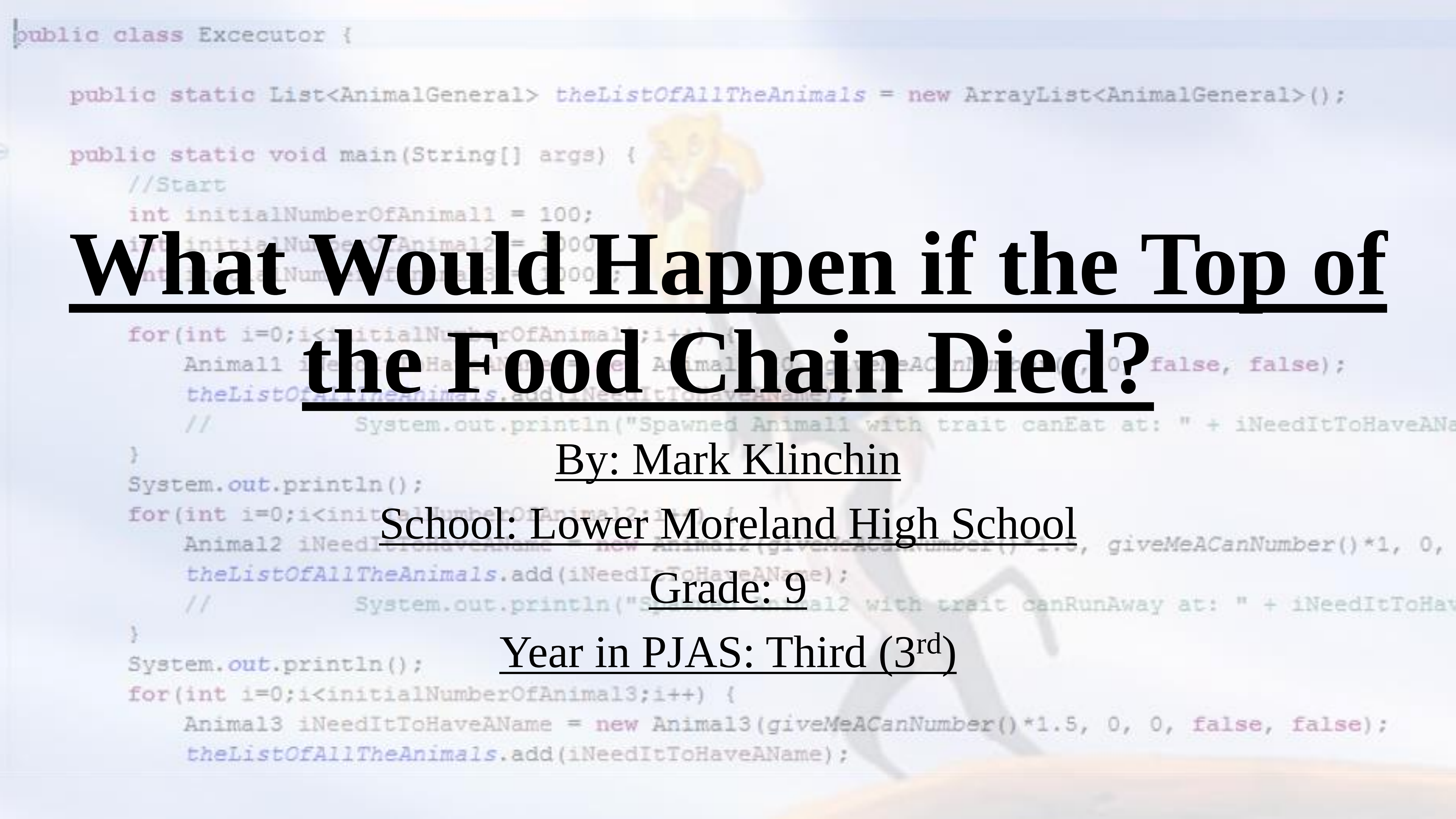
What Would Happen if the Top of the Food Chain Died?

