UNITAR GRADUATE SCHOOL

Course

2022/05 ITWM5113 Software Design and Development

Course Instructor

Dr Simon Lau

Assignment Submission

(Group/Individual)

Assignment Title: Final Project Report Project: -

Object Oriented In JAVA – Animal Kingdom

Name	Student ID	Section
WAN NOORDIANA WAN HANAFI	UNU2200288	MIT

TABLE OF CONTENT

Page Number

1. Abstract

This project objectives are to develop a program to simulate the behavior of certain animals using the Java programming language. The main goal is to practice how to define classes in Java programming. Different kinds of animals will behave in different ways.Animal Kingdom is a computer project written in Java that simulates a virtual world that is home to many creatures, including food, flytraps, tigers, bears, and giants. The class for this project has been pre-programmed, and it contains both the process and the parameters. The virtual environment has been specified, and a graphical user interface has been generated, thanks to the pre-programmed class (GUI).

The animals will be free to move about in the virtual world, and each one will exhibit its own distinctive behaviour in accordance with the information that has been presented in its class. There will be a separate class for each animal, such as tiger.java, bear.java, and others like them. The functions getMove() and function toString() { [native code] }() are responsible for defining the critter's behaviour, which includes how the critter moves and interacts with the world and other critters. getMove() is responsible for defining the string that represents the critter. function toString() { [native code] }() is responsible for defining the string. The getColor() function is where the colour parameter for Critter will be defined. Critter will also have its own colour parameter. When a critter travels throughout the globe, it will meet other critters and engage in conversation with them. Some critters will infect others, causing them to change type. Some animals will change their behaviour but keep their species intact as they do so. In a short amount of time, the vast majority of the creatures will evolve into ones that are more prone to infection.

2. Project Instructions

For this project, supporting code will be provided in order to run the simulation. Below files need to be downloaded and included in the same project folder.

a) Critter.java
b) CritterInfo.java
c) CritterPanel.java
d) CritterModel.java
e) CritterMain.java
f) CritterFrame.java
g) Food.java
h) FlyTrap.java

Every object in this world is a "critter", where Critter is the super class with default behavior defined. You will be writing five classes, each representing a different type of Animal: Bear, Tiger, WhiteTiger,Giant and NinjaCat. All the classes you right should be sub classes of Critter. On each round of the simulation, each critter is asked for 3 pieces of information:

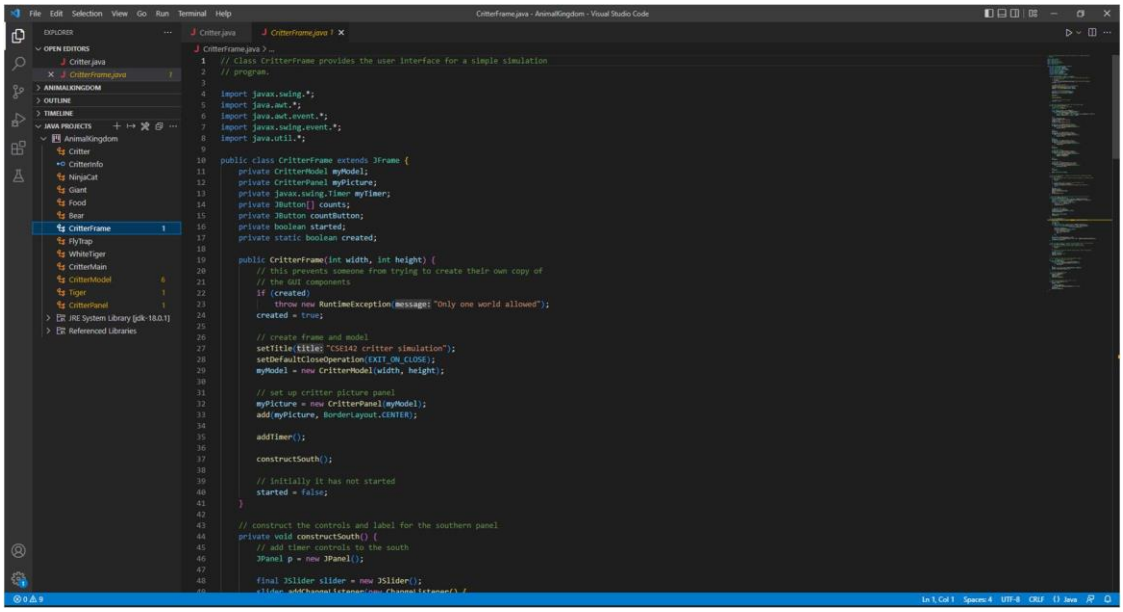
a) How should it act?
b) What color is it?
c) What string represents that critter?

