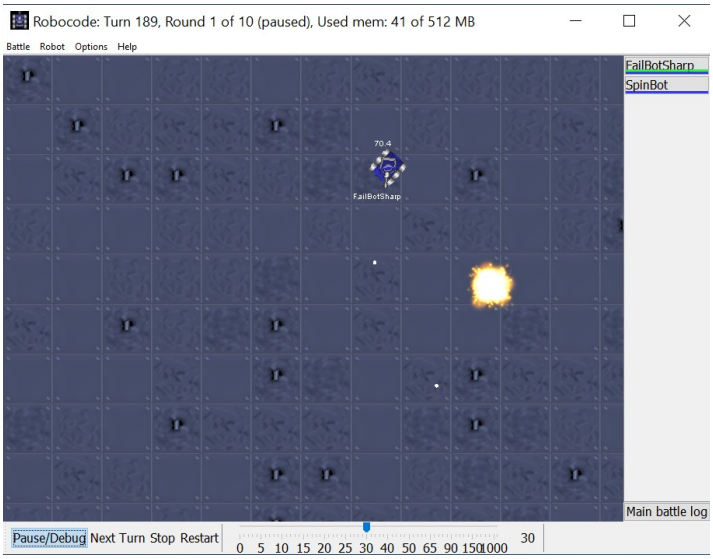
Manual: Robocode

Overview

Robocode is a competitive environment in which competitors implement custom tank robots which battle in an arena. In this project, students will design, architect, and implement a custom decision-making AI that will act as part of a team of autonomous agents. Students will implement a single software robot from the ground up that functions in the Robocode environment. Robocode robots are tanks with radar and a single turret. Your goal will be to match or exceed the performance (based on score as computed by the environment) of the robot team provided to you with the project materials, affectionately known as **TeamSocket** (composed of four **PassBot, FailBot, and FailBotOg** instances). Robots will have the following specification **which students must follow**:



Package: `student`
JAR File: **`[TeamName].jar`** ***(alphanumerical)***
Class: **`[BotName]`**
Filename: **`[BotName].java`** ***(Only one file!)***
Extends: `robocode.TeamRobot`

The Robocode platform requires the Java Runtime Environment (JRE); we will use Java 21.

Development

The most straightforward way to develop a Robocode robot that meets the specification is to use the provided VS Code project workspace, which incorporates a sample student robot file together with several Robocode examples and the benchmark robots; this environment has been tailored to directly run Robocode but allow for development within the IDE. A full specification of the API is detailed in javadoc folder (HTML) included with this manual. Students **must implement** a robust, extensible, abstracted decision-making architecture (Decision Tree, State Machine, Behavior Tree, or similar structure). The robot and all its dependent classes must be contained in a single Java source file. Students may use nested classes. Team names must not have spaces and must be exclusively alphanumerical characters.

Most students will find it easiest to develop in VS Code after installing the official Microsoft "**Extension Pack for Java**". (It notably *does not include the Oracle Java Extension;* this is not an accident.) Make sure you rebuild the project before testing, as the built-in change detection is not completely reliable.

To work within VS Code after installing the extensions, open the `Project` folder from the VS Code menu. To build the project, select your Java source file in the "**Explorer**" panel; the "**Java Projects**" listing should appear just under the file lists. At the top of this list (when selected), the "**Rebuild All**" item will rebuild the source. The project can then be run from the "**Run and Debug**" left side panel option (shaped like a "play" button).

Students must only use those resources defined in this document. Teamwork is for coordination and protocols, but not for the code of individual robots; each student must design and implement a unique robot to be part of a robot team. When students use outside resources, proper citation of strategies developed by others is ***crucial***. Failure to properly cite use of ideas developed and/or published by others will be considered academic dishonesty.

When a student uses one of the sanctioned resources in the development of his/her robot, the student will clearly cite the source of the strategy in source code and in their report.

UNDER NO CIRCUMSTANCES ARE STUDENTS PERMITTED TO COPY CODE FROM ANOTHER SOURCE. All implementations of code must be in "clean room" style – students may read about techniques but must write all code from scratch. Robots may only use data generated by the student or derived from references.

Tournament Rules

The tournament will adhere to the following rules:

- Robots submitted will be provided (compiled) to future classes.
- Students may submit at most one robot as an entrant into the tournament.
- All robots submitted for a grade must be submitted as entrants to the tournament.

Robots could be disqualified if they…

- Do not start due to bugs in their codes, or cause the Robocode environment to crash
- Use neural network weights / scripts / code generated or provided by a **third party**

However, an acceptable robot might still…

- Use training sets generated by a third party to train a neural network
- Dictate behavior based on a script rather than hard-coded instructions

Team Requirements

Each student must include exactly one robot on a team with other students. The follow rules apply:

- All teams must be 2-4 robots in size
- All teams must choose a team name and team color
- All robots in a team must use the same radar color and bullet color

Students will work together to prepare their team strategy. Students may not share any code to implement such a strategy, though they may develop a communication strategy and protocol code that can be shared.

