ETSA03-ADM-LAB1

Lab 1
Getting started with Robocode

Version 1.4 approved

Prepared by Markus Borg
Dept. of Computer Science, Lund University

March 24, 2020

Revision History

Name	Date	Reason For Changes	Version
Markus Borg	2017-12-08	Initial draft.	0.1
Markus Borg	2017-12-09	Lab scope completed and tested.	0.2
Markus Borg	2018-03-07	Complete draft based on Robocode ReadMe.	0.3
Markus Borg	2018-03-20	Updated after internal review.	1.0
Markus Borg	2019-03-04	Updated version for 2019.	1.1
Markus Borg	2019-03-26	Minor language improvements.	1.2
Mathias Haage	2020-03-18	Minor updates for 2020.	1.3
Markus Borg	2020-03-24	Minor clarifications.	1.4

1 Introduction

Lab 1 will help you get started with Robocode and robot development. More specifically, Lab 1 covers:

- Installation and configuration of a non-trivial software framework.
- The fundamentals of Robocode.
- Implementation of your first Robocode robot.
- A demonstration of team battles.
- Using Eclipse for Robocode robot development.

The instructions for Lab 1 are almost exclusively copied from the "ReadMe for Robocode" by Flemming N. Larsen from February 27, 2013 – available as open source under Eclipse Public Licence v1.0 at http://robocode.sourceforge.net/docs/ReadMe.html as well as content from the RoboWiki: http://robowiki.net/

2 Before the lab

To save time during the lab session, please read through this section and follow the instructions.

Have a quick look at the API to get a general feeling of what it is all about:
http://robocode.sourceforge.net/docs/robocode/

2.1 What is Robocode?

Robocode is a programming game where the goal is to code a robot battle tank to compete against other robots in a battle arena. So the name Robocode is a short for "Robot code". The player is the programmer of the robot, who will have no direct influence on the game. Instead, the player must write the AI of the robot telling it how to behave and react on events occurring in the battle arena. Battles are running in real-time and on-screen.

The motto of Robocode is: Build the best, destroy the rest!

Besides being a programming game, Robocode is used for learning how to program, primarily in the Java language, but other languages like C# and Scala are becoming popular as well.

Schools and universities are using Robocode as part of teaching how to program, but also for studying artificial intelligence (AI). The concept of Robocode is easy to understand, and a fun way to learn how to program.

Robocode offers complete development environment, and comes with its own installer, built-in robot editor and Java compiler. Robocode only pre-requires that a JVM (Java Virtual Machine) to exist already on the system where Robocode is going to be installed. Hence, everything a robot developer needs to get started is provided with the main Robocode distribution file (robocode-xxx-setup.jar). Robocode also supports developing robots using external IDEs like e.g. Eclipse, IntelliJ IDEA, NetBeans, Visual Studio etc., which supports the developer much better than the robot editor in Robocode.

The fact that Robocode runs on the Java platform makes it possible to run it on any operating system with Java pre-installed, meaning that it will be able to run on Windows, Linux, Mac OS, but also UNIX and variants of UNIX. Note that Java 6 or newer must be installed on the system before Robocode is able to run. See the System Requirements for more information.

Be aware that many users of Robocode (aka Robocoders) find Robocode to be very fun, but also very addictive. :-)

Robocode comes free of charge and is being developed as a spare-time project where no money is involved. The developers of Robocode are developing on Robocode because they think it is fun, and because they improve themselves as developers this way.

Robocode is an Open Source project, which means that all sources are open to everybody. In addition, Robocode is provided under the terms of EPL (Eclipse Public License).

2.2 History of Robocode

The Robocode game was originally started by Mathew A. Nelson as a personal endeavor in late 2000 and became a professional one when he brought it to IBM, in the form of an AlphaWorks download, in July 2001.