These 3 pieces of information are provided by 3 methods present in each Critter class. You will beresponsible for overriding these methods and programming their appropriate behaviour:

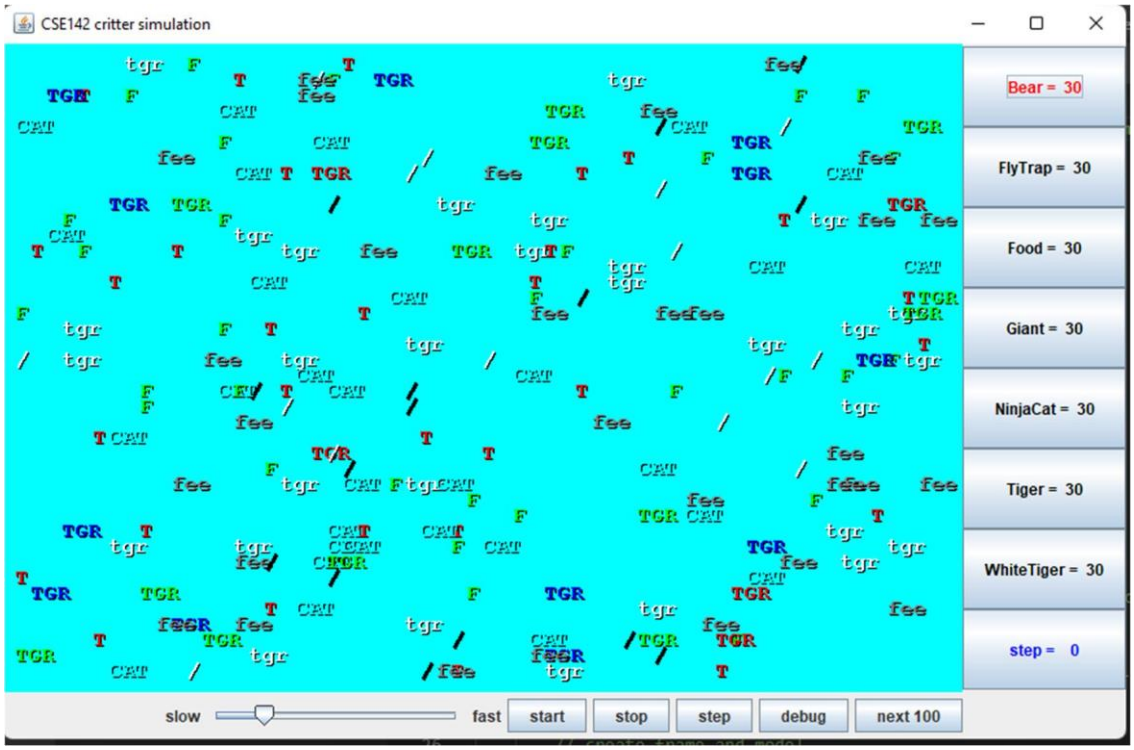
```
public Action getMove(CritterInfo info) {
...
}
public Color getColor() {
...
}
public String toString() {
...
}
```

3. Tool Used

Tools used to develop and run the simulation is “Visual Studio Code”.