By: Mark Klinchin

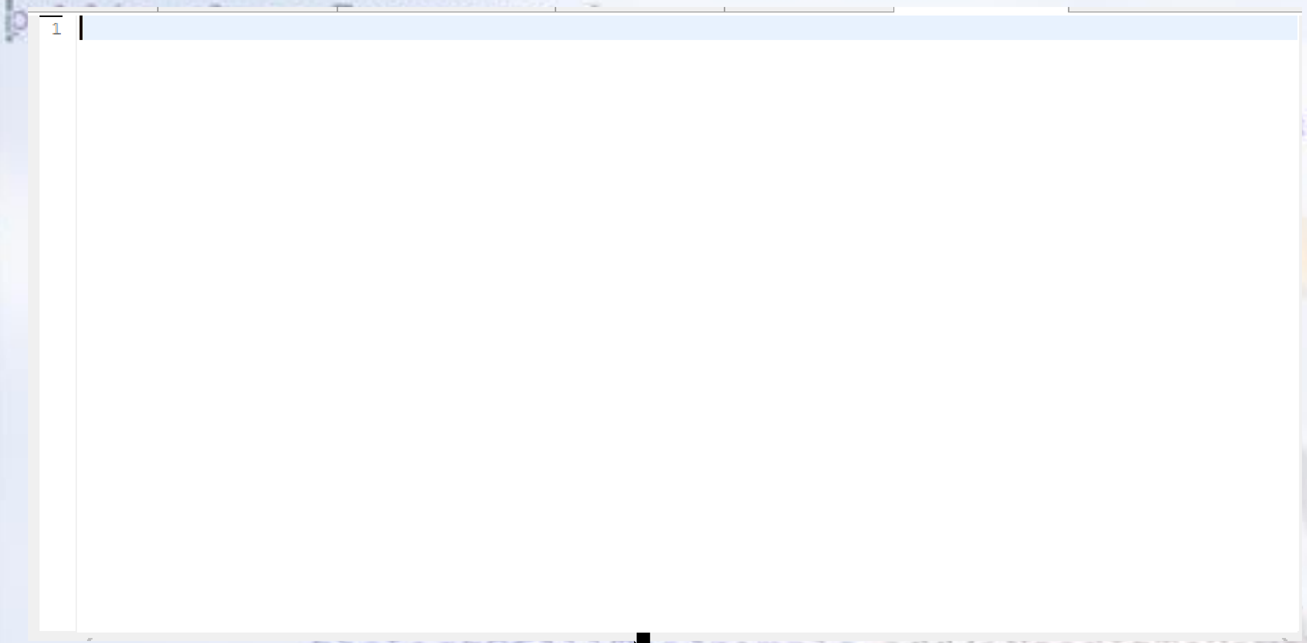
School: Lower Moreland High School

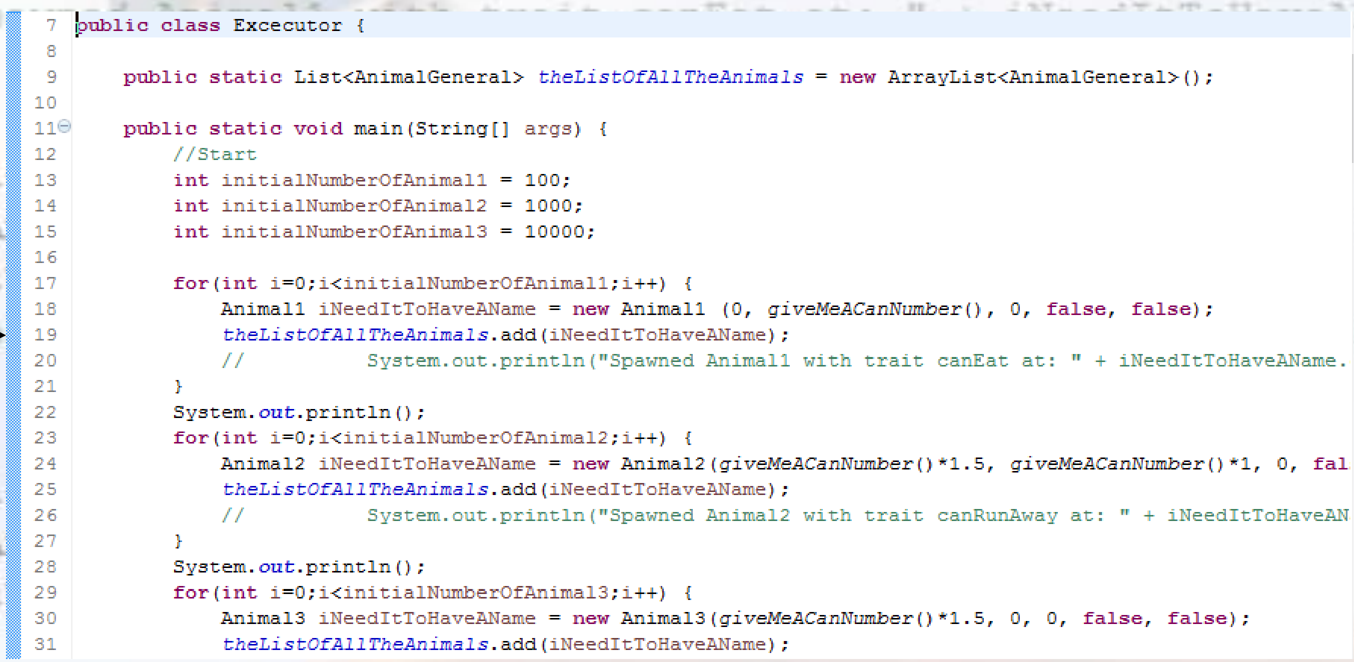
Grade: 9

Year in PJAS: Third (3^{rd})



```
public class Excecutor {

    public static List<AnimalGeneral> theListOfAllTheAnimals = new ArrayList<AnimalGeneral>();

    public static void main(String[] args) {
        //Start
        int initialNumberOfAnimal1 = 100;
        int initialNumberOfAnimal2 = 1000
        int initialNumberOfAnimal3 = 1000;

        for(int i=0;i<initialNumberOfAnimal1;i++) {
            Animal1 iNeedItToHaveAName = new Animal1(giveMeACanNumber(), 0, false, false);
            theListOfAllTheAnimals.add(iNeedItToHaveAName);
            //          System.out.println("Spawned Animal1 with trait canEat at: " + iNeedItToHaveANa
        }
        System.out.println();
        for(int i=0;i<initialNumberOfAnimal2;i++) {
            Animal2 iNeedItToHaveAName = new Animal2(giveMeACanNumber()*1.5, giveMeACanNumber()*1, 0,
            theListOfAllTheAnimals.add(iNeedItToHaveAName);
            //          System.out.println("Spawned Animal2 with trait canRunAway at: " + iNeedItToHav
        }
        System.out.println();
        for(int i=0;i<initialNumberOfAnimal3;i++) {
            Animal3 iNeedItToHaveAName = new Animal3(giveMeACanNumber()*1.5, 0, 0, false, false);
            theListOfAllTheAnimals.add(iNeedItToHaveAName);
```

