# Robotics Essential Information (AY 21/22)

David A Cohen, Nicola Paoletti

In this document we show you how to use Eclipse for robot programming in Java. Read the **whole** document carefully and keep it handy throughout the course. *Technical words are sometimes hyperlinked to a glossary of terms at the end of this document.*

## Getting Started

Robot EV3 programming can be simple in Java using a set of free libraries called the EV3 API. An Integrated Development Environment (IDE) helps you write code. It marks errors in your code and suggests fixes. It can tidy your code, fix your imports and rename variables. We have configured Eclipse, a Java IDE, to play nicely with the EV3 API libraries.

### Working arrangements

You will be working on the Lego EV3 in person in the Bedford labs. The robot is kept in your robot locker. You arrange to meet several times a week in the lab to work on your robot code. If one team member can't attend in person, they can send their compiled robot code (a jar file) to the members who have physical access to the robots. They will then upload the code into the Lego EV3 and test it.

### Creating an Eclipse project for programming EV3 Robots

Eclipse and Java need to know what is in the EV3 API. You achieve this by using a specific Eclipse project. To create a leJOS EV3 project,

1. In Eclipse, `File->New ->Other...-> leJOS EV3 -> leJOS EV3 Project`.
2. Give it a memorable name like `leJOS Project` in the Project Name box.
3. In the JRE box, tick `Use an execution environment JRE`, and set it to JavaSE-1.7. **Do not skip this step otherwise your program won't run on the brick.**
4. Press the Finish button.
5. Set up the ant building instructions:
   (a) Click the project name in Eclipse and select `New->File`. Name the new file `build.xml` and open it.
   (b) Copy the contents of the skeleton `build.xml` file found on Page 4 into your file.

**Create a new leJOS EV3 project for each worksheet or robotics program that you write.**

### Configuration on own PC

If you wish to work on your own PC, before step 1 above you first need to:

1. Download LeJOS at sourceforge.net/projects/lejos and install it on your computer (see sourceforge.net/p/lejos/wiki/Windows Installation for instructions on how to install on Windows[1])

2. Download and install Eclipse (www.eclipse.org/downloads)

3. Install and configure the LeJOS Eclipse plugin, see instructions at this link.

A video tutorial on setting up your own PC is available on Moodle.

## Build, upload and run your program

1. To generate the executable `.jar` file in Eclipse, right-click the `build.xml` file and click `Run As -> Ant build`. The `.jar` file will be under the folder `dist` of your project (you might have to refresh the Eclipse project if the folder doesn't show up).

2. Upload the executable into the brick using the EV3 Control app:

   (a) Make sure the robot is switched on (it must be switched on before attaching it to the laptop/PC)

   (b) Make sure that the robot is connected to the laptop/PC, either with USB or Bluetooth.

   (c) Run the EV3 Control program:
   - In the Lego lab or own PC: in Eclipse, select the leJOS EV3 menu and choose EV3 Control, or
   - In own PC: directly run the `ev3control` executable found in the `bin` directory of your leJOS installation.

   (d) Establish a connection: enter the IP address `10.0.1.1`[2] in the 'Name:' field, and press 'Connect'.

3. Move to the 'Programs' tab of EV3 Control and upload the generated `.jar` file

4. Select the uploaded program and click 'Run Program'.

A video tutorial on uploading and running your program on EV3 Control is available on Moodle.

**One-click run:** Alternatively, in Eclipse you can directly select the class containing the main method, right-click, `Run as -> LeJOS EV3 program`. This automatically builds your Java source, uploads it into the brick and runs it. We tend to prefer manual upload though as it provides better control and facilitates sending `.jar` files remotely if needed.

## Connecting via Bluetooth (instead of USB)

1. You must **rename your robot** using the 'Change Friendly Name' box on the 'Tools' tab in EV3 Control.

2. Pair the EV3 robot to your laptop/PC (can be done on both the Robot menu or EV3 control app)

3. Disconnect USB and connect EV3 Control using Bluetooth.

See also the Bluetooth video tutorial on Moodle.

[1]You don't have to flash the SD card, so no need to run the EV3SDCard utility at the end of the installation.
[2]The IP address is also displayed on top of the main LeJOS screen.

## If the robot program crashes and the screen looks weird

Read the error message ('Console' Tab) – it is just like you get on a PC when a Java program crashes. To clear the error, use the robot's Back key, or the 'Stop Program' button in EV3 Control.