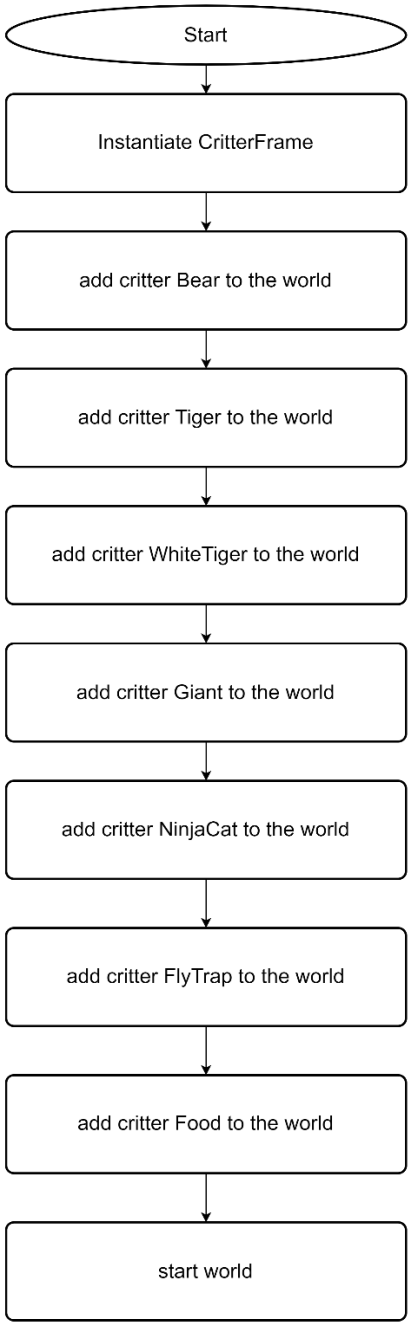


“Visual Studio Code”



“Java Simul

4. Program Workflow and Logics



Flow chart for CritterMain (figure 1.1)

4.1 Behaviour

a) getColor - color to display

b) getMove - movement

c) toString - letter to display

4.2 A Critter Subclass

```
public class name extends Critter { ... }

public abstract class Critter {
    public boolean eat()
    public Attack fight(String opponent)
            // ROAR, POUNCE, SCRATCH
    public Color getColor()
    public Direction getMove()
            // NORTH, SOUTH, EAST, WEST, CENTER
    public String toString()
}
```

4.3 Critter Class Extension - Bear

Bear.java > ...

```java
// Critter class extension for a critter called a 'Bear'
// It is represented by a '/' or a '\'
// Wan Noordiana Wan Hanafi

import java.awt.*;

public class Bear extends Critter {
    private boolean polar;
    private int moves;

    public Bear(boolean polar){
        this.polar=polar;
        getColor();
    }

    public Color getColor() {
        //Color.WHITE for a polar bear (when polar is true),
        // Color.BLACK otherwise (when polar is false)
        if (this.polar){
            return Color.WHITE;
        } else {
            return Color.BLACK;
        }
    }
```

```java
    public String toString(){
        //Should alternate on each different move between a slash character (/)
        // and a backslash character () starting with a slash.
        if (moves%2==0){
            return "/";
        } else {
            return "\\";
        }

    }

    public Action getMove(CritterInfo info){
        //always infect if an enemy is in front, otherwise hop if possible, otherwise turn left.
        moves++;
        if(info.getFront()==Neighbor.OTHER){
            return Action.INFECT;
        } else if (info.getFront()==Neighbor.EMPTY){
            return Action.HOP;
        } else {
            return super.getMove(info);
        }
    }

}
```

