**Chapter 12**

Ex. 12.1 – Create a simulator object. Call the simulateOneStep method. Does the number of foxes change if this method is only called once?

Yes when I called the method the numbers of both foxes and rabbits changed.

Ex 12.2 - Does the number of foxes change on everystep? What natural process do you think is being modeled that causes the foxes to increase or decrease?

Yes, the number of foxes changes with every step. And the natural processes being modelled are likely a birth rate and a death rate that is affected by the population of rabbits.

Ex 12.3 – Call the simulate method with a parameter to run the simulation continuously for a significant number of steps (50 or 100). Do the number of foxes and rabbits increase or decrease at similar rates?

No, they increase and decrease at different rates.

Ex. 12.4 – If the simulation runs for an extended period of time a clear pattern or ebb and flow is seen. As the population of rabbits increases the foxes follow are a slower rate. Once the rabbits reach carrying capacity their population decreases. The foxes follow this trend but there is a short delay between the rabbit population decreasing and the foxes decreasing.

Ex. 12.5 – Use the reset method to create a new starting state. Run the simulation again. Is it identical to the first? If not, is there a broadly similar pattern emerging?

The simulations were not identical but they did have the same overarching trends.

Ex. 12.6 – If you run the simulation long enough will either completely die off? If so what might cause this?

Yes, it is possible for either to die off completely. If there are to many foxes when the rabbits are at their lowest population they are all eaten and it is possible for the foxes to be too far from the rabbits and starve. The simulation ends if either population hits zero.

Ex. 12.7 – In the source code of the simulate class, find the simulate method. There is a line labeled delay, un comment it out and experiment with different delays. Leave it in a state that makes the simulation look useful.

It changes how long each step is displayed. I chose to set it to 160 as I thought it was a fair balance between letting you actually read the populations and getting through the steps in a reasonable time.

Ex. 12.8 – Make not of the numbers of foxes and rabbits at the beginning and end of a long run. This will be helpful when we make changes on the regression testing.

First 3 frames

|  |  |
| --- | --- |
| Rabbits | Foxes |
| 778 | 182 |
| 938 | 152 |
| 1038 | 149 |

Last 3 frames

|  |  |
| --- | --- |
| Rabbits | Foxes |
| 2933 | 264 |
| 3018 | 265 |
| 3065 | 269 |

Ex. 12.9 – After the simulator has run awhile, reset set it. If you run the first few steps it should re-create the first few steps. Why is this?

This is caused because it is generated the original random factors used previously. To get it to change the randomized must be reset then it will generate new random numbers to start the simulation with.

Ex. 12.10 – To break this repeatability setting the use Shared filed to false breaks the repeatability. Change it and test it out. Then change it back. Repeatability will be important for later testing.

Done.

Ex. 12.11 – Do you feel that omitting gender as an attribute in the rabbit class is likely to lead to an inaccurate simulation. Write down arguments for and against including it.

Well for the sake of simplicity and saving time it is much easier to not include gender. Because if it is added instead of simply having a chance of reproduction to only female rabbits a condition of it being next to a male should also be met. this increases time it takes the complexity of the code and if such an accurate system isn’t needed there is no point.

An argument for will be that we want the system to be a better representation of the real world. So, we should include to have a better simulation.

Ex. 12.12 – Are there other simplifications that you feel are present in our implementation of the rabbit class, compared with real life?

The biggest things I can think of are the fact that there isn’t any food constraint, litter size isn’t variable and disease isn’t included. All of these things would have a considerable impact on the how accurate our simulation is.

Ex. 12.13– Experiment with the effects of altering some or all the values of the class variables in the Rabbit class. For instance, what effect does it have on the populations if the breeding probability of rabbits is much higher or much lower than it currently is?

When the breeding rater of the rabbits increases, their population grows much more quickly and the foxes take a long time to catch up. When the breeding rate is decreased, the foxes end up starving because there aren’t enough rabbits to feed them.

Ex. 12.14– Asses the degree to which we have simplified the model of foxes and evaluate whether you feel the simplifications are likely to lead to an inaccurate simulation.

The fox has many of the same issues as the rabbit. Gendered has been omitted from them however the fox is better because it does require food. This is a huge hep when it comes to increasing the accuracy of the simulation.

Ex. 12.15– Does increasing the maximum age of foxes lead to a significantly higher number of foxes throughout the simulation, or is the rabbit population more likely to be reduced to zero?

Increasing the maximum age of foxes doesn’t make a any notable difference in the simulation.

Ex. 12.16 – Experiment with different configurations. Do some cause one to disappear? Are some configurations rather stable?

If there are large differences in the numbers one species disappears quickly. And if things reasonable ratio in between the simulation is rather stable.

Ex. 12.17 – Experiment with different size fields. Does the size of the field affect the likelihood of a species survival?

