# Java using LEGO Mindstorms and LeJOS

University of Idaho

# Contents

# Chapter 1

# Introduction

This text is an introduction to Java programming using the LEGO NXT robotics kit and the LeJOS Java library. LeJOS (pronounced like the Spanish word "lejos" for "far") is a Java Virtual Machine that can be run on the NXT (or the EV3) brick. It is installed on the brick as firmware and allows Java programs that are written with the LeJOS API to run directly on the brick.

These instructions focus on a Windows implementation, but all of the components should work on a MacOS computer as well.

In this text it is assumed that all of the programming will take place within the Eclipse IDE (Integrated Development Environment). Eclipse is freely available (see install directions below).

Note LeJOS consists of two major components. The first component is a version of the Java virtual machine that is loaded onto the Lego brick. If "Java virtual machine" doesn't mean much to you, basically its software that allows the brick to run Java programs. The second component is an Eclipse plugin that makes it easier to write programs the use LeJOS. "Plugin" basically means a bunch of predefined code that you can borrow to write your own programs.

the next section explains how to install Eclipse and the LeJOS plugin that goes with it. The section after that explains how to install LeJOS to a Lego brick.

## 1.1 Setting up Java and Eclipse

1. Install Java JDK x86 from:
   http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
   *Make sure to select x86, for your operating system. Don't select the x64 version.* I generally find it easiest to chose "save as" and save it to the desktop, then run it from the desktop after it downloads.
   Note that you will need to select the accept option on the page before downloading.

2. Install Eclipse. First, download Eclipse from:
   http://www.eclipse.org/downloads/
   *Select the 32 bit version.* Chose "save as" and save it to your desktop. once the file has downloaded, right-click on it and select "Extract all...". (The LeJOS site has instructions at: http://www.lejos.org/nxt/nxj/tutorial/Preliminaries/UsingEclipse.htm if you need additional details).

3. Run Eclipse. Double-click on the unzipped Eclipse folder and run the Eclipse.exe to start

Eclipse. The first time you run Eclipse it will ask what workspace to use, use the default workspace, which is called "workspace" and should already be filled in.

4. Load the LeJOS plugin in Eclipse. (These instructions are taken from the website: http://www.lejos.org/nxt/nxj/tutorial/Preliminaries/UsingEclipse.htm)

   (a) From within Eclipse select "Install new software..." from the help menu.

   (b) Paste the web address "http://www.lejos.org/tools/eclipse/plugin/nxj/" into the text box labeled "Work with:". Then hit Enter, *not* the add button.

   (c) Chose to install the leJOS plug-in and click the Next button. Click through the next few pages of the Install dialog and eventually click on Finish. Eclipse will now download the plug-in. The leJOS plug-in is not signed so you will get a warning pop-up – press the "okay" button.

   (d) The plugin should now install after which you will be prompted to restart Eclipse. Go ahead and restart Eclipse.

5. After you have restarted Eclipse jump to the sample program in Section 2.1 of this text to check that Eclipse installed properly.

## 1.2 Setting up the Lego Brick to work with LeJOS

An important part of LeJOS is the "Java Virtual Machine" code that gets loaded on to the Lego brick. This code allows the brick to run Java programs. This code is called *firmware* because it is *persistent* in the brick's memory. That means that when the brick is powered down the code stays in its memory, but it also means that it can be replaced, for example, if you want to go back to the original Lego firmware.

(To return to the original Lego firmware plug the brick in using a USB cable. (Firmware updates cannot be done via Bluetooth.) Start the Lego Mindstorm program. Go to the tools menu and select the "Update NXT Firmware" option. Press the download button.)

To install the LeJOS firmware requires the proper device driver software (software that tells your computer how to talk to a device). There are two options for installing the device driver:

*Option 1:* Install the original Lego Mindstorms software, this will include all of the required drivers.

*Option 2:* Go to:
http://www.robotc.net/support/troubleshooting/fantom-driver.php
Follow Step 1: download the zipped file "NXT_USB_Driver_120.zip".
Once it has been downloaded, right click on the zipped folder and select "extract all" to unzip it.
Open the unzipped folder and run "setup.exe".
We are using Eclipse and LeJOS, so it is not necessary to follow Step 2.

After the drivers have been installed you can upload (or "flash") the LeJOS firmware to the brick:

1. Plug in the Lego brick using a USB cable (firmware updates cannot be done using bluetooth).

2. Start Eclipse

3. From the LeJOS NXJ menu (near the center top of the Eclipse window) select "Upload Firmware" and follow the instructions. Note, if you don't see the LeJOS NXJ menu option it means that the Eclipse plugin for LeJOS was not installed properly, go back to the previous section.