4.4 Critter Class Extension - Tiger

Tiger.java > ...

```java
// Critter class extension for a critter called a 'Tiger'
// It is represented by an 'TGR' that changes color every 3 moves
// Wan Noordiana Wan Hanafi

import java.awt.*;
import java.util.*;


public class Tiger extends Critter {
    private int colorMoves;
    Color tigerColor;
    Random rand = new Random();

    public Tiger(){
        colorMoves=0;//1,2,3
        getColor();
    }

```

```java
    public Color getColor() {
        //Randomly picks one of three colors (Color.RED, Color.GREEN, Color.BLUE) and uses that color for
        // then randomly picks one of those colors again for the next three moves,
        // then randomly picks another one of those colors for the next three moves, and so on.
        if (colorMoves%3==0){ // set new color
            int x=0;
            while (x==0){
                int i=rand.nextInt(bound: 3); //0.Red 1.Green 2.Black
                if (i==0 && this.tigerColor!=Color.RED){
                    this.tigerColor= Color.RED;
                    x++;
                } if (i==1 && tigerColor!=Color.GREEN){
                    this.tigerColor=Color.GREEN;
                    x++;
                } if (i==2 && tigerColor!=Color.BLUE){
                    this.tigerColor=Color.BLUE;
                    x++;
                }
            }

        }
        return tigerColor;
```

```
42
43      public String toString() {
44          return "TGR";
45      }
46
47      public Action getMove(CritterInfo info) {
48          //always infect if an enemy is in front,
49          // otherwise if a wall is in front or to the right, then turn left,
50          // otherwise if a fellow Tiger is in front, then turn right, otherwise hop.
51          colorMoves++;
52          if (info.getFront()==Neighbor.OTHER){
53              return Action.INFECT;
54          } else if (info.getFront()==Neighbor.WALL||info.getRight()==Neighbor.WALL){
55              return Action.LEFT;
56          } else if (info.getFront()==Neighbor.SAME){
57              return Action.RIGHT;
58          } else {
59              return Action.HOP;
60          }
61      }
62  }
63
```

4.5 Critter Class Extension – WhiteTiger

WhiteTiger.java > ...

```
1   // Critter class extension for a critter called a 'WhiteTiger'
2   // Represented by a 'tgr'
3   // Wan Noordiana Wan Hanafi
4
5   import java.awt.*;
6
7   public class WhiteTiger extends Tiger {
8       boolean hasInfected;
9
10      public WhiteTiger(){
11          hasInfected=false;
12      }
13
14
15      public Color getColor() {
16          //Always Color.WHITE.
17          return Color.WHITE;
18      }
19
20
```

```
    public String toString() {
        //"tgr" if it hasn't infected another Critter yet, "TGR" if it has infected.
        if (hasInfected){
            return super.toString();
        } else {
            return "tgr";
        }
    }


    public Action getMove(CritterInfo info) {
        //Same as a Tiger.
        // Note: you'll have to override this method to figure out if it has infected another Critter.
        if (info.getFront()==Neighbor.OTHER){
            hasInfected=true;
        }
        return super.getMove(info);

    }
}

```

4.6 Critter Class Critter Class Extension – Giant

Giant.java > ...

```
// Critter class extension for a critter called a 'Giant'
// It is represented by 'fee', 'fie', 'foe', or 'fum'
// Wan Noordiana Wan Hanafi

import java.awt.*;

public class Giant extends Critter{
    private int moves;

    public Giant(){
        moves=1;
        getColor();
    }

    public Color getColor (){
        return Color.GRAY;
    }


    public String toString() {
        //"fee" for 6 moves, then "fie" for 6 moves, then "foe" for 6 moves, then "fum" for 6 moves, then
        if (moves<=6){
            return "fee";
        } else if (moves<=12){
            return "fie";
        } else if (moves<=18){
            return "foe";
        } else {
            return "fum";
        }
    }
```

```
32
33      public Action getMove(CritterInfo info) {
34          //always infect if an enemy is in front, otherwise hop if possible, otherwise turn right
35          //track moves
36          if (info.getFront()==Neighbor.OTHER){
37              countMoves();
38              return Action.INFECT;
39          } else if(info.getFront()==Neighbor.EMPTY){
40              countMoves();
41              return Action.HOP;
42          } else {
43              countMoves();
44              return Action.RIGHT;
45          }
46      }
47
48      public void countMoves(){
49          if (moves==24){
50              moves=1;
51          } else {
52              moves++;
53          }
54      }
55  }
56
```