IBM was interested in Robocode, as they saw an opportunity to promote Robocode as a fun way to get started with learning how to program in Java. IBM wrote lots of articles about Robocode, e.g. like Rock 'em, sock 'em Robocode! from AlphaWorks / developerWorks at IBM, a series of articles like Secrets from the Robocode masters, and "Robocode Rumble / RoboLeague".

The inspiration for creating Robocode came from Robot Battle, a programming game written by Brad Schick in 1992, which should still be alive. Robot Battle was, in turn, inspired by RobotWar, an Apple II+ game from the early 1980s.

The articles from IBM and the Robocode community behind the RoboWiki made Robocode very popular as programming game, and for many years Robocode has been used for education and research at schools and universities all over the world.

In the beginning of 2005, Mathew convinced IBM to release Robocode as Open Source on SourceForge. At this point, the development of Robocode had somewhat stopped. The community around Robocode began to develop their own versions of Robocode with bug fixes and new features, e.g. the "Contributions for Open Source Robocode" and later on the two projects, RobocodeNG and Robocode 2006, by Flemming N. Larsen.

Eventually, Flemming took over the Robocode project at SourceForge as administrator and developer in July 2006 to continue the original Robocode game. The RobocodeNG project was dropped, but Robocode 2006 was merged into the official Robocode version 1.1 containing lots of improvements. Since then, lots of new versions of Robocode have been released with more and more features and contributions from the community.

In May 2007, the RoboRumble client got built into Robocode. RoboRumble is widely used by the Robocode community for creating up-to-date robot ranking lists for the 1-to-1, Melee, Team, and Twin Dual competitions.

Since May 2010 a .NET plugin is provided for Robocode using a .NET / Java bridge, which makes it possible to develop robots for .NET beside developing robots in Java. This part was made by Pavel Savara, who is a major Robocode contributor.

2.3 System Requirements

In order to run Robocode, Java 6 Standard Edition (SE) or a newer version of Java must be installed on your system. Both the Java Runtime Environment (JRE) and the Java Developer Kit (JDK) can be used. Note that the JRE does not include the standard Java compiler (javac), but the JDK does. However, Robocode comes with a built-in compiler (ECJ). Hence, it is sufficient running Robocode on the JRE.

Also note that it is important that these environment variables have been set up prior to running Robocode:

```
JAVA_HOME must be setup to point at the home directory for Java (JDK or JRE).
Windows example: JAVA\_HOME=C:\\Program Files\\Java\\jdk1.6.0\_41
UNIX, Linux, macOS example: JAVA\_HOME=/usr/local/jdk1.6.0\_41

PATH must include the path to the bin of the Java home directory
(JAVA_HOME) that includes java.exe for starting the Java virtual Machine (JVM).
Windows example: PATH=\%PATH\\%;\%JAVA_HOME\%
UNIX, Linux, macOS example: PATH=\${PATH}:\${JAVA_HOME}/bin
```

Java is already installed and configured on the computers in E:Alfa and E:Beta, but you might want to double check the environment variables if you plan to work on a private laptop.

2.4 Understanding the basics

To quickly grasp the fundamentals of Robocode battles, please read the following sections on the RoboWiki's "Beginner Guides".

http://robowiki.net/wiki/Robocode_Documentation

- Game physics
- The anatomy of a robot
- Scoring in Robocode
- Frequently asked questions

3 At the lab

This section describes what you should complete during the lab session.

3.1 Install Robocode

First things first – let's get Robocode up and running on your local machine.

1. Locate Robocode version 1.9.3.7 on SourceForge:
   `https://sourceforge.net/projects/robocode/files/robocode`
2. Download `robocode-1.9.3.7-setup.jar`
3. Install Robocode by running the jar-file. If needed, follow detailed instructions on RoboWiki:
   `http://robowiki.net/wiki/Robocode/Download_And_Install`
4. Start your command line interpretor and navigate to the Robocode installation folder. Start Robocode by executing the following command in the Robocode installation folder (feel free to copy-paste):
   `java -Xmx512M -cp libs/robocode.jar robocode.Robocode`
5. Explore Robocode by starting a few battles from the Battle menu. The robot SittingDuck is the standard robot used for the most basic target practice.