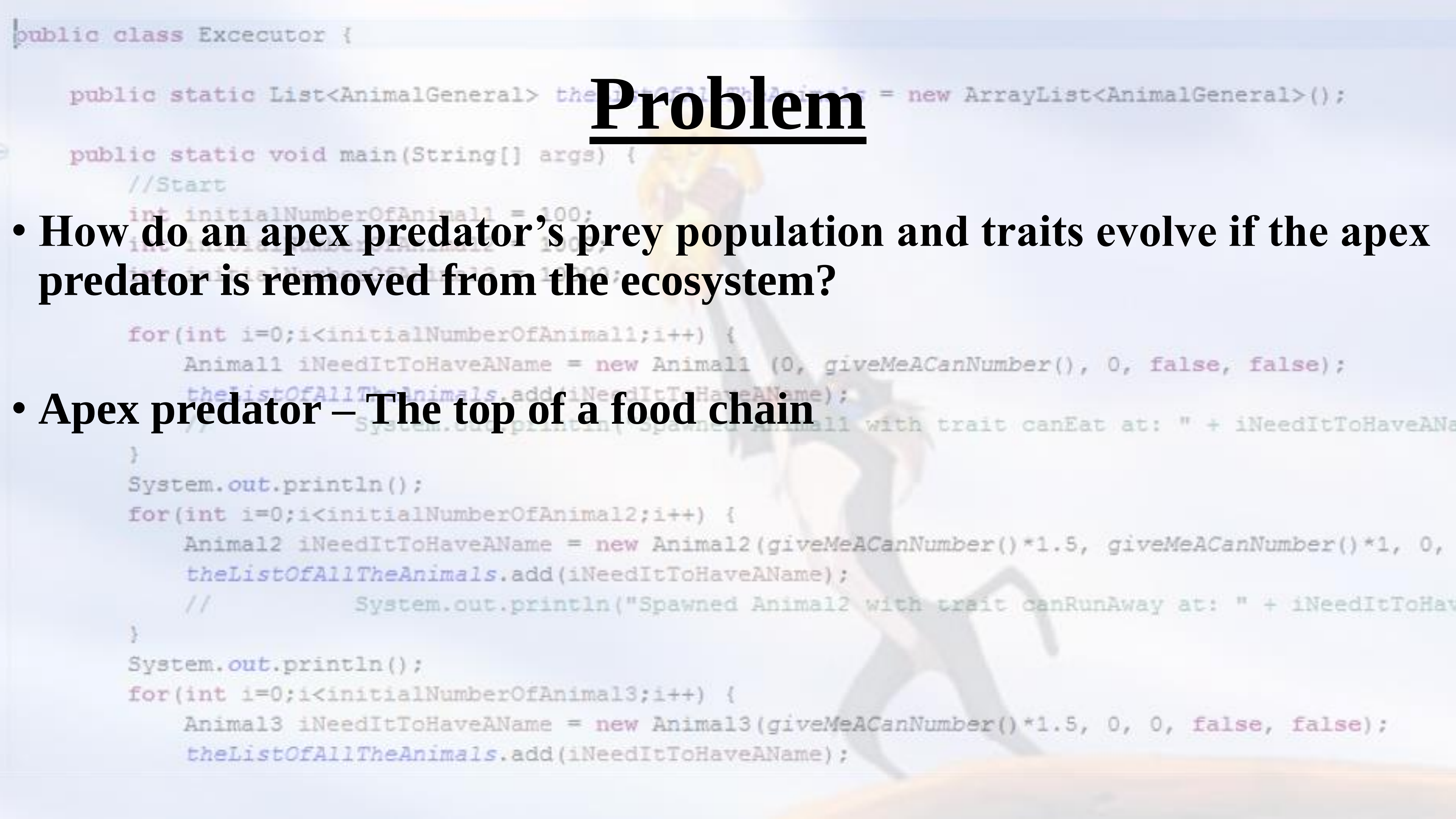




```
public class Excecutor {

    public static List<AnimalGeneral> theListOfAllTheAnimals = new ArrayList<AnimalGeneral>();

    public static void main(String[] args) {
        //Start
        int initialNumberOfAnimal1 = 100;
        int initialNumberOfAnimal2 = 1000;
        int initialNumberOfAnimal3 = 10000;

        for(int i=0;i<initialNumberOfAnimal1;i++) {
            Animal1 iNeedItToHaveAName = new Animal1 (0, giveMeACanNumber(), 0, false, false);
            theListOfAllTheAnimals.add(iNeedItToHaveAName);
            //          System.out.println("Spawned Animal1 with trait canEat at: " + iNeedItToHaveAName.
        }
        System.out.println();
        for(int i=0;i<initialNumberOfAnimal2;i++) {
            Animal2 iNeedItToHaveAName = new Animal2(giveMeACanNumber()*1.5, giveMeACanNumber()*1, 0, fal
            theListOfAllTheAnimals.add(iNeedItToHaveAName);
            //          System.out.println("Spawned Animal2 with trait canRunAway at: " + iNeedItToHaveAN
        }
        System.out.println();
        for(int i=0;i<initialNumberOfAnimal3;i++) {
            Animal3 iNeedItToHaveAName = new Animal3(giveMeACanNumber()*1.5, 0, 0, false, false);
            theListOfAllTheAnimals.add(iNeedItToHaveAName);
```