4. Go to the sample program in Section 2.2 of this text to test that everything installed properly.

5. *If* you have a robot built, jump to the sample program in Section 2.3 of this text to test that every thing installed properly.

# Chapter 2

# Hello, World!

This chapter presents three basic programs. First is the Java version of the standard first program: Hello, World!. This program is pure Java and does not use LeJOS or the Lego brick. It is useful for testing that your installation of Java and Eclipse is correct, and can be used as the starting point for other Java programs.

Second is the LeJOS verions of Hello, World!. This program uses LeJOS and runs on the Lego brick. It is useful for testing that you installation of LeJOS is correct.

Third is the robot version of Hello, World!. This program uses LeJOS, runs on the Lego brick and requires that you have built a simple robot with two motors to drive it. (Such as the tribot at http://www.education.rec.ri.cmu.edu/content/lego/building/build_shows/rem.pdf)

## 2.1 Java Hello World!

Listing 2.1.2 presents the code for the basic Hello World program. To create the program in Eclipse you need to go through three general steps:

1. Create a new project – From the file menu select `new` > `Java project`. Name the project *HelloWorld*.

2. Create a new (main) class – Once the project is created you should see a folder named *HelloWorld* in the Package Explorer pane on the left hand side of the Eclipse window. Right click on the folder and select `new` > `class`. Name the class *HelloWorld*. *Make sure* to select the check box that says `public static void main(String[] args)` under the question "Which method stubs would you like to create?". This will create a `main()` function. Every Java project must have a `main()` function, which is where the program starts running. You should leave the "Inherited abstract methods" box checked.

3. Examine the code – When Eclipse creates the `HelloWorld` class it creates some default code that will appear in the center pane in Eclipse. The code looks like Listing 2.1.2.

4. Run the code – Click the green run triangle from near the middle of the control bar. You may not see anything happen – the program doesn't do anything yet – but you also shouldn't get any error messages.

### 2.1.1 Adding to the code

Enter the line
`System.out.println("Hello, World!");`
between lines 4 and 5 of the Hello, World! program. (Note that the code in Eclipse will not have line numbers, so you will need to refer to Listing 2.1.2 to know where to enter the new line of code.

Whenever you add code to a program, try to figure out what the commands do. Most will be difficult to understand at this point, but some of them may begin to make sense. As you get more practice with Java it will become easier to read code, even code that includes commands you haven't seen before.

### 2.1.2 Running Hello, World!

Run the new program by pressing the run button again. Now you should see the message:
`Hello, World!`
show up in the console pane, which is usually at the bottom of the Eclipse window.

Listing 2.1: Blank program created by Eclipse when you start a new project and class with a `main()` function. Keywords (also known as reserved words) are in bold, comments are in italics, and variables are colored maroon.

```
1 public class HelloWorld {
2
3         public static void main(String[] args) {
4                 // TODO Auto-generated method stub
5
6         }
7
8 }
```

There are a few things to notice about this new command. The first part of the command is `System.out`. The `System` part says that we want to use a command the accesses the operating system. The `.out` part says that we want an output *stream*, that is a place where we can send data for output to the screen. The `.println()` part says that we want to use the `println()` function that is part of the `out` stream. The `"Hello, World!"` part says that we want to send the *literal string* `Hello, World!` to the output stream (a literal string is just a string of characters). Finally, the command ends with a semicolon `;`. Semicolons are like the Java equivalent of periods – they are used to end most commands the way a period is used to end sentences.[1]

So, this statement can be though of as asking the operating system for an output stream and then using the `println()` function on that output stream to send data, in the form of a literal string, to the screen. Many variations on this basic command are possible in Java. Output streams can connect to other devices. For example, later we will use a different output stream to send data to the Lego brick. There are also input streams that are used to get data, for example, from the keyboard.

`println()` is a function, which is a separate block of code that completes a specific task. The `printlf()` task is to print a line of text. In this case the text "Hello, World!".

[1]Semicolons were probably originally chosen instead of actual periods in part because its too easy to confuse periods with decimals.

When you use a new piece of code it's a good idea to experiment with it to understand exactly how it works. It's almost impossible to damage a computer by experimenting with a program, at worst you'll end up with a program that has errors in it and will have to go back to an earlier, working version. In this case there are a couple of things to try:
Change the text in the `println()` function to make the program print a different message (don't forget to include the "'s).
Add additional `System.out.println()` commands to get the program to print more than one line.