4.7 Critter Class – NinjaCat

NinjaCat.java > ...

```
1   import java.awt.*;
2
3   public class NinjaCat extends Tiger {
4
5       public boolean hasInfected;
6
7       public NinjaCat (){
8           hasInfected=false;
9       }
10
11      public Color getColor() {
12          if (hasInfected){
13              return Color.MAGENTA;
14          } else {
15              return Color.orange;
16          }
17
18      }
19
20
```

```
21      public String toString() {
22          if (hasInfected){
23              return "Z";
24          } else {
25              return "z";
26          }
27
28      }
29
30
31      public Action getMove(CritterInfo info) {
32          //same as Tiger, but changes color when has infected
33          if (info.getFront()==Neighbor.OTHER){
34              hasInfected=true;
35          }
36          return super.getMove(info);
37
38      }
39  }
40
```

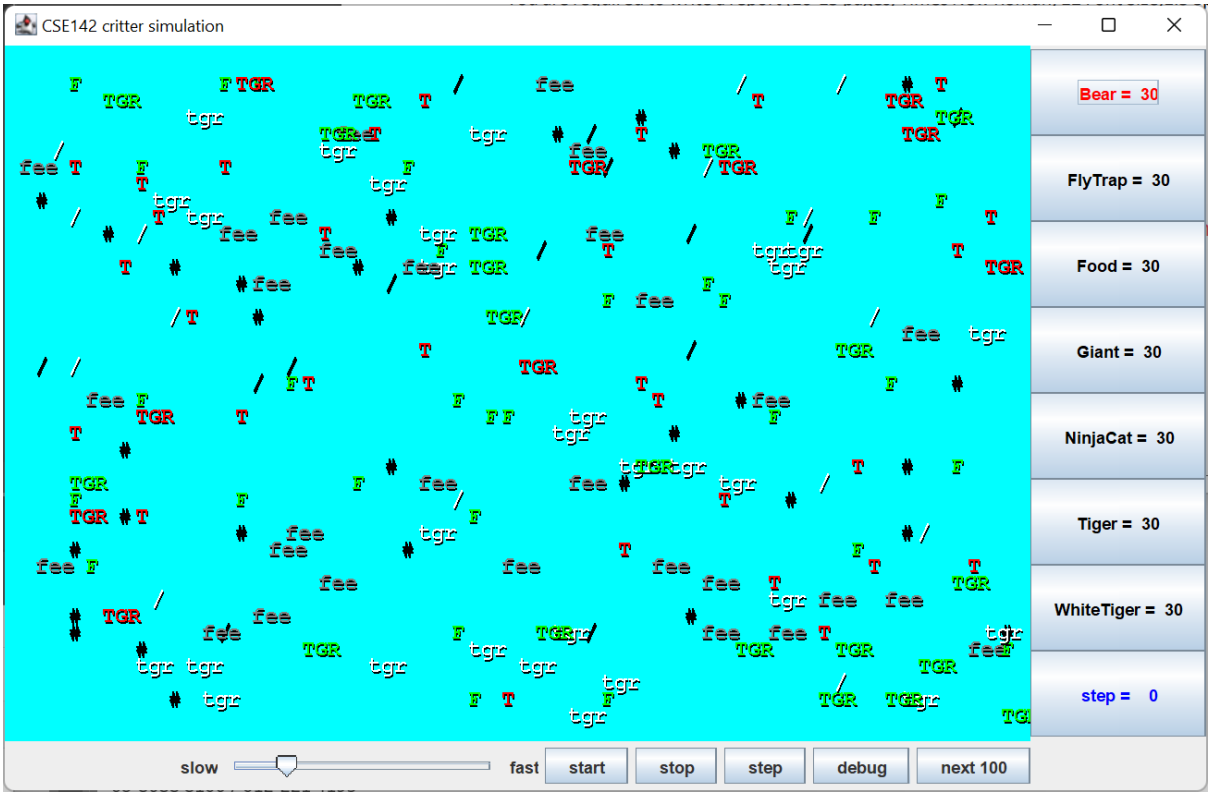
5 Simulation Result

The critter simulation program is successfully executed when all the critter classes being enable in the CritterMain.java. The graphical user interface will show up with all the critter inside the main screen which represents the virtual world for animal kingdom. The critter counter at the right side of the windows will show how many critters are in the virtual world. It shows the number of each critter.

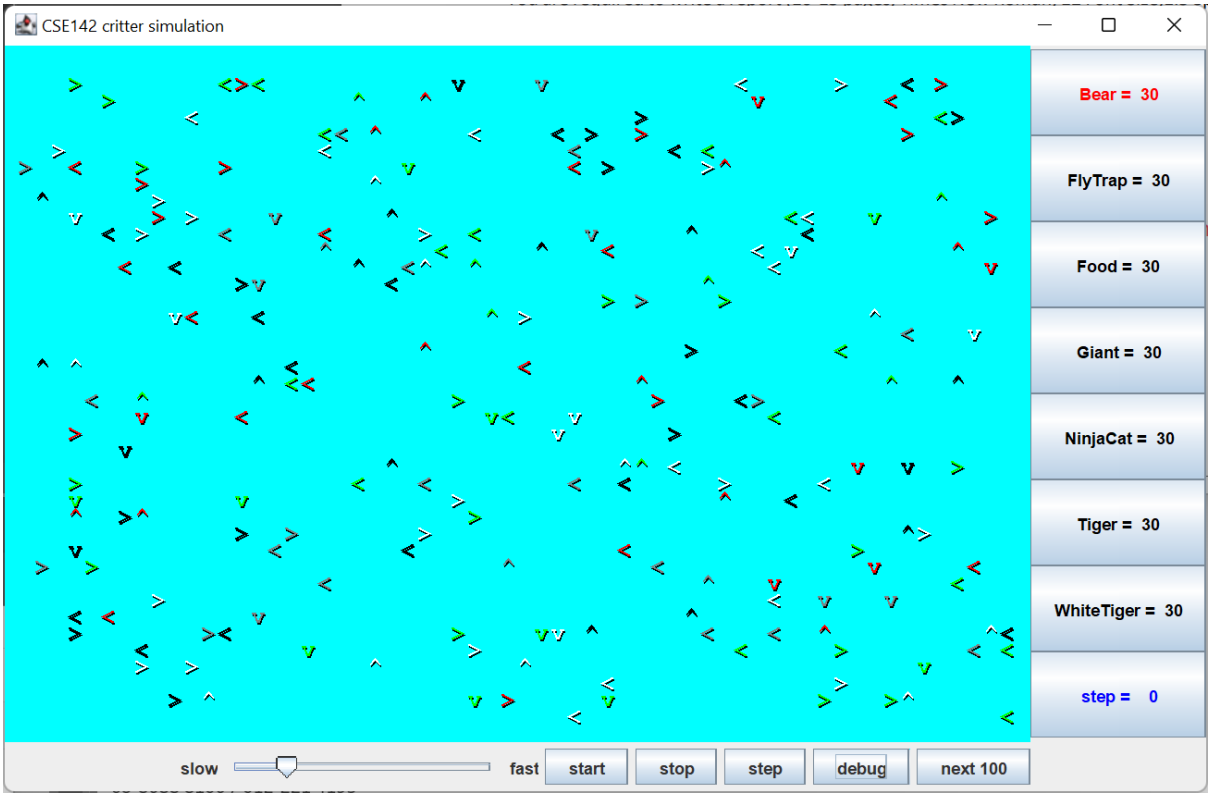
Option button and slider are at the bottom of the screen. There are 5 button to control the simulation:

