

Introduction to Lego NXT Robot and LeJOS

John Kelleher

DIT

Overview

1 **NXT Robot**

- Programmable Brick
- Sensors
- Actuators
- IO

2 **Programming Robots**

3 **leJOS**

- Introduction to Java
- The Java Compilation Process
- Intro to Object Oriented Programming
- Your first java program: HelloWorld.java
- Introduction to leJOS
- Compiling and running your first leJOS program
- helloworld.java for nxt

- Our companion during the course will be the *Lego Mindstorms NXT* robot.



The NXT Brick

48MHz CPU



1/12th



1/67th

- **NXT Programmable Brick:** This is the *brains* of the robot. It contains a 32-bit processor with 4 input ports and 3 output ports, a matrix display, a speaker, USB and bluetooth.

The NXT Brick

256 KBytes HD



1/100,000th



1/1,000,000th

- **NXT Programmable Brick:** This is the *brains* of the robot. It contains a 32-bit processor with 4 input ports and 3 output ports, a matrix display, a speaker, USB and bluetooth.

The NXT Brick

64 KBytes RAM



1/100,000th



1/200,000th

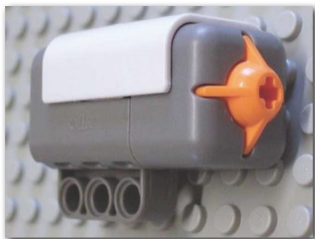
- **NXT Programmable Brick:** This is the *brains* of the robot. It contains a 32-bit processor with 4 input ports and 3 output ports, a matrix display, a speaker, USB and bluetooth.

Sensors

4 x sensor ports
referred to by number



Touch Sensor

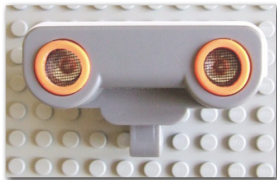


Measures touch

Digital sensor (i.e. only on or off)

- **Touch sensor:** this reacts to touch or release. We use this sensor to check if the robot has bumped into anything.

Ultrasonic Sensor

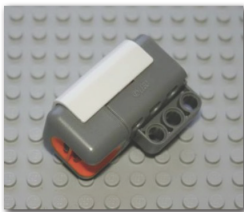


Measures distance in cm using reflected ultrasound

Noisy! Range 0cm to 255cm, +/- 3cm

- **Ultrasonic sensor**: using this sensor the robot can find out the distance between it and an object that is close by. This type of sensor is often called a **range sensor**

Light Sensor



Measures either ambient or reflected light

Returns percentage of calibrated range

- **Light sensor:** this sensor detects different colours and light intensity. We would use this sensor if we want to get the robot to follow a particular coloured line.

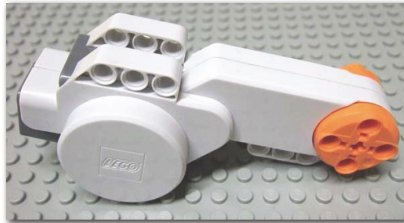
- **Sound sensor**: this sensor can detect loud sounds in the vicinity of the robot, such as a clap.

Motors

3 x motor ports
referred to by letter



Motors



Servo motor with integrated rotation encoder

Encoder allows accurate movements

Bluetooth



Wireless communication with PC, phone or other NXT

Communication must be between paired devices

The Rest

USB port

LCD screen

Buttons



- Robots are made to perform useful tasks. Each one is designed to solve a specific problem, in a specific way.
- Creating a successful robot takes a team effort between humans and machines.

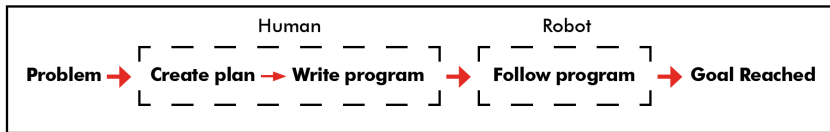
Role of the Programmer

The human is responsible for identifying the task, planning out a solution, and then explaining to the robot what it needs to do to reach the goal.

Role of the Robot

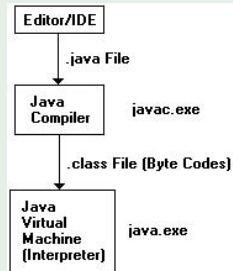
The machine is responsible for following the instructions it is given, and thereby carrying out the plan.

- A special programming language must be used to translate the necessary instructions from human to robot.
- There are many such languages: in this class we will use the with **Lejos** programming language.
- This language is **similar** to the Java programming language you have already learned.



- is a tiny Java Virtual Machine
- with a subset of classes you usually use
- and a slightly different approach to deploying code than standard Java

Compilation Process



Commands

- 1 **javac Filename.java** Compiles the source file **Filename.java** into bytecode, creating a file **Filename.class**.
- 2 **java Filename** Runs the bytecode in the file **Filename.class** on the java virtual machine.

