Lab 6: BugWars!

EEE320 - fall 2022

Introduction

In this lab you will use the BugBattle framework to build a creature that can out-compete and out-survive all your classmates' creatures. The source files for the BugBattle system (current version 1.13) are provided with this handout on Moodle.

Competition

The Bugwars! final competition will be held during the lecture periods on Monday, 5 December 2022. The competition will be a one-on-one, double-elimination tournament. After the main competition we can also run some melee rounds, if you'd like.

Submission

2 December 2022

You must submit at least one initial bug implementation to the Moodle lab 6 part 1 assignment prior to the lab period on December 2. During the lab, we will run competitions between bugs from different groups. This will enable you to gauge your bugs' performance against your peers without having to worry about other groups seeing your bug's secrets. There are no marks for this submission, but it would be a shame to miss it!

7 December 2021

You must submit your final bug implementation and your lab report to the the Moodle lab 6 part 2 assignment not later than 0800 hrs EST 7 December 2021. See the lab report requirements section below. Your bug implementation must include exactly one competition-ready main bug implementation plus any Propagators, supporting classes, and bug subclasses, conforming to the implementation constraints described below.

Ensure your submitted Python file includes your group members' names in the module comment and is named as described in the implementation constraints described below.

Setup

Download the source files and add them to a fresh PyCharm project. Ensure the project runs. The classes `Hunter` and `SuperPlant` provide examples of how one can write a bug within the framework.

Useful tip: In the PyCharm navigator pane, click the gear icon and select "show members". This will make it easier to navigate, particularly with the large number of classes in the `shared` module.

Tasks

Implementation

Your task is to design, implement, and document the world's smartest, most ferocious bug. A bug that beats all the other bugs and achieves World domination. An awesome bug!

Implementation Constraints

The entire implementation of your bug must be contained in a single `.py` file, including any `Propagator` subclasses, other helper classes, or cooperating creature classes. Your main `Creature` subclass must be the first class in this file. Name your file using your group member surnames, first letter of each surname in upper case, concatenated with no spaces. So, e.g., if your group members were Phillips and Lapointe, your file name would be `PhillipsLapointe.py`. Put this file in the `competitors` directory of your project.

You will need to update the `launch.py` file to let the framework find your competitor.

During your development phase you may have multiple `competitor` files to let you run different versions of your bugs against one another.

Your bug must be a subclass of `Creature` and must obey the requirements documented in the class comment for the `Creature` class. You must also observe all other constraints or requirements documented in other class comments.

Your bug must work within the spirit of the simulation world. Specifically, you may not access any private methods or attributes of framework classes, and you may not access any methods or attributes marked as "framework only" with a leading `f_` or `F_` in their name. If there is any doubt in your mind whether something might constitute cheating in the game, ask the instructor.

You should not modify anything outside the `competitors` module other than to make necessary changes to the `launch` module. (It would be pointless to do so anyway, since during the competition your bug will be executing in a "clean" version of the framework provided by the instructors.)

Lab Report Requirements

Your lab report must be readable and self explanatory, including the following sections:

a. An introduction and description of your bug and other code, to place your work in context for the reader,

b. A strategy section to describe how your bug operates and interacts with its environment, that is, its strategy for survival,

c. A description of the bug's implementation, including the use of any design patterns,

d. Appropriate diagrams documenting your design. As a minimum, you must include a class diagram that shows the complete structure of your bug, including any helper classes, etc. You must also include at least one sequence diagram illustrating the operation of the `doTurn()` method for your bug. If there are several alternative executions of `doTurn()` you should provide one diagram for each rather than trying to fit everything on a single diagram.

e. A description of your testing strategy for your bugs. How did you determine whether your bugs were working as intended?

f. A discussion of the use of patterns in the provided BugBattle framework. What patterns are used and where, are they appropriate, and are there other places in the framework where patterns could be usefully applied?

g. A detailed explanation of a new organ that you think would make the simulation more interesting; and

h. A conclusion section.

Mark breakdown

Out of a total of `125`, plus bonus:

- serious attempt at `Creature` implementation: `50`
- report text: `30`
- report diagrams: `20`
- general code quality: `25`
- competition bonus: `5` for finishing in top half; `10` for top 4, `25` for the winner