3.2 Develop your first robot using the Robot Editor

This is an adapted version of Robocode's classic "My First Robot Tutorial" that explains how to create your first robot. Detailed instructions are available at:
`http://robowiki.net/wiki/Robocode/My_First_Robot`

Creating a robot can be easy. Making your robot a winner is not. You can spend only a few minutes on it, or you can spend months and months. I'll warn you that writing a robot can be addictive! Once you get going, you'll watch your creation as it goes through growing pains, making mistakes and missing critical shots. But as you learn, you'll be able to teach your robot how to act and what to do, where to go, who to avoid, and where to fire. Should it hide in a corner, or jump into the fray?

Robocode comes with an internal code editor. It will not be used during the course, but it serves as a good way to get an first insight in robot implementation.

1. The first step is to open up the Robot Editor. From the main Robocode screen, click on the Robot menu, then select Source Editor. If Robocode detects that you already have a Java compiler installed on your system, accept using it.

2. When the editor window comes up, click on the File menu, then select New Robot.

3. In the dialogs that follow, type in a name for your robot, and enter a unique package name. During the project, your group should use the package name `groupXX` – you might just as well use it already now.

4. Investigate the code. The robot main loop in the `run()` method tells the robot to move back and forth and turn the gun. Three different events describe behavior when another robot is found, the robot is hit by a bullet, and the robot collides with a wall, respectively.

5. Save your robot by selecting the Save in the File menu. Follow the prompts to save your robot.

6. Compile the robot by selecting Compile in the Compiler menu. You will get to know the Codesize of your robot, and that it is categorized as a NanoBot. In some Robocode rumbles, the codesize is used the divide competing robots into weight classes. In the LU Rumble, you do not need to worry about this.

7. Close the Robot Editor and start a battle with your new robot.

8. Open the Robot Editor again and uncomment the parts about color (lines 3 and 21). Select some nice colors for your robot, save, and recompile your robot. Start a new battle and verify the new color scheme.

9. During the course, you will deliver robots packaged as jar-files using the Robot Packager. From the Robot menu, select Package robot or team. Add your new robot and click Next. In the next dialog, you can toggle whether or not to include source files – check the box when delivering to your project supervisor, but do not check it when delivering to your future customers. Enter a reasonable version number and a short description, and click Package! Verify that you get a corresponding jar-file in your file system.

3.3 Watch your first robot team battles

A robot team is a group of robots that fight together to beat an enemy team. You can make a team of as many robots as you want, but teams are usually made of 5 bots – in the LU Rumble, you will normally not be allowed to field more than 5 robots in a team! In Robocode, robots in a team can be of different classes, or they can be 5 instances of the same robot.

You implement team robots by extending the `TeamRobot` class that provides some functions to exchange information about robots and to check if a robot is your friend or

your foe. Robocode provides also a special interface called `Droid`, which allows you to implement a special kind of robot without radar, but with an extra 20 energy. When you create a team, the first robot is the "leader" and gets an extra 100 energy (200 total, or 220 for a Droid leader). Teams with droids have the advantage of having more energy, but suffer from lack of information and get in serious troubles if the leader gets killed. In the LU Rumble 2019, only the leader and the normal robots will count toward the maximum team size of 5 – you are free to add Droids to go beyond this limit.

Note that team play in the LU Rumble 2019 allows for a wide range of team types. For example:

- 1 team leader + 4 normal bots of the same class.
- 1 team leader + 4 normal bots of different classes.
- 1 team leader + 2 normal bots (if they are very expensive).
- 1 team leader + 6 droids (without radar, but with 20 extra energy, that receive orders from the leader).
- 1 team leader + 3 normal bots + 2 droids

Of course, the good part of teams is that they work together to win the battle. There are many different ways to make them work together, and for sure the way they coordinate their actions will make the difference on the battlefield. For example:

- The simplest way is a collusion where bots belonging to a team avoid shooting each other.
- A more advanced team will exchange information about themselves and the enemies to gather intelligence and plan their actions.
- Another step could be to coordinate the bots to avoid "colliding" (in a wide sense) and may be coordinating fire.
- A more advanced step should be to create real team strategies, for example flocking behavior.