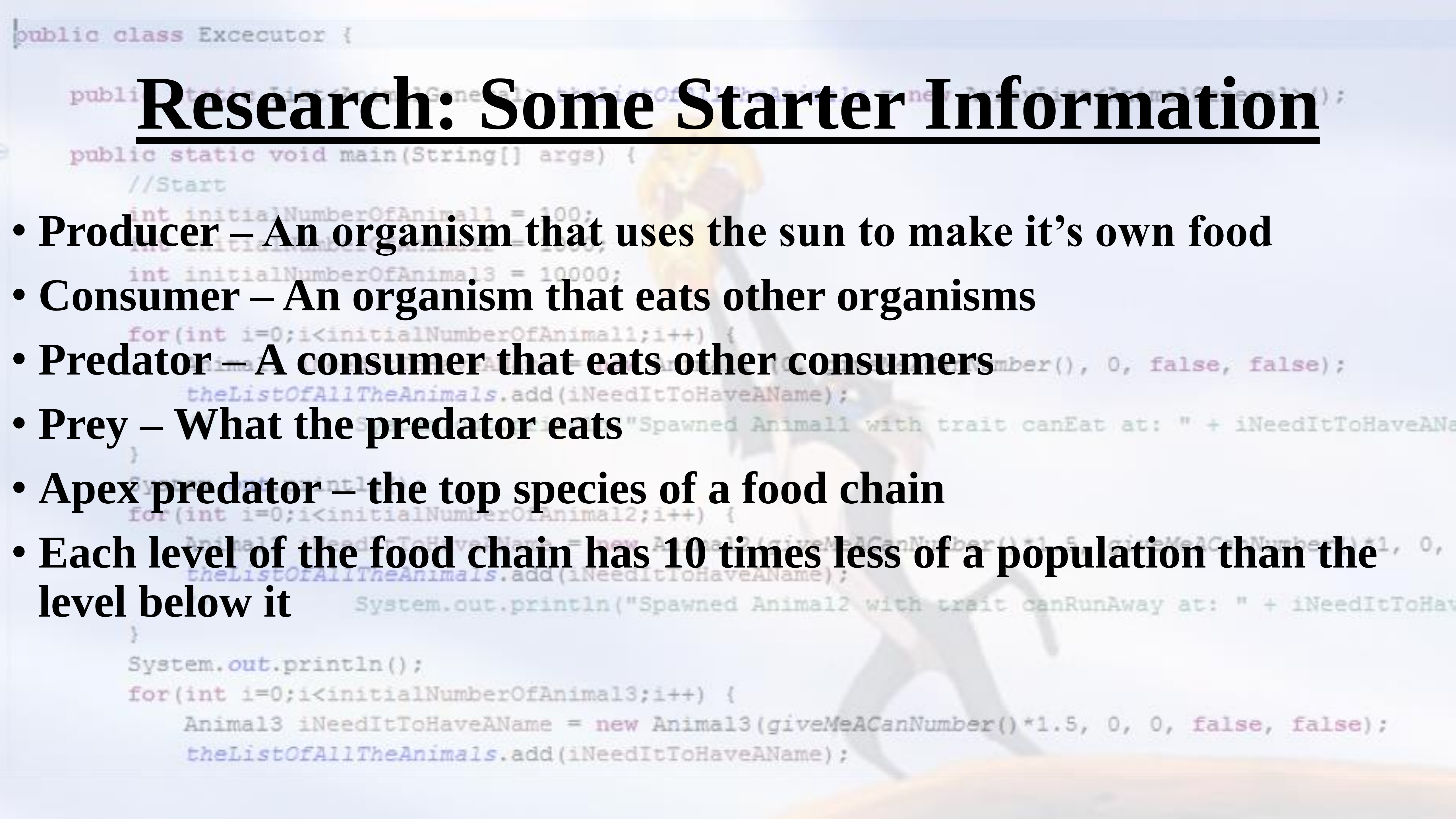
Problem

- **How do an apex predator’s prey population and traits evolve if the apex predator is removed from the ecosystem?**
- **Apex predator – The top of a food chain**



```
public class Excecutor {

    public static List<AnimalGeneral> theListOfAllTheAnimals = new ArrayList<AnimalGeneral>();

    public static void main(String[] args) {
        //Start
        int initialNumberOfAnimal1 = 100;
        int initialNumberOfAnimal2 = 100;
        int initialNumberOfAnimal3 = 1000;

        for(int i=0;i<initialNumberOfAnimal1;i++) {
            Animal1 iNeedItToHaveAName = new Animal1 (0, giveMeACanNumber(), 0, false, false);
            theListOfAllTheAnimals.add(iNeedItToHaveAName);
            //          System.out.println("Spawned Animal1 with trait canEat at: " + iNeedItToHaveANa
        }
        System.out.println();
        for(int i=0;i<initialNumberOfAnimal2;i++) {
            Animal2 iNeedItToHaveAName = new Animal2(giveMeACanNumber()*1.5, giveMeACanNumber()*1, 0,
            theListOfAllTheAnimals.add(iNeedItToHaveAName);
            //          System.out.println("Spawned Animal2 with trait canRunAway at: " + iNeedItToHav
        }
        System.out.println();
        for(int i=0;i<initialNumberOfAnimal3;i++) {
            Animal3 iNeedItToHaveAName = new Animal3(giveMeACanNumber()*1.5, 0, 0, false, false);
            theListOfAllTheAnimals.add(iNeedItToHaveAName);
```

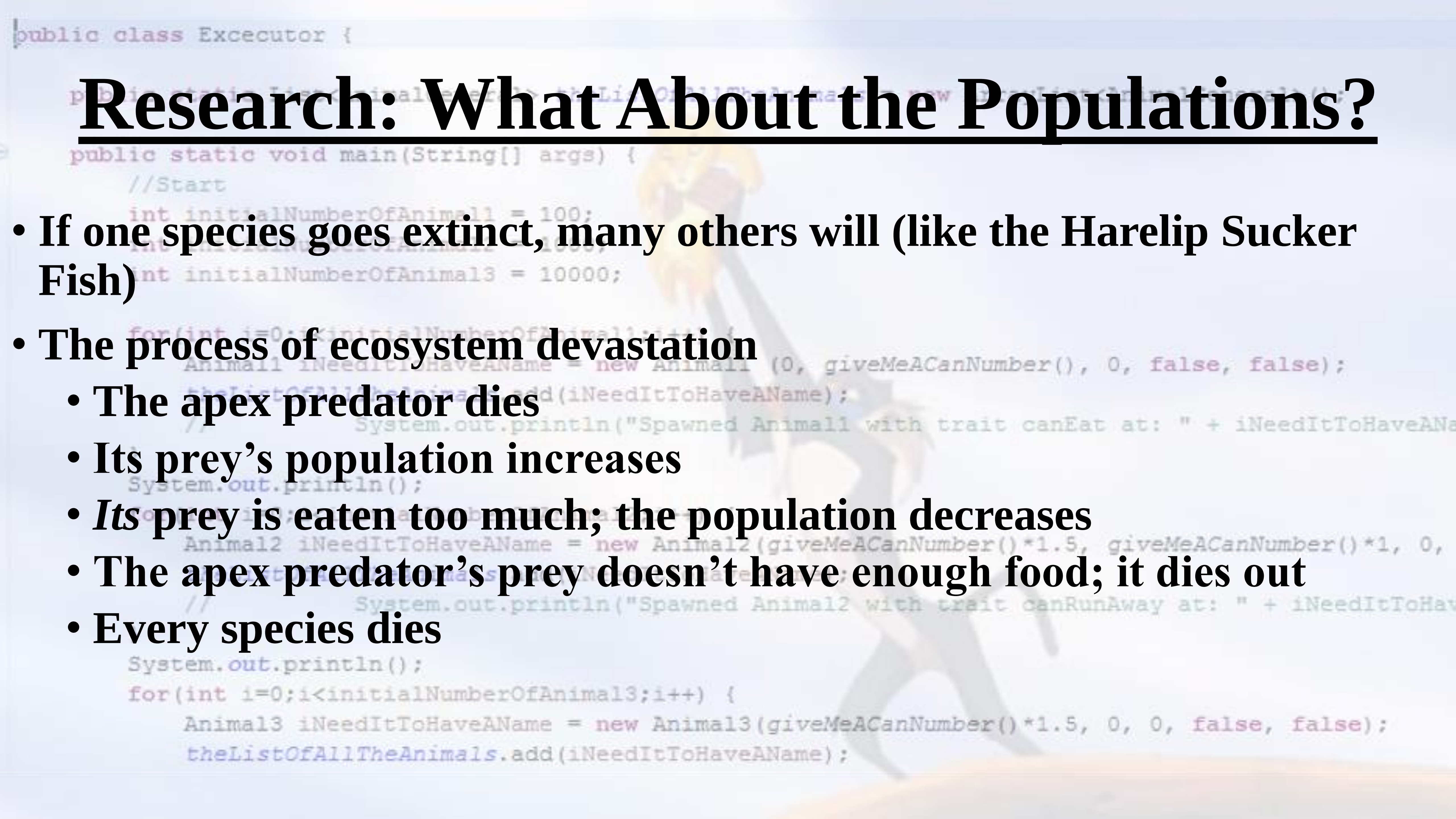
Research: Some Starter Information



- **Producer – An organism that uses the sun to make it’s own food**
- **Consumer – An organism that eats other organisms**
- **Predator – A consumer that eats other consumers**
- **Prey – What the predator eats**
- **Apex predator – the top species of a food chain**
- **Each level of the food chain has 10 times less of a population than the level below it**

Research: What About the Populations?

- If one species goes extinct, many others will (like the Harelip Sucker Fish)
- The process of ecosystem devastation
  - The apex predator dies
  - Its prey’s population increases
  - *Its* prey is eaten too much; the population decreases
  - The apex predator’s prey doesn’t have enough food; it dies out
  - Every species dies