Java is Object Oriented

- All code is organised in **objects**. Software objects consist of internal states variables and methods (functions) that operate on the internal state variables.
- A **class** is the blueprint from which individual objects are created. This means that all Java code is embedded within class data structures.
- You create Java objects in **instantiating** (making an instance of) a Java class.
- A variable in Java can use a class as its data-type. This means that once you create an object (i.e. an instance of a Java class) you can store that object in a variable that uses the same class as its data-type.

A Basic Java Class

```
1 class Box {  
2   double width;  
3   double height;  
4   double depth;  
5 }
```

Creating and Object of type Box

```
1 //create a box object called mybox using the Box class  
2 Box mybox = new Box();
```

HelloWorld.java Source Code

```
1 public class HelloWorld {  
2     public static void main (String[] args) {  
3         System.out.println ("Hello World");  
4     }  
5 }
```

Java Class Definition

- The code in the example box above defines a simple Java program.
- Line 1 and 5 in the above program name and delimit the extent of the HelloWorld class.
- As you can see all of the program is inside the class HelloWorld

HelloWorld.java Source Code

```
1 public class HelloWorld {  
2     public static void main (String[] args) {  
3         System.out.println("Hello World");  
4     }  
5 }
```

main Method

- Once we run a program the Java Virtual Machine needs to know where the program starts.
- Every java program must contain a **main method**. When a java program is run Java looks for the main method and starts executing from there. Line 2 is the standard way of defining a main method in Java. The curly bracket on line 4 marks the end of the main method.

HelloWorld.java Source Code

```
1 public class HelloWorld {  
2     public static void main (String[] args) {  
3         System.out.println("Hello World");  
4     }  
5 }
```

The Java dot Notation

- **Dot notation** is a way to refer to an object's instance variables and methods using a dot (.) operator
objectname.variablename
- E.g., if you have an object named `myCustomer` that has an internal variable called `orderTotal`, you refer to that variable's value as in this statement:
`float total = myCustomer.orderTotal;`

HelloWorld.java Source Code

```
1 public class HelloWorld {  
2     public static void main (String[] args) {  
3         System.out.println ("Hello World");  
4     }  
5 }
```

The System Class

- In Java the **System** class provides a set of methods that the user can invoke when they want to interact with the Java operating system. By default, one object of type System is created for each Java program.
- For example, using some of its methods, you can obtain the current time and the settings of various properties associated with the system.

HelloWorld.java Source Code

```
1 public class HelloWorld {  
2     public static void main (String[] args) {  
3         System.out.println("Hello World");  
4     }  
5 }
```

The System Class - Predefined Streams

- System also contains three predefined system variables, **in**, **out**, and **err**.
- **System.in** refers to the standard in; by default this is the keyboard.
- **System.out** refers to the standard output; **System.err** refers to the standard error stream; by default both of these are the console.

HelloWorld.java Source Code

```
1 public class HelloWorld {  
2     public static void main (String[] args) {  
3         System.out.println("Hello World");  
4     }  
5 }
```

The System Class - Predefined Streams

- **System.in** is an object of type **InputStream**
- **System.out** and **System.err** are objects of type **PrintStream**. The **PrintStream** class defines the **println** method which outputs a string to the screen.

HelloWorld.java Source Code

```
1 public class HelloWorld {  
2     public static void main (String[] args) {  
3         System.out.println ("Hello World");  
4     }  
5 }
```

System.out.println("Hello World");

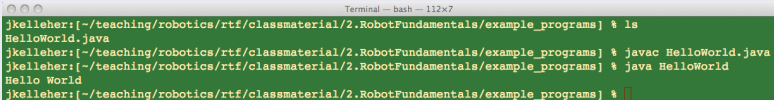
So, line 3 of the program `System.out.println("Hello World");` invokes the `println` method from the `PrintStream` class through the instance of this class that is stored in the `out` member variable of the `System` object that is created by default when the program using the `System` class as its template. This results in the message `Hello World` being printed to the console.

Your first java program: HelloWorld.java

HelloWorld.java Source Code

```
1 public class HelloWorld {  
2     public static void main (String[] args) {  
3         System.out.println("Hello World");  
4     }  
5 }
```

Compiling and Running HelloWorld.java



Terminal — bash — 112x7

```
jkmaleher:[~/teaching/robotics/rtf/classmaterial/2.RobotFundamentals/example_programs] % ls  
HelloWorld.java  
jkmaleher:[~/teaching/robotics/rtf/classmaterial/2.RobotFundamentals/example_programs] % javac HelloWorld.java  
jkmaleher:[~/teaching/robotics/rtf/classmaterial/2.RobotFundamentals/example_programs] % java HelloWorld  
Hello World  
jkmaleher:[~/teaching/robotics/rtf/classmaterial/2.RobotFundamentals/example_programs] %
```

What is leJOS NXJ?