Yes, the size of the field has a large effect on a species survival. If the field is large then the foxes are more likely to become isolated and die out. If the field is small then the rabbits are at risk because the foxes have such an easy time finding them.

Ex. 12.18 – Compare the results of running a sim with a single large field vs two simulation with fields half the size of the first. Do you notice any significant differences in population dynamics between the two scenarios?

Yes, when the field is split in half there to population of both species is significantly reduced over time.

Ex. 12.19– Repeat but change sixes of the fields. Does it matter how the single field is split?

Aside from the reduction of overall population if one area is too small it increases the likelihood of that portion to have a species die off.

Ex. 12.20– Modify The findFood method so that rabbits all in adjacent tiles get eaten in a step. Assess the impact of this change on the results of the simulation.

Done and saved

This change doesn’t seem to affect the likelihood of survival but it does seem to increase the rate at which rabbit population decreases

Ex. 12.21– Impose a make food level on fox.

Done and saved.

Ex. 12.22– Given the random elements in the simulation, argue why the population in an apparently stable simulation may go extinct.

Since the populations rise and fall there is a chance that in that a it’s lowest population one of the species may have bad luck. Such as the rabbits being surrounded by foxes or the last foxes not being able to find food. This can cause a species to go extinct.

Ex. 12.23– Modify the populate method of the simulator to determine whether setting an initial age of populations at zero is always catastrophic. Run a few times with different initial states.

This change causes issues because rabbits can’t breed until a certain age whereas foxes can eat from the first move. This makes it so there is an increased chance of the foxes eating all the rabbits before they can even reproduce and this can cause the simulation to end.

Ex. 12.24– Make the same change to foxes and change rabbits back to previous state. Once foxes are old enough to breed does the simulation start to behave like the original version? What does this tell us about initial populations impact on the simulations out come?

If foxes reach breeding age this has little effect on the simulation. But until then they have a higher chance of dying off.

Ex. 12.25– Check that you thoroughly understand how the field and animal lists are kept consistent between the simulations method in Simulator, hunt in Fox, and run in Rabbit.

Done

Ex. 12.26– Do you think it would be better for Simulator not to keep separate lists of foxes and rabbits but to generate these lists again from the contents of the field at the beginning of each simulation step?

Both the proposed method and the current have benefits and hindrances. Currently there is a risk of things becoming inconsistent and this issue would be fixed nu generating new lists from field. The largest issue of generating these lists before each step is that it slows down the program.

Ex. 12.27– Write a test to ensure that at the end of the simulation there is no animal (dead or alive) in the field that ins not in one of the list and vice versa. Should there be and dead animals in any of those places at that stage?

Ex. 12.28– Identify some similarities and differences between the Fox and rabbit classes. Make separate lists of fields methods , constructors and distinguish between the class variables (Static fields) and instances of variables.

The rabbit and fox class have the share these Class Variables:

BREEDING\_AGE, MAX\_AGE, BREEDING\_PROBABILITY, MAX\_LITTER\_SIZE, and Random rand.

Fox has one more class variable and it is RABBIT\_FOOD\_VALUE.

They both have the following fields:

Age, alive, and Location location.

The fox also has one more field and that is foodLevel.

The constructors are identical except of names and the foxes initializing its extra field.

The two both have methods named: incrementAge, breed, canBreed, isAlive, setLocation(int row, int col), and setLocation(Location).

The rabbit does have two unique methods and they are run and setEaten.

The fox class has three unique methods and they are hunt, increment, and findFood.

Ex. 12.29– Candidate methods for placement in a super class. Which methods are truly identical in the fox and rabbit classes?

There are only four truly identical methods. They are the getLocation, isAlive, setDead, and setLocation methods.

Ex. 12.30– If the two values of a particular class variables were identical would it make and difference to your assessment of which methods are truly identical?

For what we are doing it doesn’t make help. This is because we will want to change the value for each class. If we didn’t want to make things different than having them more sense to treat methods using the field as identical.

Ex. 12.31– What sort of regression testing strategy could you establish before undertaking the process of refactoring on the simulation? Is this something you could conveniently automate?

Yes, this can be easily automated by using Junit test classes. We can use them to check if changes have affected the functionality of our code.

Ex. 12.32– Create an animal superclass and make the changes discussed above.

Done and saved

Ex. 12.33– How has using inheritance improved the project so far?

This helps us eliminate code duplication and stream line the fox and rabbit classes.

Ex. 12.34– Why is it not possible for it to treat each object in the collection simply using the object type?

It isn’t possible because the act method is available only in the animal class.

Ex. 12.35– Is it necessary for a class with one or more abstract methods to be defined as abstract?