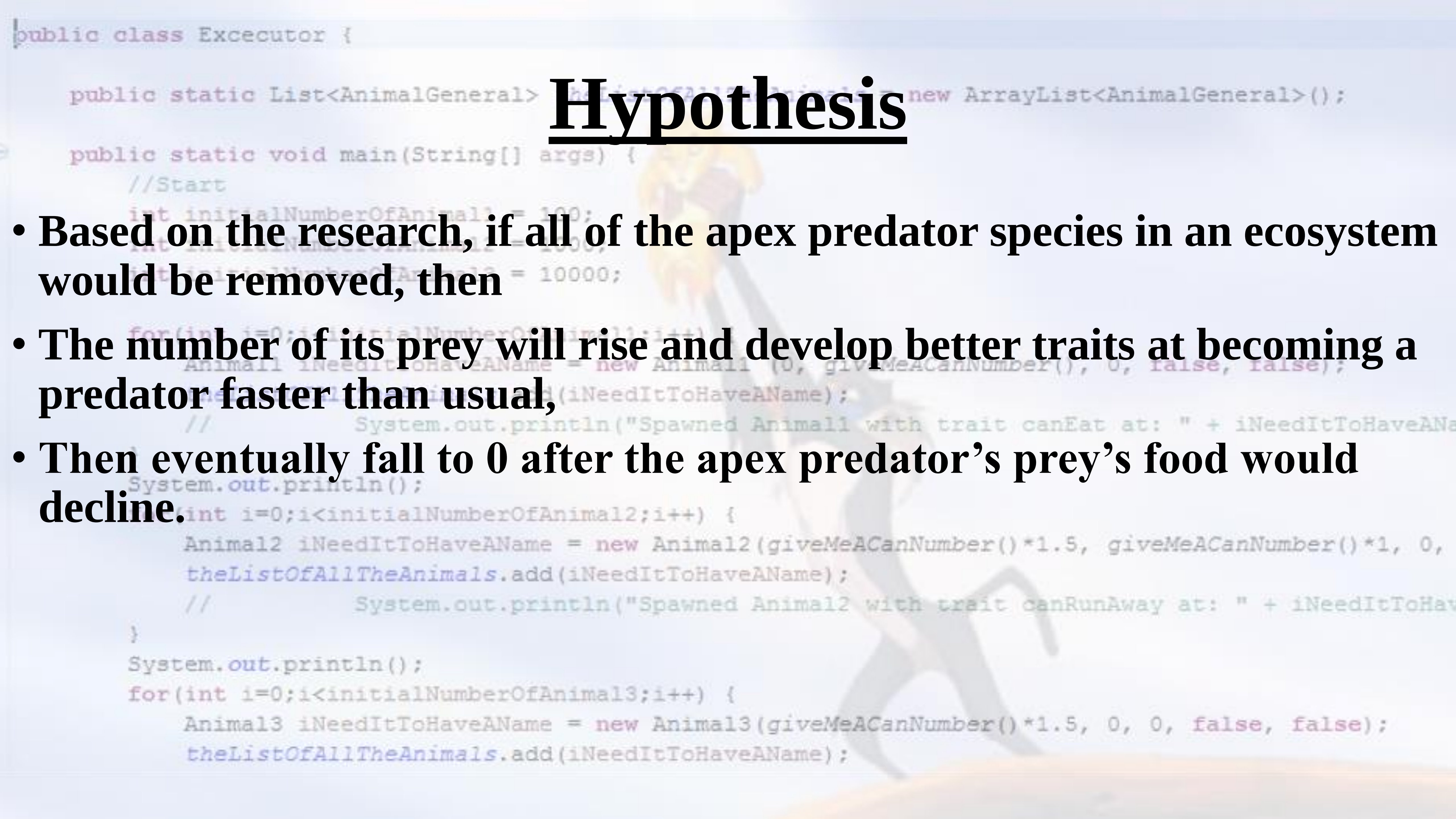
```
public class Excecutor {

public static void main(String[] args) {
    //Start
    int initialNumberOfAnimal1 = 100;

    int initialNumberOfAnimal3 = 10000;

        Animal1 iNeedItToHaveAName = new Animal1 (0, giveMeACanNumber(), 0, false, false);
                                  .add(iNeedItToHaveAName);
    //      System.out.println("Spawned Animal1 with trait canEat at: " + iNeedItToHaveANa

    System.out.println();

        Animal2 iNeedItToHaveAName = new Animal2(giveMeACanNumber()*1.5, giveMeACanNumber()*1, 0,

    //      System.out.println("Spawned Animal2 with trait canRunAway at: " + iNeedItToHav

    System.out.println();
    for(int i=0;i<initialNumberOfAnimal3;i++) {
        Animal3 iNeedItToHaveAName = new Animal3(giveMeACanNumber()*1.5, 0, 0, false, false);
        theListOfAllTheAnimals.add(iNeedItToHaveAName);
```

Research: What About the Traits?

- The study of the three-spined sticklebacks
  - Without prey, their numbers increased
- UC Santa Cruz’s study of animals that humans killed off
  - When that happened, the process described earlier occurred every time
  - It happened with wolves in Yellowstone, lions and leopards, wildebeests, sea otters, sharks, and whales (whose plankton they did not each resulted in 105 million tons of CO_2)
- The study of the Trinidadian guppies
  - They have a natural predator
  - Without the predator, the number of guppies increases, and it can get food in a better way (sharper teeth), making it a better predator