## Stuff you Really Need to Know!

- Make sure the brick switched on when you are trying to upload a program!
- The dark grey button is in the middle of the robot is the ENTER button.
- The grey button just below the bottom left of the LCD screen is the ESCAPE/BACK button.
- The other four buttons are UP, DOWN, LEFT and RIGHT for navigating the leJOS menus.
- The BACK interrupts a running program, or exits a menu screen.
- Do Not Ever Reboot the brick by removing the battery. This can brick the robot.
- Pressing BACK together with the ENTER button and waiting will (eventually) reboot the EV3.

## Code Cleanliness - Knowing what you are doing with a robot

- Keep older, working versions of the code safe.
- Is the brick running the code that you think it is running?
- Put a `printVersion()` method in each class file.

  ```
  public void printVersion() {
      LCD.drawString("V2 - author", 0, 0);
  }
  ```

  Change the version number (V3, V4, V4.1, V5...) from time to time, and use your name, not author :). In your main method print out all of the version strings to the EV3 Control Console.

| **Key Combination** | **Effect** |
|---|---|
| TAB / Shift TAB | Increase / Decrease the indent of selected text |
| Ctrl+/ | Comment / uncomment a line or selection |
| Ctrl+I | Correct indentation of selection |
| Ctrl+Shift+F | Auto-format the file (prettifies everything) |
| Alt+Shift+R | Rename (refactor) selected element (variable/class) and all references |
| Ctrl+Shift+O | Organise Imports (add needed or remove unnecessary packages) |
| Ctrl+Q | Jump to last edited (changed) location |
| Ctrl+Shift+P | Jump to matching bracket (when your cursor is on a bracket) |

## Other sources of Information

- The Internet has many tutorials, EV3 robot ideas and example programs.
- The leJOS Wiki is a great place to start.
- The leJOS news page has much (some?) interesting material.
- Browse the EV3 API web page to discover (more) helpful classes and methods.
- **Ask questions on Moodle.**