IF a class has abstract methods it must be an abstract class. This is because concrete classes can’t override the abstract method (gist of message given by compiler when attempted)

Ex. 12.36 – Is it possible for a class with no abstract methods to be defined as abstract?

Yes, it is possible to have an abstract class without any abstract methods. This could be useful if you wanted a class but wanted to make sure no instances could be created of it.

Ex. 12.37 – Would it ever make sense to define a class as abstract if it has no abstract methods?

Yes, they could be instances where this would make sense. For example in our zuul game if we wanted to add monsters and characters other than the player a being class could act as a super class and we would never want an instance of being created without the additional details of a sub class.

Ex. 12.38 – Which classes in the java.util package are abstract?

The following classes are all abstract in the java.util package: AbstractCollection, AbstractList, AbstractMap, AbstractSequentialList, and AbstractSet.

The concrete classes that extend them are: ArrayList, Arrays, LinkedList, LinkedHashMap, LinkedHashSet, Stack, Vector, TreeSet, and TreeMap.

Ex. 12.39 – Can you tell from the API document for an abstract class which of its methods are abstract? Do you need to know which methods are abstract?

Yes, in the method summary column we can see what methods are abstract. We do need to know what these methods are because when creating a class that extends it those methods must be implemented or it won’t work.

Ex. 12.40 – Why are overriding rules particularly significant in our attempts to introduce inheritance into this application?

Override methods can be inherited from concrete classes. Overriding in inheritance occurs when a method defined in a superclass and another method in a subclass have the same name and header in separate locations and with different bodies.

Ex. 12.41 – Review Split the simulator into two classes. One responsible for only the simulation and the other responsible for population generation. Call this class PopulationGenerator.

Done and saved SEE EX 12.32

Ex. 12.42 – Pay attention to Graph View outputs. Explain the meaning of the graph you see. Explain the way it looks. Is there a relationship between the two curves?

A Black curve fluctuates considerable, rising and falling, while a smaller red one follows a similar trend but it is never quite as dramatic a change. Its peaks appear when the gray graph is about mid decline and hits its lowest point the gray graph begins to skyrocket. These graphs show the relationship between the population of rabbits (gray graph) and the population of foxes (red graph). When rabbits hit their highest population foxes begin to increase and then can’t be sustained and they begin a decline. The decline in foxes allows more rabbits to be born and this creates a cycle of high and low populations for both species

Ex. 12.43 – Experiment with smaller fields, Does the graph give new insight or help explain what you see?

All the graph does is confirm what we say previously. In a smaller environment it is easier for foxes to find food so they either eat all the rabbits who don’t have room to run or they eat almost all the rabbits and starve when they can’t get to the remaining. It shows us that there is a minimum size of field to create and maintain a semi-stable system.

Ex. 12.44 – record the number of foxes and rabbits over a small number of steps to prepare for the regression testing of changes to come.

|  |  |  |
| --- | --- | --- |
| Step # | Foxes | Rabbits |
| 1 | 171 | 882 |
| 2 | 165 | 951 |
| 3 | 175 | 1022 |
| 4 | 178 | 1126 |
| 5 | 172 | 1188 |

Ex. 12.45 – Move age field from fox and rabbit to Animal. Initialize it to zero. Set up and accessor and mutator method so Fox and Rabbit can change it individually.

Done and saved

Ex. 12.46 – Move the canBreed method from fox and rabbit into Animal. Rewrite it as shown in 12.8. Provide appropriate versions of getBreedingAge in fox and rabbit that return the distinctive breeding age values.

Done and saved

Ex. 12.47 – Move the incrementAge method from fox and rabbit to Animal by providing an abstract getMaxAge method and concrete versions in fox and rabbit.

Done and saved

Ex. 12.48 – Can the breed method be moved to Animal? If so make the change.

Yes, it can be made. Done and saved

Ex. 12.49 – In light of these changes consider the visibility of each method, make an changes you feel are necessary.

Done

Ex. 12.50 – Was it possible to make these changes without having any impact on other classes in the project?

Yes, I was able to make these changes without any changes in the other classes. The only changes that had to be made were changing methods of the classes from private to protected.

Ex. 12.51 – Define a completely new type of animal for the simulation that is a subclass of Animal.

Done and saved

Ex. 12.52 – Is it possible use the to move the give birth method into the animal class?

Yes it is possible. Done and saved.

Ex. 12.53 –

Ex. 12.54 –

Ex. 12.55 –

Ex. 12.56 –

Ex. 12.57 –

Ex. 12.58 –

Ex. 12.59 –

Ex . 12.60

Ex . 12.61

Ex . 12.62

Ex . 12.63

Ex . 12.64

Ex . 12.65

Ex . 12.66

Ex . 12.67

Ex . 12.68

Ex . 12.69

Ex . 12.70

Ex . 12.71

Ex . 12.72