Building a Tournament Team Package

Before the tournament each team must submit a single jar package containing all team robots, built as follows:

1. Add all robot source (java) files to the **StudentRobot** project in the **Robocode** solution.
2. Change StudentRobot's Assembly Name (Project → Properties → Application) to the team's name.
3. Clean and rebuild the project using VS Code. **DO NOT SKIP THIS STEP.**
4. Check that each robot functions in the Robocode environment.
5. Create a team with the robots (Robot → Create a robot team.)
6. From `env/Project/robots/`, zip up `TeamName.jar` together with the `student` folder.

Robot Scoring

This assignment consists of two parts: **robot performance** (for a grade) and the **tournament**.

Robot Performance

Each team's performance will be measured against the robot team provided as part of the assignment (TeamSocket) and will be graded as follows:

- All testing battles will be for 100 rounds on a battlefield of size 1200x1200
- Individual student robots will be posed against **FailBot** and teams against team **TeamSocket**
- Scores will be proportional, with robots scoring as well or better than benchmarks graded 100%

Tournament

The tournament battles consist of a team bracket on terrain as follows:

Rounds of Battle: 5
Battle Field Size: 1200x1200

Battle Ranking

Each round, robots will be awarded points according to survival, damage, and elimination of opponents. Robots / teams scoring 1st the team competition will advance to the next round.

Ties

In the event of a tie for a single battle, sudden death rounds will be added until clear winners emerge. In the event of a tie for the win, the tied competitors will participate in three rounds of battle on a 1200x1200 battlefield, with additional sudden death rounds as necessary to determine clear ranking.

Hall of Fame

The winning team members will be placed in a battle with current Hall of Fame robots. All robots will compete together in a Free-For-All battle on a 1200x1200 battlefield for 1000 rounds. The top ten robots will be placed in the Hall of Fame according to rank; the top four teams will be placed in the Team Hall of Fame. Ties will be broken using sudden death rounds (on 1200x1200) with all competitors until a clear order emerges.

Deliverables

Students will submit their **individual robots** with an **individual report** as well as a **team package** (see submission section) for the competition and a **team report**:

Individual Submission	Team Submission
Robot Source File (**java**) – ***Only one file!***	Team Package (**zip** – see below)
Robot Design Document (**pdf**)	Team Design and Post-Mortem (**pdf**)

Robot Design Document

The design document should focus on the AI's design, architecture, and resulting behavior and should be **no more than 1000 words**. (1000 is the limit, *not the requirement* – but make sure there's enough detail to understand your design!) This report should be completed by each student individually and should focus on the strategy and implementation of the student's robot, not including team-specific considerations. The design document should include figures to visualize the design of the student's robot and should identify precisely how ideas covered in class or in publications from respected venues were integrated into the agent.

Team Design and Post-Mortem Document

The team document should include team-specific design, architecture, and emergent behavior, as well as general strategy descriptions. It should be **no more than 2000 words**. (*2000 words is not required.*) The document should not retread information from the individual reports; instead, it should focus on how the robots coordinate as a team to employ a cohesive strategy.

In addition to the design descriptions, the report should also include descriptions of successes ("what went right") and failures ("what went wrong") throughout the development. This can include technical, social, or logistical challenges students faced in developing their team of robots. This post-mortem section should also include an individual reflection from each student on the project – how/if they learned from the project and how they might approach future work on teams.

Grading

Grading of the project will be as follows:

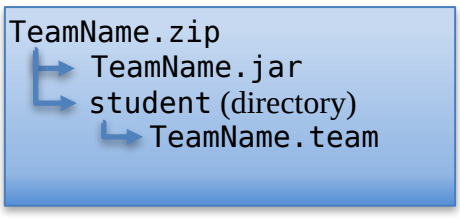
Grading Breakdown	
Criteria	**Portion**
Individual Robot Performance	30%
Robot Team Performance	30%
Robot Design Document	20%
Team Design & Post-Mortem Document	20%

Performance grades will be based on proportion as compared to the benchmarks:

Grading Examples		
Student Score	**Benchmark Score**	**Grade**
1600	1500	100%
1500	1500	100%
600	1500	40%

Submission

There are four submissions required for this project – one for each deliverable. All submissions will be via Canvas. The team package will be a zip file with the following structure:



Robot source files and reports will be submitted as separate files to the individual assignment, and the team package and team report will be submitted as separate files to the team assignment. For teams of students mixed between online and campus students, a team submission must be made in each section separately.