## 2.2 LeJOS Hello, World!

The steps for creating a LeJOS project are almost the same as for other Java projects, except that you must chose LeJOS project:

1. Create a new LeJOS project – From the file menu select `new` > `project`.
2. From the LeJOS folder select `LeJOS NXT Project`. Select next and name the project *LejosHelloWorld*.
3. Create a new (main) class – Once the project is created you should see a folder named *LejosHelloWorld* in the Package Explorer pane on the left hand side of the Eclipse window. Right click on the folder and select `new` > `class`. Name the class *LejosHelloWorld*. *Make sure* to select the check box that says `public static void main(String[] args)` under the question "Which method stubs would you like to create?". This will create a `main()` function.
4. Examine the code – When Eclipse creates the `LejosHelloWorld` class it creates some default code that will appear in the center pane in Eclipse. Again the code looks like Listing 2.1.2 except that the name of the class in this project is `LejosHelloWorld`.
5. Run the code – Click the green run triangle from near the middle of the control bar. This time you should get a pop-up box. Choose run as `LeJOS NXT Project`. This is generally only necessary the first time a LeJOS project is run, after that Eclipse remembers the choice.

### 2.2.1 Adding to the code

Enter the lines
```
System.out.println("Hello, World!");
Button.waitForAnyPress();
```
between lines 4 and 5 of the program. Eclipse should automatically add the line:
```
import lejos.nxt.Button;
```
at the beginning of the program.

(Note: if Eclipse doesn't add the extra line you should see a red X to the left of the second line that was added and the word `Button` will have a red line under it. Click on the word `Button` and select the first quick fix: "import 'Button' (LeJOS NXT)". If you don't get this option it probably means that LeJOS was not installed properly.)

Now run the code again. You should see the screen on the Lego Brick say "Hello, World!". Press any key on the brick to get back to the menu.

The first line that was added is the same as before, it prints the literally string "Hello, World!", but this time the code is running of the brick, so the words appear on the brick.

The second line: `Button.waitForAnyPress();` does what it says, it causes the program to pause and wait for a button to be pressed. Because the code is running the brick its waiting for a button on the brick to be pressed – pressing a keyboard button won't have any effect. Without this line the program would print "Hello, World!", but then immediately end and switch back to the brick's main menu and you wouldn't get to see the message.

As with the previous program try modifying it by changing the message and by adding new lines.

## 2.3 Robot Hello, World!

This section covers the robot equivalent of Hello, World! To make the program work you will need to build a robot with left and right motors. A good choice is the basic robot from Carnegie Mellon, build instructions available here: $http://www.education.rec.ri.cmu.edu/content/lego/building/build_shows/rem$

Create a new *LeJOS NXT* project and main class following the instructions from the previous section. Once the project and main class are created, add the following lines:

```
Motor.A.forward();
Motor.B.forward();
try {
        Thread.sleep(1000);
} catch(InterruptedException ex) {
        Thread.currentThread().interrupt();
}
Motor.A.backward();
Motor.B.backward();
try {
        Thread.sleep(1000);
} catch(InterruptedException ex) {
        Thread.currentThread().interrupt();
}
```

Now run the program. Remember it should be run as a LeJOS NXT Project (see previous section). You should see the robot move forward for a second and then move backwards for a second.

The commands like: `Motor.A.forward()` tell the motor connected to port `A` (or `B`) on the brick to go forward (or backward). The block of code:

```
try {
        Thread.sleep(1000);
} catch(InterruptedException ex) {
        Thread.currentThread().interrupt();
}
```

"tries" to make the current thread (this part of the program) sleep (i.e. pause) for 1000 milliseconds – 1 second. Because the motors are already turning, either forwards or backwards, they continue to turn for the second. Because the sleep command may cause an *exception* it has to be embedded

in a try-catch block.[2] These try-catch blocks will have to be used for a few different commands, but there will always be an example to work from.

Now that the basic program is working try modifying it. Here are a few things to try:

1. Change the sleep values so the robot moves for longer.
2. Add another set of forward and backward commands so that the robot goes back and forth another time. (Later loops will be introduced that will make it much easier for the robot to do repetitive commands.)
3. If one motor turns forwards and the other motor turns backwards (or turns at a different speed) the robot will turn instead of moving in a straight line. Try making the robot turn left, then right.

[2] An exception is simply a special type of error that can occur when a program is running. For now we won't worry about them.

# Index