Krishna Prasanna Kumar (K21004839) Shozab Anwar Siddique K21054573

* A description of your simulation, including the types of species that you are simulating, their behaviour and interactions.

A predator prey simulation consisting of dinosaurs and plants. The Dinosaur Abstract class is divided into three sectors primary, secondary and tertiary. These consist of 5 different Dinosaur species (Velociraptor, Tyrannosaurus, Spinosaurus, Triceratops and Brachiosaurus). We have decided to remove step numbers since we have added speed to the dinosaurs. We added speed with real-time hence the removal of steps, this has meant we have made modifications to both the simulate long simulation and simulate one step. The simulate one step is now a simulate one second method and the simulate long simulation now runs for 120 seconds which roughly lasts 7-8 days in the simulation.

Plants generate energy and grow with appropriate amounts of rain and sunlight. Dinosaurs require food to breed, move around and stay alive. Primary dinosaurs feed on plants to gain energy. Eating a plant doesn’t remove them from the simulation as they are given time to grow and be feed upon. Secondary dinosaurs need to feed upon primary dinosaurs to gain energy and tertiary feed upon primary and secondary dinosaurs. Eating another dinosaur removes them from the display and removes the dinosaur’s existence.

-disease- Organisms have a probability of having a disease. If a plant has a disease and it is eaten by an dinosaur, then the dinosaur gets infected by a different disease (e.g., food poisoning). The disease can spread from dinosaurs that are directly adjacent to it. Diseases cannot spread to different dinosaurs but rather give rise to other diseases due to difference in the biology of every dinosaur. After an individual dinosaur is infected 3 things can happen, it could die, recover from the disease or they continue to survive with the disease with a slower act speed. If the mother of the dinosaur has a disease, then the child is made to have the disease as well when it is born.

-speed- Each dinosaur is given its own speed so in this simulation some dinosaurs will move, eat and breed more often than others. Prey are initiated with a greater activity as they need to run away from the predators and the predators are made to have a slower overall activity. Producers don’t have the application of speed.

-sleeptime Each organism is assigned with a time during the day during which it does not act at all (i.e it sleeps). So, plants and some other prey dinosaurs are idle during the night. Some predators are awake during the night as they hunt for food (making it easier to eat as prey can’t run away while they are asleep). Other predators can sleep during the day allowing for enough time for the prey to repopulate and flee.

-gender Every single dinosaur object is initiated with a random gender (male or not male). When a male and female of the same species are directly adjacent to one another they breed. Only the female gives birth to children otherwise they proceed to find other partners. There is a limit to the number of times that any dinosaur can breed over their lifetime, and they can only breed if they are above a certain age.

Base Tasks This coursework allows you to have a lot of flexibility on what you deliver — be creative!

• Your simulation should have at least five different kinds of acting species. At least two of these should be predators (they eat another species), and at least two of them should not be predators (they may eat plants). You can either treat plants as if they’re always available or simulate their growth/death (see challenge task).

Initially common methods from rabbits and foxes were moved up the hierarchy to remove repeated code to the extent that each new dinosaur’s class mostly consists of the constants and accessor methods for the constants as these methods are used up in the hierarchy and elsewhere in the program. Making the three sectors inherit from the dinosaur class made it simpler to associate diseases with sectors of dinosaurs as well as simplifying the numerous preys that a predator can consume. The tertiary predators (Tyrannosaurus) store all adjacent locations in a list and for each of them it determines whether what the object stored in a field is and if is an instance of the other dinosaurs and it has not already exceeded it max eating limit then the prey is killed (removed) and the food level is increased for the predator. A very similar approach is used for primary and secondary dinosaurs. The main difference between tertiary/secondary and primary is that after the primary eats a plant the plant is not removed, instead its growth is reset so that it can be feed upon later. Brachiosaurus and Triceratops are the 2 prey and eat producers. To reduce coupling, we decided to add several layers of inheritance, for example the tertiary layer, this was introduced to create a food chain. We moved a lot of methods and fields out of the bottom layer into the top layers of inheritance. We introduced an organism superclass which stores the similar methods and fields of both plants and dinosaurs. This has made our solution more extendible since it is easy to add new dinosaurs at the bottom of the inheritance hierarchy or add a new entire type of organism at the top of the inheritance hierarchy.

• At least two predators should compete for the same food source.

Primary dinosaurs are eaten by dinosaurs in the tertiary class and secondary class. The dinosaurs from the tertiary class can of course, eat the secondary dinosaurs. This means that secondary dinosaurs compete with the tertiary dinosaurs as well as amongst themselves for prey nearby as the dinosaur closer to prey is more likely to eat the dinosaur and the one that’s not close is likely to die of starvation. Primary dinosaurs also compete for producers.

• Some or all of the species should distinguish male and female individuals. For these, the creatures can only propagate when a male and female individual meet. Meet means they need to be within a specified distance to each other, for example in a neighbouring cell. You will need to experiment with the parameters for breeding probability to create a stable population.