Hypothesis

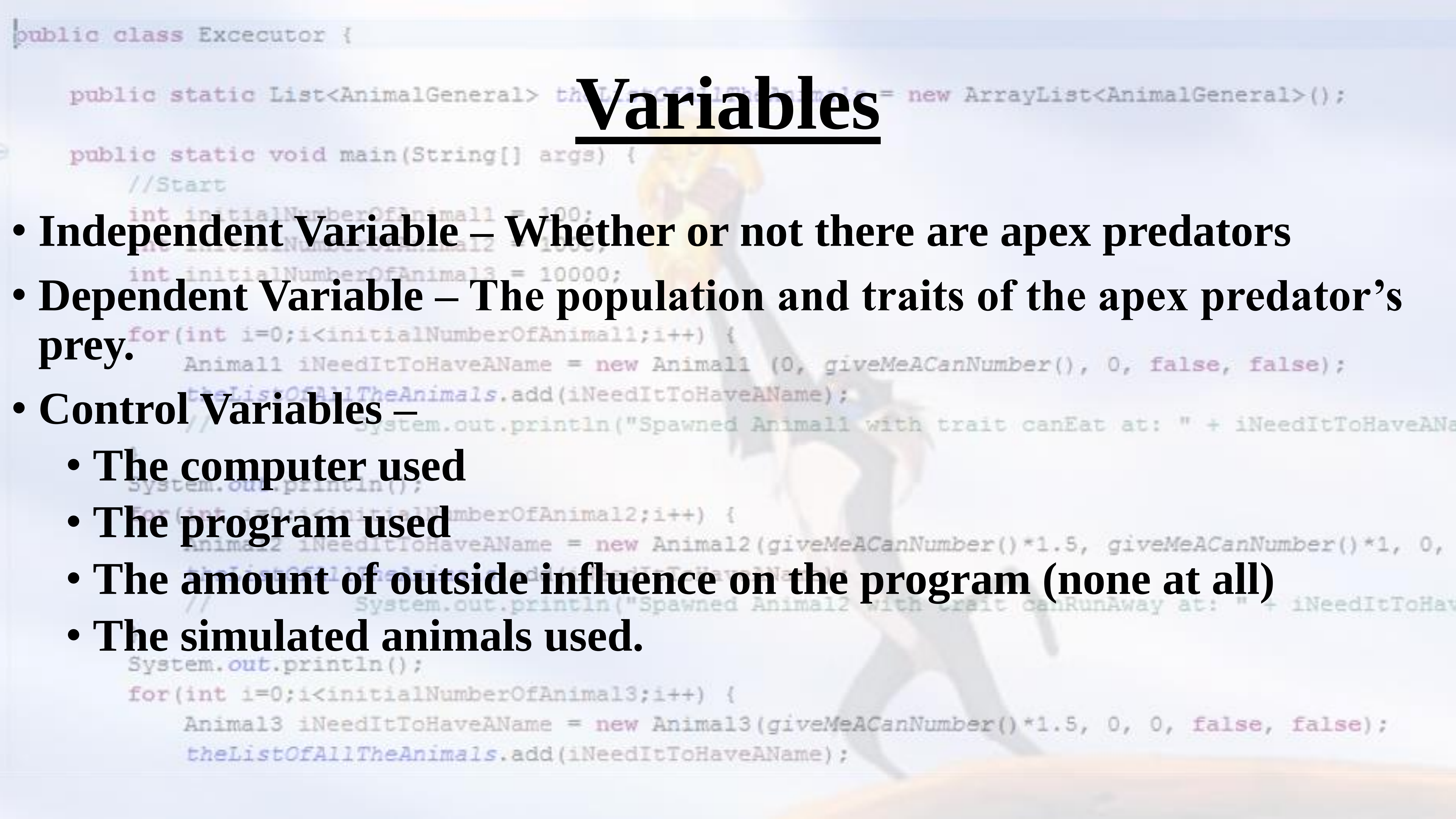
- Based on the research, if all of the apex predator species in an ecosystem would be removed, then
- The number of its prey will rise and develop better traits at becoming a predator faster than usual,
- Then eventually fall to 0 after the apex predator’s prey’s food would decline.



```
public class Excecutor {

    public static List<AnimalGeneral> theListOfAllTheAnimals = new ArrayList<AnimalGeneral>();

    public static void main(String[] args) {
        //Start
        int initialNumberOfAnimal1 = 100;
        int initialNumberOfAnimal2 = 1000;
        int initialNumberOfAnimal3 = 10000;

        for(int i=0;i<initialNumberOfAnimal1;i++) {
            Animal1 iNeedItToHaveAName = new Animal1 (0, giveMeACanNumber(), 0, false, false);
            theListOfAllTheAnimals.add(iNeedItToHaveAName);
            //          System.out.println("Spawned Animal1 with trait canEat at: " + iNeedItToHaveANa
        }
        System.out.println();
        for(int i=0;i<initialNumberOfAnimal2;i++) {
            Animal2 iNeedItToHaveAName = new Animal2(giveMeACanNumber()*1.5, giveMeACanNumber()*1, 0,
            theListOfAllTheAnimals.add(iNeedItToHaveAName);
            //          System.out.println("Spawned Animal2 with trait canRunAway at: " + iNeedItToHav
        }
        System.out.println();
        for(int i=0;i<initialNumberOfAnimal3;i++) {
            Animal3 iNeedItToHaveAName = new Animal3(giveMeACanNumber()*1.5, 0, 0, false, false);
            theListOfAllTheAnimals.add(iNeedItToHaveAName);
```

Variables

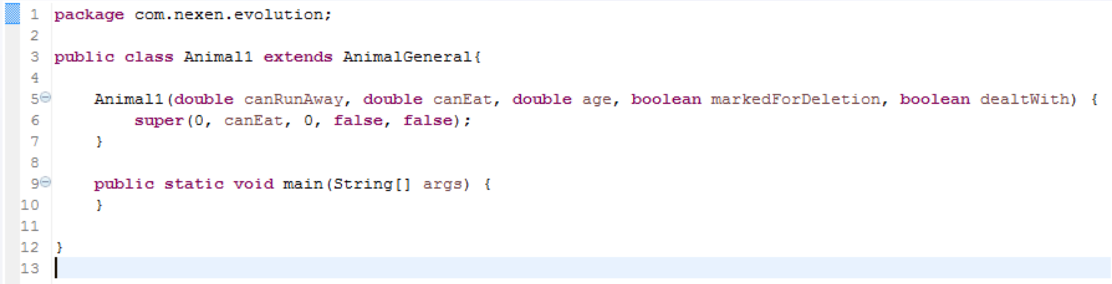
- Independent Variable – Whether or not there are apex predators
- Dependent Variable – The population and traits of the apex predator's prey.
- Control Variables –
  - The computer used
  - The program used
  - The amount of outside influence on the program (none at all)
  - The simulated animals used.



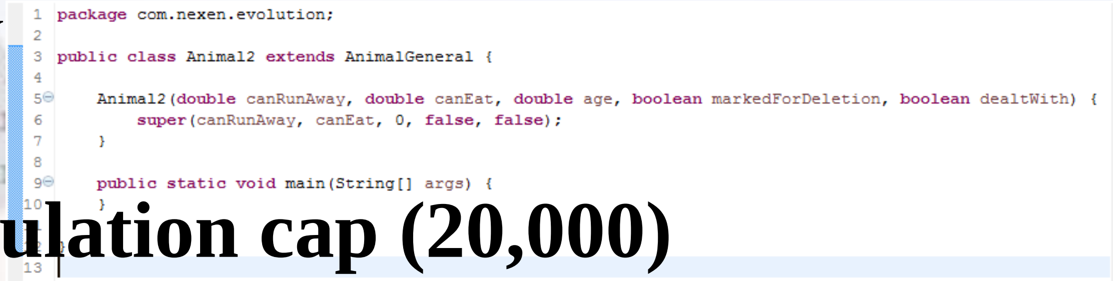
```
public class Excecutor {

    public static List<AnimalGeneral> theListOfAllTheAnimals = new ArrayList<AnimalGeneral>();

    public static void main(String[] args) {
        //Start
        int initialNumberOfAnimal1 = 100;
        int initialNumberOfAnimal2 = 1000;
        int initialNumberOfAnimal3 = 10000;

        for(int i=0;i<initialNumberOfAnimal1;i++) {
            Animal1 iNeedItToHaveAName = new Animal1 (0, giveMeACanNumber(), 0, false, false);
            theListOfAllTheAnimals.add(iNeedItToHaveAName);
        //      System.out.println("Spawned Animal1 with trait canEat at: " + iNeedItToHaveANa
        }
        System.out.println();
        for(int i=0;i<initialNumberOfAnimal2;i++) {
            Animal2 iNeedItToHaveAName = new Animal2(giveMeACanNumber()*1.5, giveMeACanNumber()*1, 0,
            theListOfAllTheAnimals.add(iNeedItToHaveAName);
        //      System.out.println("Spawned Animal2 with trait canRunAway at: " + iNeedItToHav
        }
        System.out.println();
        for(int i=0;i<initialNumberOfAnimal3;i++) {
            Animal3 iNeedItToHaveAName = new Animal3(giveMeACanNumber()*1.5, 0, 0, false, false);
            theListOfAllTheAnimals.add(iNeedItToHaveAName);
```