## Skeleton `build.xml`

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!-- Change the next property to give your jar file a (more) sensible name -->
<project basedir="." default="jar" name="CHANGE ME">
    <!-- Change the next property to be the name of your program class that contains the `main` method -->
    <property name="main-class" value="CHANGE ME" />
    <property environment="env" />
    <property name="source.dir" value="src" />
    <property name="lib.dir" value="${env.EV3_HOME}/lib/ev3" />
    <property name="class.dir" value="build" />
    <property name="jar.dir" value="dist" />
    <property name="lejos.home" value="${env.EV3_HOME/}" />
    <property environment="env" />
    <property name="debuglevel" value="source,lines,vars" />
    <property name="ant.build.javac.target" value="1.7" />
    <property name="ant.build.javac.source" value="1.7" />
    <path id="libraries.path">
        <fileset dir="${lib.dir}">
        <include name="*.jar" />
        </fileset>
    </path>
    <target name="clean" description="delete old files">
        <delete dir="${class.dir}" />
        <delete dir="${jar.dir}" />
    </target>
    <target name="compile" description="build class files" depends="clean">
        <mkdir dir="${class.dir}" />
        <javac source="7" srcdir="${source.dir}" destdir="${class.dir}" includeantruntime="false">
            <classpath refid="libraries.path" />
        </javac>
    </target>
    <target name="jar" depends="compile">
        <mkdir dir="${jar.dir}" />
        <jar destfile="${jar.dir}/${ant.project.name}.jar" basedir="${class.dir}">
            <manifest>
            <attribute name="Main-Class" value="${main-class}" />
            <attribute name="Class-Path" value="${lejos.home}/lib/ev3classes.jar ${lejos.home}/libjna/usr/share/java/jna.jar" />
            </manifest>
        </jar>
    </target>
</project>
```

### Troubleshooting on building Ant file

- **Do not forget to edit the `CHANGE ME` values.**
- In some systems/versions, the environmental variable `${env.EV3_HOME}` for the location of the LeJOS installation might be named differently, e.g., `${ev3.home}`. If so, you Ant build will fail. This can be solved by checking the list of env variables in Eclipse (open a new run configuration, switch to the 'Arguments' tab and press the 'Variables' button) and replacing all occurrences of `${env.EV3_HOME}` in the above skeleton with the correct name.
- If you get a `ClassNotFoundException` is most likely because you entered the wrong class name under the `"main-class"` property of the `build.xml` file. Common mistakes are:
  - Putting the `.java` suffix (it's not needed)
  - Entering the wrong classpath: if your class `MyClass.java` is inside a package `mypackage`, then the `"main-class"` property must be set to `mypackage.MyClass`

# Glossary

**Ant** A program that builds and deploys a project using commands contains in a file called `build.xml`.

**Command Window** A command window is used to process commands on files in a particular folder. When you start a command window it you are in your home folder. Commands you type will act on the files in your home folder. Do not type `DEL *.*` You can use the `DIR` command to list the contents of the folder your command window is currently working on and the command `CD foldername` to move into a folder. Use the `PWD` command to tell you which folder you are working in at any time. The special command `CD ..` moves up to the parent folder (which contains the folder you were in). You can use the up arrow on your keyboard to get previous commands back. Then you can edit them if necessary and press return to run them again. It is very helpful that the command window completes file names. Just press the TAB key when you have started to type in a file name.

**Compile** Compiling is the process of turning your human-readable Java file (source file) into a (shorter) version that the Java Virtual Machine (JVM) can understand. The JVM then reads this shorter (bytecode) version as a sequence of instructions for the machine (e.g Lego brick) it is running on. On a PC these would be instructions like "read a file called `fred.txt`". On a robot you might usefully have instructions to make a motor start turning. These compiled Java files can be put together into a Jar file – much like a zip file – to make them easier to upload to the brick.

**Eclipse** Eclipse is an IDE. It is a program that helps you to write Java programs correctly and easily. In the Robotics Lab we have set up Eclipse to help you to write Java programs for your Lego EV3 robots using the EV3 API. There is a short "follow me" video on Moodle to help you install Eclipse on your home Windows machine.

**EV3 API** An Applications Programmer Interface (API) is a set of extra commands and objects added to a computer language to make it usable in a particular application. For example leJOS EV3 has added the `LCD` class to the EV3 API for robotics .

**Home folder** Every user on a system has a special folder where they store all of their files and folders. Only they can even read files in this folder. This is called their home folder and we use the uppercase HOME to refer to it. On Linux it is often `/home/username` and on Windows something like `H:\users\username`. At college these folders are shared between both systems so changes you make on one system are seen on the other. At home you are unlikely to have more than one disk, so your HOME folder will be on the `C:` drive.

**IDE** A program that does everything a programmer could need (as part of a team). It helps you code, compile, deploy, store, organise and run code.

**Java** Java is a modern programming language. Java is easily extended by writing libraries of functions that you might want to use. Such functions as "make the Motor plugged into port A go forward" for a Lego EV3 Java program would be useful and so are included in the EV3 API. Java has versions. Newer Java versions support nice new modern language features. **Careful: The JVM on the EV3 brick only supports Java 1.7**. The current Java version is at least 1.15.

**JVM** A program that reads Java bytecode and executes the desires of the programmer.

**Lego EV3** LEGO who make children's brick construction sets, together with MIT (Massachusetts Institute of Technology), built a brick that is a fully working computer. The Lego EV3 (linux based computer) brick can read sensor values and control motors. We can control this brick using Java by installing a system called leJOS EV3 onto the brick.

**leJOS EV3** leJOS EV3 is a system making Java available for programming Lego EV3 robots. It includes a standard Version 1.7 JVM. We will be programming using the libraries of functions, the EV3 API, that leJOS provides. We will also be using the tools provided by leJOS EV3 on the PC, and the menu system provided by leJOS EV3 on the Lego EV3 brick.

**leJOS EV3 project** Eclipse assumes that you are working on several projects and indeed you are. You will be coding several projects for CS1822 Robotics. Eclipse keeps all the files for each project separate in their own folder. In this way Eclipse can keep your preferences for each project separate. You can have Java projects for the desktop environment, for Android devices and for Lego EV3 robots. Each project for your robot needs to be a leJOS EV3 project. Choose the Project context menu, leJOS EV3 entry and select "Convert to leJOS EV3 project". Of course you need only do this once for each project. It is up to you how to organise your programs into projects. It is perhaps simplest to have a new leJOS EV3 project for each program that you write.

**Upload** Uploading is the process of getting a Java project onto the EV3. We need to (compile the Java files and) put the resulting file (which is a `.jar` file) onto the Lego EV3 brick so that you can run the program to see how the robot is made to behave. If you are working separately then you can upload the file to the brick by sending it to your colleague with the brick using Teams and watching it perform using their webcam. If you are brave then you can set up the SoftEther system to allow you to control the brick over the internet.

**Workspace** Eclipse stores each Java project in its own folder inside a special workspace folder. The workspace folder can be anywhere, but typically it is in a folder called `eclipse-workspace` in your home folder. Right click and use the `File->Properties` on an leJOS EV3 project to find out where your workspace is.