Implementing a team introduces some problems that you would not find in 1v1 or melee battles. For example:

- Assigning the "good spots" to the bots (only one at each spot).
- Dealing with friendly fire and line-of-sight.
- Coordinating team movements.
- Identifying the bots on your own team.

If you have not tried team battles in Robocode yet, start a battle between Team:MyFirstTeam in the `sampleteam` package and a single SittingDuck robot. Watch how the team beats the sitting duck. Try starting a few more battles with different robots to study how the team performs. Also try a battle between two Team:MyFirstTeams. Can you think of any obvious improvements to the team behavior?

3.4 Getting started with Eclipse for robot development

During the course, you will use the Eclipse IDE rather than the Robot Editor. The RoboWiki has a great guide of how to get started with robot development using Eclipse. Please follow the instructions on the following sections available at:
`http://robowiki.net/wiki/Robocode/Eclipse`
Note that text in bold font below matches specific headers under "External Editors" on the link above.

1. **Creating a project in Eclipse**. Follow all instructions under this header.
2. **Creating a Robot in Eclipse**. Follow the instructions, and eventually copy the source code from the robot you created in Robot Editor. Import the packages required to make it compile.[1][2]
3. **Adding an Eclipse project to Robocode** Add the robot project into Robocode by following these instructions:
   `http://robowiki.net/wiki/Robocode/Add_a_Robot_Project` Note: Add the path to the Eclipse project (with `/src` and `/bin` as sub-folders), not the Eclipse workspace.
4. **Running your robot from Eclipse**. Follow all instructions under this header. Note that "Run As → Run..." might be called "Run As → Run Configurations" in your version of Eclipse.

—— REACH THIS POINT TO PASS LAB 1! ——

5. **Debugging your robot with Eclipse**. Follow all instructions under this header.[3]

[1] Please note that the newest versions of Eclipse will ask if you want to create a module-info.java file. If you get this question, you should select "Don't create". If you by mistake create a module-info.java, then you should make sure to delete it to avoid compiler and classpath issues. The module-info.java is not used for Robocode packages.

[2] In the Libraries tab, when adding robocode.jar as an external jar, there may be a choice to place it in modulepath or classpath. This is selected by clicking on respective path before clicking the add external jar button. Please make sure robocode.jar is placed below classpath.

[3] This is needed for the rest of the labs. If you do not do it now, you need to do it before next lab session.

4 After the lab

After having experimented with the fundamentals of Robocode, it is now time for you and your group to start thinking about features your robot shall offer the market. An important first step is to study the Robocode API to better understand what your future robot could potentially do: `http://robocode.sourceforge.net/docs/robocode/`

Before the next exercise session, your group has two hours scheduled to work in the project. This time does not have a room assigned, and also no teacher will be available. Use this time to gather the project group and start discussing what will be required by a robot team to win the LU Rumble. Try to align the group and come up with a high-level business plan. Good questions to discuss include:

- What type of robot shall your team offer to other groups? A leader, a droid, or a normal bot?
- What type of robots will be in demand?
- What could be a niche market?
- What will be commodity features in robots?
- What could be differentiating robot features?

Furthermore, as your group will not only robot supplier, but also a robot customer, think about what kind of robot you would like to buy. The time you spend together before the next exercise session involves all three main activities in the course, i.e., engineering, monetization, and strategizing.

Successful development of software products for an open market requires making critical decisions under time pressure – decisions of both technical nature as well as business nature. Your team soon needs to decide what type of robot to develop! At the next exercise, your group will start outlining the features you will implement in your robot.