Experiment: Setup I: The Fundamentals

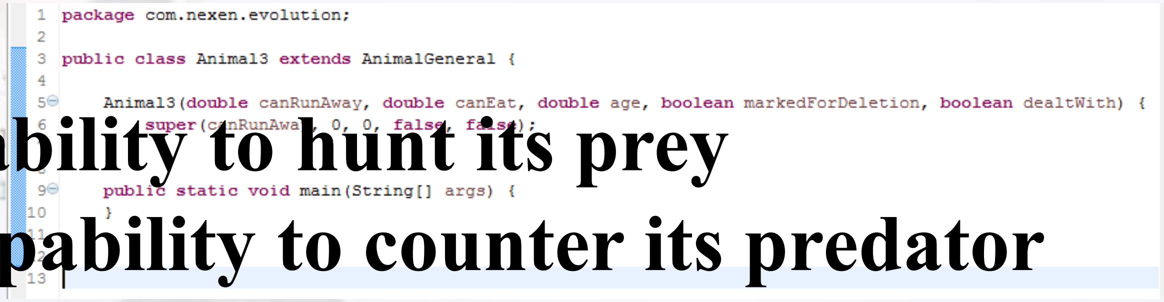
- The Animals
  - Animal1 – The apex predator
  - Animal2 – The apex predator’s prey
  - Animal3 – The bottom animal
  - The producers – Animal 3 has a population cap (20,000)
- Concepts
  - canEat - simulates a predator’s capability to hunt its prey
  - canRunAway - simulates a prey’s capability to counter its predator



```java
package com.nexen.evolution;

public class Animal1 extends AnimalGeneral{

    Animal1(double canRunAway, double canEat, double age, boolean markedForDeletion, boolean dealtWith) {
        super(0, canEat, 0, false, false);
    }

    public static void main(String[] args) {
    }

}
```



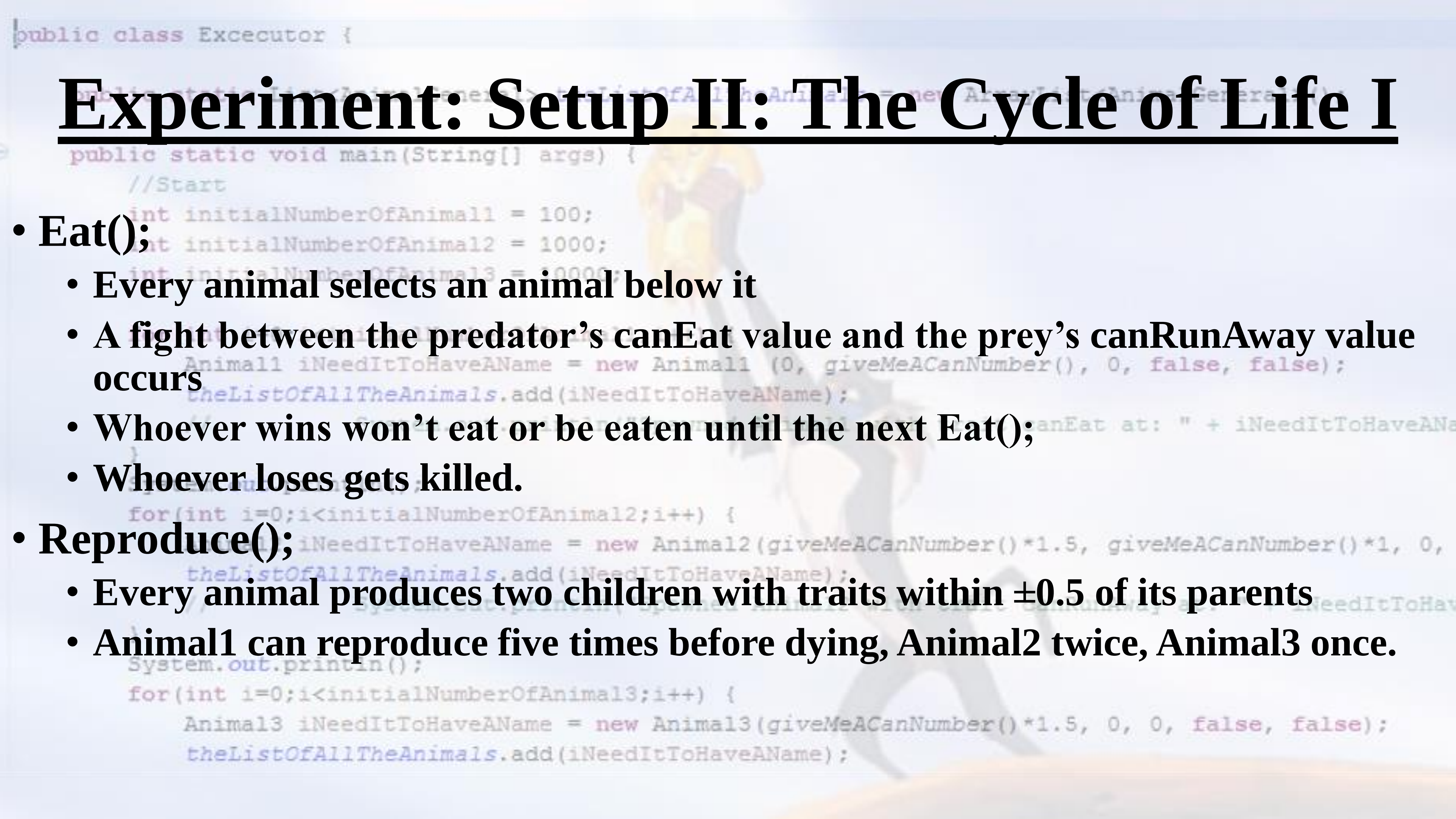
```java
package com.nexen.evolution;

public class Animal2 extends AnimalGeneral {

    Animal2(double canRunAway, double canEat, double age, boolean markedForDeletion, boolean dealtWith) {
        super(canRunAway, canEat, 0, false, false);
    }

    public static void main(String[] args) {
    }
```



```java
package com.nexen.evolution;

public class Animal3 extends AnimalGeneral {

    Animal3(double canRunAway, double canEat, double age, boolean markedForDeletion, boolean dealtWith) {
        super(canRunAway, 0, 0, false, false);

    public static void main(String[] args) {
    }
```

Experiment: Setup II: The Cycle of Life I

- Eat();
  - Every animal selects an animal below it
  - A fight between the predator’s canEat value and the prey’s canRunAway value occurs
  - Whoever wins won’t eat or be eaten until the next Eat();
  - Whoever loses gets killed.
- Reproduce();
  - Every animal produces two children with traits within ±0.5 of its parents
  - Animal1 can reproduce five times before dying, Animal2 twice, Animal3 once.