- is a Java Virtual Machine for the LEGO MINDSTORMS NXT ". It allows you to program LEGO " robots in Java.
- includes all the classes in the NXJ API as well as tools used to upload code to the NXT brick.
- is freely available at: lejos.sourceforge.net

leJOS setup

- In order to run leJOS NXJ on your NXT robot you need to replace the NXT firmware with the leJOS NXJ firmware and set up your programming environment.
- See:
lejos.sourceforge.net/nxt/nxj/tutorial/Preliminaries/GettingStarted.

- 1 **compile**: You compile NXJ code with the standard compiler. But the leJOS VM does not support loading, so all classes must be linked in.
- 2 **link**: The linker takes your main class and processes dependencies to produce an nxj binary. The binary must be uploaded to the NXT
- 3 **upload**: You upload the your binary to the NXT using the USB connectivity.
- 4 **run**: Once the binary is on the NXY you can invoke it using the buttons on the robot.

nxjc and nxj Tools

- Compile the program with the **nxjc** command
- Then link, upload and run it with the **nxj** command.

HelloWorld.java Source Code

```
1 public class HelloWorld {  
2     public static void main (String[] args) {  
3         System.out.println("Hello World");  
4     }  
5 }
```

leJOS and Java

leJOS is an extension of Java. As a result, leJOS programs look very like Java programs. For example, the Java program **HelloWorld.java** program we looked at earlier will run in leJOS.

HelloWorld.java Source Code

```
1 public class HelloWorld {  
2     public static void main (String[] args) {  
3         System.out.println ("Hello World");  
4     }  
5 }
```

leJOS and Java

Some points to note:

- 1 leJOS requires the standard main method for the program entry point and leJOS supports the standard java System.out.println method
- 2 If you run this program as it is on the NXT, it will display Hello World and then immediately return to the menu, so you will not be able to see what is displayed (unless you are very quick).

Updated HelloWorld.java Source Code for NXT

```
1 public class HelloWorld {  
2     public static void main (String[] args) {  
3         System.out.println("Hello World");  
4     }  
5 }
```

leJOS and Java

We either need the program to sleep for a while to allow the text to be read, or to wait for a button to be pressed. Let us wait for a button to be pressed.

Updated HelloWorld.java Source Code for NXT

```
1 import lejos.nxt.*;
2
3     public class HelloWorldNxt {
4         public static void main (String[] args) {
5             System.out.println("Hello World");
6             Button.waitForPress();
7         }
8     }
```

Importing Packages and Classes

- To do this we need to **import** the leJOS NXJ Button class in the program.
- Importing a class or package into a program allows you to use the class (or classes in the package) in your program.

Updated HelloWorld.java Source Code for NXT

```
1 import lejos.nxt.*;
2
3     public class HelloWorldNxt {
4         public static void main (String[] args) {
5             System.out.println ("Hello World");
6             Button.waitForPress();
7         }
8     }
```

Importing Packages and Classes

- We find out which class we need to include by looking at the leJOS NXJ API online and finding a method (function) that does what we want. We then include the class that the method is a member of.

Updated HelloWorld.java Source Code for NXT

```
1 import lejos.nxt.*;
2
3     public class HelloWorldNxt {
4         public static void main (String[] args) {
5             System.out.println("Hello World");
6             Button.waitForPress();
7         }
8     }
```

Importing Packages and Classes

- If we look at the leJOS API we will see that the method **waitForPress()** that waits for any button to be pressed is in the **Button** class which is in the **lejos.nxt** package.
- So we can either include **lejos.nxt.Button** or **lejos.nxt.*** to allow any of the standard lejos.nxt classes to be used in the program.

Updated HelloWorld.java Source Code for NXT

```
1 import lejos.nxt.*;
2
3     public class HelloWorldNxt {
4         public static void main (String[] args) {
5             System.out.println("Hello World");
6             Button.waitForPress();
7         }
8     }
```

Compiling and Running

- Type in the above code into notepad and save the file as HelloWorld.java

Updated HelloWorld.java Source Code for NXT

```
1 import lejos.nxt.*;
2
3     public class HelloWorldNxt {
4         public static void main (String[] args) {
5             System.out.println("Hello World");
6             Button.waitForPress();
7         }
8     }
```

Compiling and Running

- Open a terminal and navigate to the directory you saved HelloWorld.java in.
- To compile your code type: **nxjc HelloWorld.java**

Updated HelloWorld.java Source Code for NXT

```
1 import lejos.nxt.*;
2
3     public class HelloWorldNxt {
4         public static void main (String[] args) {
5             System.out.println("Hello World");
6             Button.waitForPress();
7         }
8     }
```

Compiling and Running

- Now make sure that you NXT brick is connected to your computer via the USB cable and is turned on.
- You can upload your program to the brick with the command: **nxj -u HelloWorld**

1 **NXT Robot**

- Programmable Brick
- Sensors
- Actuators
- IO

2 **Programming Robots**

3 **leJOS**

- Introduction to Java
- The Java Compilation Process
- Intro to Object Oriented Programming
- Your first java program: HelloWorld.java
- Introduction to leJOS
- Compiling and running your first leJOS program
- helloworld.java for nxt