COMP 1406-Z

PROJECT REPORT 12/14

14 DECEMBER 2020 / 8:00 AM / ROOM -NA-

OOP PRINCIPLE APPLICATION AND ITS EFFECTS

Abstraction

- Abstraction was essential in the controller class, most specifically in the step method. By having the complicated code hidden within the simple-named methods (ie. move(), draw(), resolveCollisions(), etc.), implementing the various functions of the controls became much easier to understand. That of which I found beneficial every time I had to debug the various problems I had to face; as it made it very easy to figure out which sections of the code I needed to deal with.

Encapsulation

- Encapsulation was heavily implemented in all instance-specific variables. By having constraints and limited access to the data I found it very reassuring to know that I didn't have to worry about accidentally changing something I shouldn't have; a feeling that became more prevalent as the project got progressively more complex. Additionally, with the simple and easy-to-understand naming convention of the methods, coding and understanding the code became exponentially easier; which I found more and more essential every time I had to debug the code.

Inheritance

- Inheritance was used on the ant-related classes, whereby the concrete Queen and Worker classes were inherited from the abstract Ant class. With the bulk of the ant behaviours being located within the Ant class, the child classes became much easier to deal with as the behaviours pertaining to those specific classes were contained solely to those classes, while the behaviours shared by the Queen and Worker class was kept within the compounds of the Ant class. Just like abstraction and

encapsulation, I found this super helpful when it came to understanding, debugging, and adding behaviours to the ant-related classes. Lastly, thanks to the Ant class containing the basic behaviours pertaining to ants in general, the code in the Worker class became much less hectic, due to how I had set up the Worker class, whereby I had it control all the behaviours and change in states that occur for every possible collision event.

Polymorphism

- Polymorphism was extremely useful in the collision and draw events. For the collision events, polymorphism helped determine the class of the instance with “instanceof”, and typecasted the object to the appropriate collision check method based on the class it is an instance of. As for the draw event, polymorphism made updating what instances to draw and which ones to remove easy. By typecasting the iterator determining whether to draw the instance or remove it from the pane and array list(s) became a much simpler task.

Abstract Data Types

- Abstract data types were applied through the ArrayLists in the Simulation class. With the dynamic size of an ArrayLists, the implementation of adding and removing instances from an array became much easier.

POSSIBLE EXTENSIONS & IMPROVEMENT

1. A possible extension would include implementing environmental factors in the world. With the foundation already built and the basic behaviours set, the changes in the base code by adding an environmental stimulus to the world would really be limited to the simulation class and maybe the abstract ant class, all depending on how in-depth this extension would be. This limited range of changes in the base code can be contributed to the idea that this increase of complexity is all based on building upon the already made behaviours.
2. Another extension would include allowing the user to have more control over the overall simulation by adding more add-ons like buttons and sliders to affect different factors and variables such as the number of

queens fighting, the cost it requires to spawn a new ant, the speed of the simulation and more. With how the project is set up, adding more features such as these is just a matter of implementing these control objects in the controller class, adding methods to whatever class the controller is linked to, and linking these new methods to their respective controller; thus allowing a fairly easy process of giving more control and options to the user. With most of the code being encapsulated, adding these new controls won't actually affect the behaviour of the program, only the state.