Victoria University of Wellington
School of Engineering and Computer Science

SWEN221: Software Development

Lab Handout 4 (worth ≈1.3% of overall mark)

The purpose of this is to demonstrate by example the issues caused by poor encapsulation. You will first operate on a poorly encapsulation piece of code, making a series of changes that simulate typical things that arise once during the development and maintenance of a software application.

By the end of the lab, the tutor should have given you a grade based on your performance. **Be sure that you have been given a grade before you leave, since attendance at labs is mandatory**.

NOTE: it is also recommended that you submit your final lab code via the online submission system, which can be found on the SWEN221 homepage. This is useful in case your lab grade is lost, or you believe you have been given the incorrect grade.

1 Fixing RobotWar

A small software program, called Robot War, has been developed by the BadSoftware inc. development house. The first version of the game was released to the public some time ago, and several problems have been identified which must be fixed. You, as an employee of BadSoftware inc. are in the unenviable position of having to maintain this software project; that is, you need to fix the problems which the (very fustrated) users have identified.

You can download the `robotwar.jar` from the lecture schedule on the course home page. You should begin by running the software and getting a feel for how it works. As usual, you can run the program from the command-line like so:

```
java -jar robotwar.jar
```

Figure 1 shows a screenshot from RobotWar. Make the following changes to the Robot War software:

1. Fix the "Dead Robots Walking" problem. Sometimes it occurs that robots don't ever die. You can see this by running a game with 0 RandomBots and a number of GuardBots.
2. Fix the "Robots out of Arena" problem. Sometimes it occurs that robots can move outside the arena. Robots should never be able to move outside the arena!
3. Adjust the robots list to use an ArrayList. Currently, the robots list inside the Battle class uses a LinkedList implementation. However, since the game spends so much time iterating the robots list, it would be more efficient to use an ArrayList.

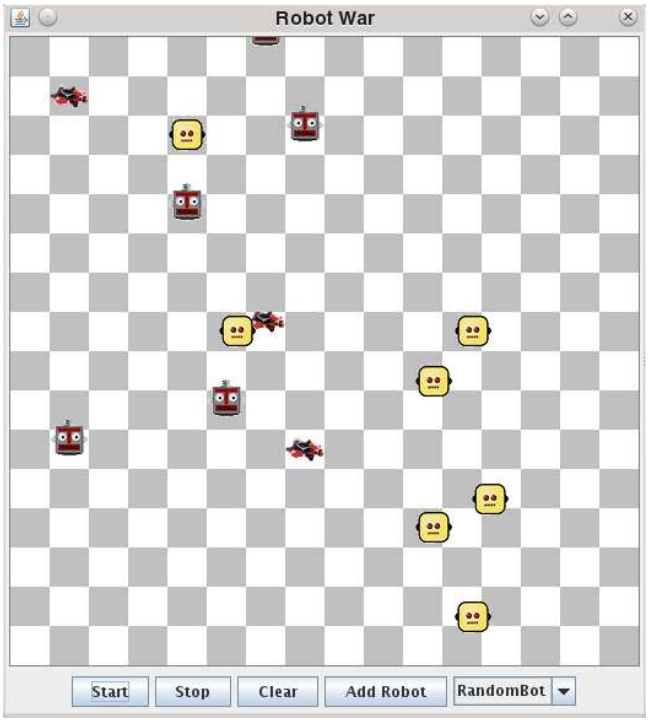

Figure 1: A screenshot from the robot war game.

4. Construct an "Alive Robots" list to improve performance. The game spends a lot of time iterating over the list of robots, but selecting only the robots which are alive. As an optimisation, you should add a second list to the Battle class called aliveRobots. This list should contain the list of currently alive robots. When a robot is destroyed, it should be removed from the list. Moreover, numerous loops which now iterate the robots list and select only alive robots should be updated to use the aliveRobots list.

2 Thinking about Encapsulation

Having now fixed various bugs in the RobotWar program, take a moment to reflect on the following issues:

1. **Scope of changes**. You should have found that the fixes you made in the previous activity typically required changing many files. Many of these fixes were related to changes in the implementation of a class. Good software design dictates that changes in implementation should be localised to the class in question. So, why was this not possible with Robot War?

2. **Encapsulating Robot War**. Many fields in Robot War are public. Following good software engineering practices, you want to change them to be private. How would you proceed to do this with Robot War?

3 Encapsulating RobotWar

You should now create a new Java project and import the `robotwar.jar` file again so as to begin this activity with the original version of the code.

Following your observations from previous activity, you should now *properly encapsulate the Robot War code by making all fields private.* When you have done this, apply the list of fixes from the first activity to your encapsulated version of Robot War. You should find that they are much easy to implement this time around, and don't require changes to multiple files.

Marking Guide

Each lab is worth just under 1% of your overall mark for SWEN221. The lab should be marked during the lab sessions, according to the following grade scale:

- **0**: Student didn't attend lab.
- **E**: Student did not really participate in the lab.
- **D**: Student's participation was *poor*. For example, he/she made some attempt to work on the lab, but did not complete any activities.
- **C**: Student's participation was *satisfactory*. That is, he/she completed at least one activity (e.g. fixed RobotWar).
- **B**: Student's participation was *good*. That is, he/she has made reasonable progress on activity 3.
- **A**: Student's participation was *excellent*. That is, he/she completed all activities.