We have created a Boolean field in the organism class called “isMale” which is assigned a random boolean value upon creation of a Dinosaur object (each have an equal probability). An accessor method is created for this. Then in the canBreed() method we check if the current object is a male, if so return false as a male is not able to give off an offspring. We also check if the dinosaur has a certain food level as it requires energy to breed. Then we check for every adjacent location if the two organisms are of the same type, if their genders are different and if they are above breeding age. If all the conditions are met, the current dinosaur can breed with a single other dinosaur adjacent to it.

• You should keep track of the time of day. At least some creatures should exhibit different behaviour at some time of the day (for example: they may sleep at night and not move around during that time).

We have made a “timeOfDay” class which determines what time of day it should be at any time. The timeOfDay class itself holds how much time each day should last as a field. Every subclass of this class represents a time period of the day (i.e. morning, evening…). Each subclass hold a start and end variables that are set from in some range 0-(duration of a day). A cycle/order for the day is created in this way. Firstly all the subclasses of “timeofday” objects are created in a createday() method in the simulator class and stored in an arraylist. After this in the simulateday() method in simulator, we use a modulo operator on the current system time against the duration of each day so that we always get a number in the range of 0-(duration of day-1). After this for each time of day object, we check if the result after the modulo operation is in the given range of the time of day (Start – end fields), if so then the current time of day object is returned. Dinosaurs have a sleep time which they will stop moving but their age continues to increase. Plants will stop growing and stop making glucose during the night. Due to the inheritance hierarchy we added, the solution is relatively extendible and we can add new times of day relatively easily.

Challenge Tasks Once you have finished the base tasks, implement one or more challenge tasks. You can either choose from the following suggestions, or invent your own. You will be graded on a maximum of four challenge tasks.

* Simulate plants. Plants grow at a given rate, but they do not move. Some creatures eat plants. They will die if they do not find their food plant.

Producer class inherits from the organism class. It acts as a superclass for each individual plant that grows and doesn’t move. Each subclass of Producer type is given a max age, growth rate, rain, sunlight levels and appropriate accessor methods for these. Primary dinosaurs cannot eat producers until they have grown to a certain height/age. Producers inherit from the organism superclass which store the similar methods and fields of all living organisms.

* Simulate weather. Weather can change, and it influences the behaviour of some simulated aspects. For example, grass may not grow without rain.

We created an abstract weather class where several weather classes inherit from. Weather is dependent on the current time of day, for example, during the Night it cannot be sunny. In addition to this, the speed of a dinosaur can be affected by the weather hence it stores a multiplier at which the speed is affected. Meteor is a special type of weather which will be a rare occurrence but will wipe out a large amount of organisms similar to how the dinosaurs became extinct, so don’t be surprised if the simulation ends after meteors. Weather can cause different species to act differently, for example the speed of dinosaurs could increase and the amount water and sunlight the plants receive could vary depending on the weather. We have added a weather inheritance hierarchy where all weather subclasses contain mainly just values that are specific to them, this makes it more extendible since we can add new types of weather easily.

* Simulate disease. Some dinosaurs are occasionally infected. Infection can spread to other dinosaurs when they meet.

A disease interface is created to hold methods required by any disease type. When plants are created, there is a chance of them having a disease. This disease is initialised in the constructor. If an infected plant is eaten, the dinosaur then is set to have a disease. The way that disease spread is that in the all adjacent locations are stored in a list and if an adjacent dinosaur is known to have a disease and both dinosaurs are of the same type, a new disease object is created and set to the current dinosaur. As this method is called every time an dinosaur moves (due to the event-based nature of the simulation), it ensures that every dinosaur is set the disease when they come in contact. There is a chance that an infected individual may recover from any disease (these probabilities are set in the subclasses of the disease interface). The method for this is created in the organism class which randomly generates a value from 0 and 1 and if the value is more than the death rate but less than the recovery rate, then the organism is cured of the disease, otherwise if its less than disease it will die. In the case that none of these conditions is met, nothing will happen allowing for times during which the dinosaur is infected over larger periods of time. We have also added immunity, if a dinosaur recovers from a disease, it adds the disease to a list of diseases that it has become immune to and will no longer catch that disease. Due to the disease class being an interface we can add new diseases easily in the future with different death rates and recovery rates.

* Speed/evolution

Speed/Evolution is the extra challenge task we decided to add, for this we decided to use real time to calculate speed and if enough time has passed for the dinosaur to move. Speed can be affected by weather, for example during the snow the speed may be 50% slower. Base speed is used to reset the speed after this reduction of speed has been made. Evolution is also linked with speed, an offspring has a probability in which the speed can increase, we have also added some other traits linked with evolution in the giveBirth method. This can sometimes be visibly seen if looked at closely.