1. Start button – to start the simulation
2. Stop button – to stop the simulation
3. Step button – to step through the simulation frame by frame
4. Debug button – To change the critter string to arrow symbol, this will make tracking critter orientation much easier.
5. Next 100 – fast forward the simulation frame to next 100 frame.
6. Slider – to control the speed of the simulation.

As can be seen in figure 8.1, it shows the simulation in normal mode (non-debug mode), it will show all the critter string as defined inside the critter class. While in figure 8.2 show the simulation in debug mode.

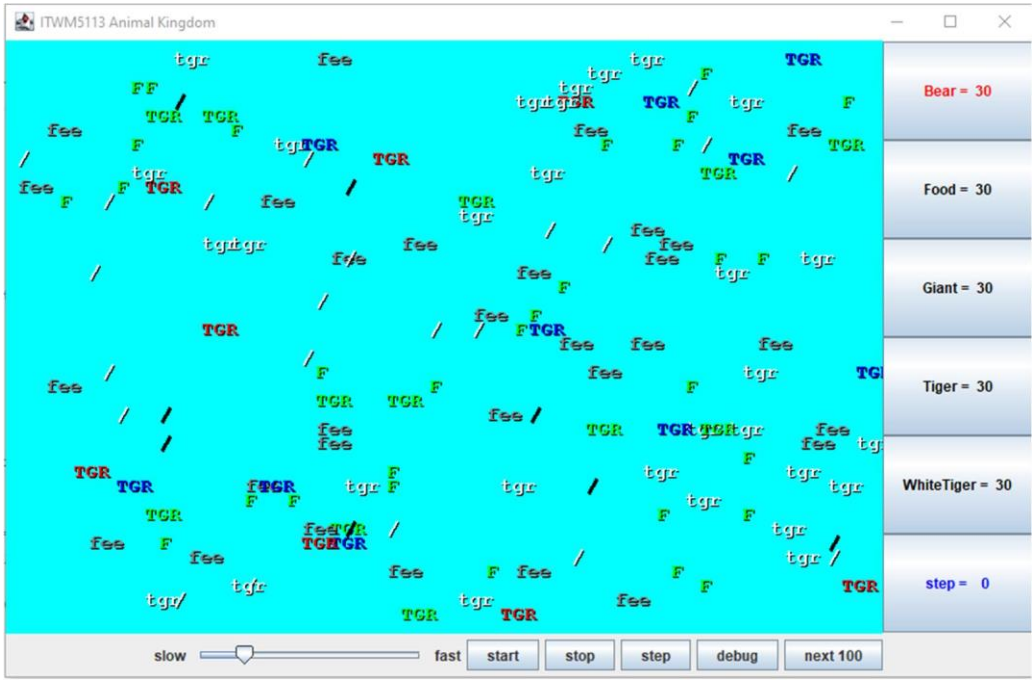


Normal Simulation Mode (Figure 8.1)

