside your new class and it should be called from the `Lab9` class. Voltage indicates whether the battery is at a usable level. Usually, if voltage is below 1 Volt, the battery is empty. The rechargable batteries should not be discharged to that level, because that can destroy them. To check the robot’s voltage, you can use static `getVoltage()` method as: `Battery.getVoltage()`.

7. Finally, when your robot is finished with its patrol task a specified number of times, it should play a celebratory tone and close the sensors it has been using (using `close()` method). In leJOS, you can use `playTone` method to play a tone with certain frequency and duration.

```java
Sound.playTone(300, 500); // frequency 300, duration 1/2 second
```
Delay.msDelay(500); // wait half of a second while the sound plays
Sound.playTone(400,500); // frequency 400, duration 1/2 second
Delay.msDelay(1000); // let the tone play half a second and wait half of a second
Sound.playTone(500,1000); // frequency 500, duration 1 second
Delay.msDelay(1000); // wait a second

Please see the outline for your two classes below:

public class Lab9
{
    public static void main (String args [])
    {
        // 1. create a Differential pilot
        // 2. set up the ultrasonic sensor
        // 3. display the welcome message and play a sound
        // 4. implement a loop that will repeatedly call the patrol method
        // 5. while patrolling, repeatedly call detectObject and checkVoltage methods
        // Note: you can call these two methods different number of times
        // For example, detectObject can be called every time robot finishes one cycle of the patrolling,
        // while checkVoltage can be called less frequently
        // 6. when the robot is finished with its patrol task, play a tone.
    }
}

public class Patrol
{
    // 1. create instance variables (if any)
    // 2. create a constructor
    // 3. implement patrol method
    // 4. implement detectObject method
    // 5. implement checkVoltage method
}

Required Deliverables

For this assignment you are invited to submit versions of the following deliverables through the
Bitbucket repository by correctly using “git add”, “git commit”, and “git push” commands.

1. Completed, fully commented, and properly formatted versions of the source code files. Only
one submission from a team is required for this document.

2. A report document containing: the answers to the questions in red in the first part of the
assignment, a video showing your robot fulfilling its patrol task, and the description of the
strategy your team developed for completing this assignment. Only one submission from a
team is required for this document.

3. A reflection document explaining the work each team member has completed and if there
were any challenges you faced while working with your team member. A submission from
each team member is required.
When you are done, please ensure that the Bitbucket Web site has a “lab9/” directory in your repository with the Java files in the list of deliverables and the other files. Please see the instructor if you have questions about assignment submission.

In adherence to the Honor Code, students should complete this assignment in their assigned groups. If you use any resources online to learn about lejos, Lego EV3 robots, or Eclipse, you need to state that in the header of your class file. Deliverables that are nearly identical to the work of other groups or online resources will be taken as evidence of violating Allegheny College’s Honor Code.

**Additional Challenge:**
Want some extra credit? Add additional sensor(s) to your robot and enhance the patrolling behavior of your robot even more by using those sensors in some way.