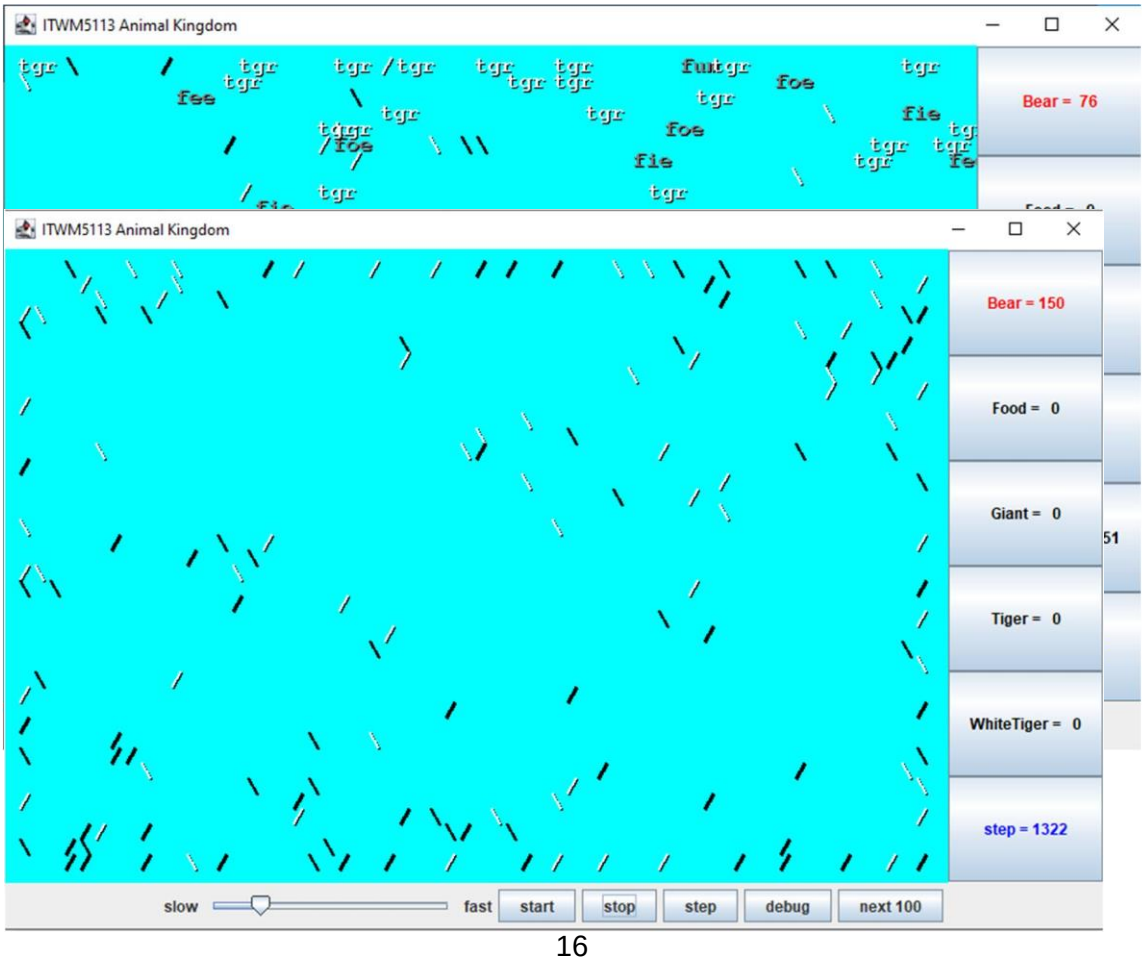


Debug Simulation Mode (Figure 9.2)

5.1 White Tiger



5.2 Bear



6.Project Repository

This whole project source code can be found in the GitHub repository as provided in the link https://github.com/DIANAHANAFI/ITWM5113-Animal-Kingdom-.git

7. Summary

The simulation results shows that the each of the class behave differently based on the define criteria. The module is based on the main superclass definition of the variables.

Some of the class variable that are static is remain the same as each class reach 0 and no changes can beseen happen after that.

A class is used for any of the following program.

a. A Program: Has a main and perhaps other static methods. Example: CritterMain
   Does not usually declare any static fields (except final)
b. An object class: Define new type of object
   Example: Critter, Tiger, WhiteTiger, Bear
   Declares object fields, constructor(s), and methods.
c. A module: Utility code implemented as static methods. Example: Math (Critter Class - Tiger.Java)