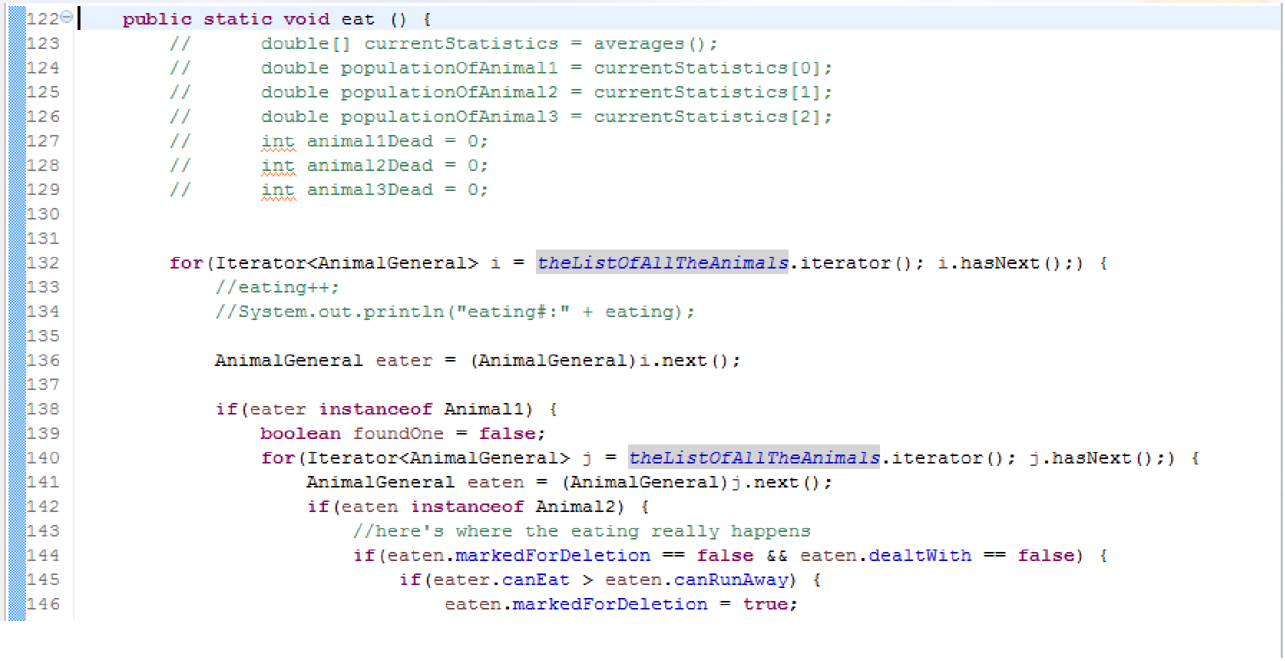
```
public class Excecutor {

    public static List<AnimalGeneral> theListOfAllTheAnimals = new ArrayList<AnimalGeneral>();

    public static void main(String[] args) {
        //Start
        int initialNumberOfAnimal1 = 100;
        int initialNumberOfAnimal2 = 1000;
        int initialNumberOfAnimal3 = 10000;
        for(int i=0;i<initialNumberOfAnimal1;i++) {
            Animal1 iNeedItToHaveAName = new Animal1 (0, giveMeACanNumber(), 0, false, false);
            theListOfAllTheAnimals.add(iNeedItToHaveAName);
            //          System.out.println("Spawned Animal1 with canEat at: " + iNeedItToHaveANa
        }
        System.out.println();
        for(int i=0;i<initialNumberOfAnimal2;i++) {
            Animal2 iNeedItToHaveAName = new Animal2(giveMeACanNumber()*1.5, giveMeACanNumber()*1, 0,
            theListOfAllTheAnimals.add(iNeedItToHaveAName);
            //          System.out.println("Spawned Animal2 with canRunAway at: " + iNeedItToHav
        }
        System.out.println();
        for(int i=0;i<initialNumberOfAnimal3;i++) {
            Animal3 iNeedItToHaveAName = new Animal3(giveMeACanNumber()*1.5, 0, 0, false, false);
            theListOfAllTheAnimals.add(iNeedItToHaveAName);
```

Experiment: Setup III: The Cycle of Life II

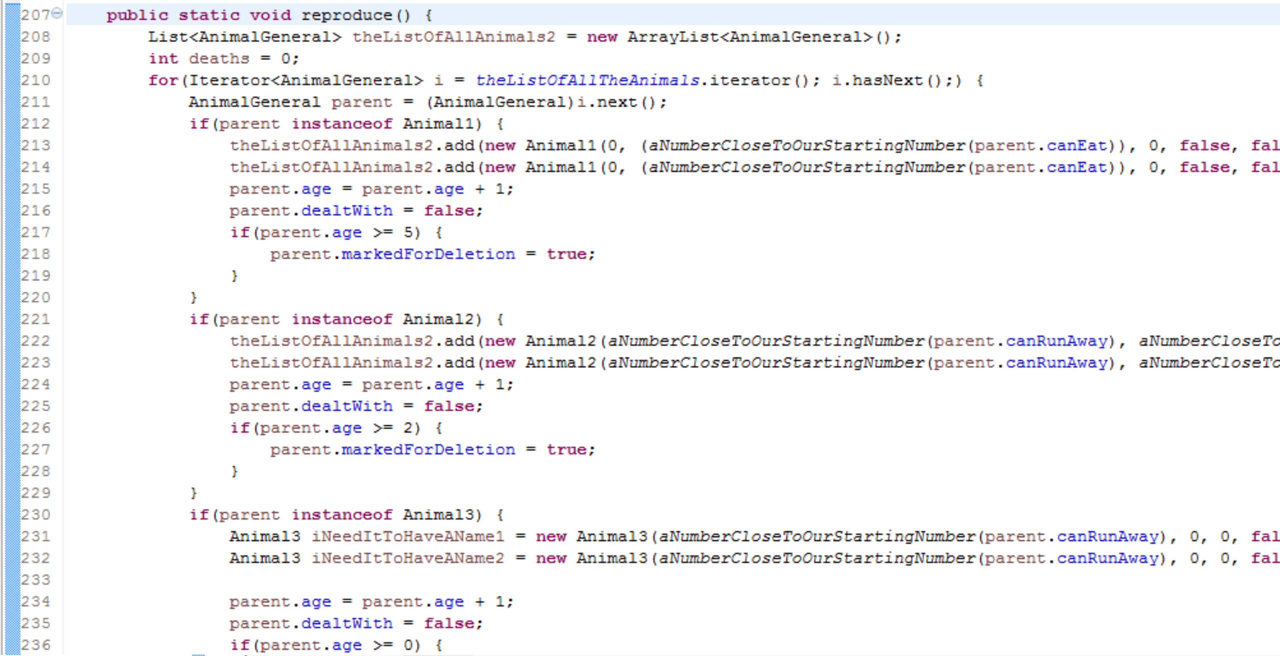
- **Survey**
  - **The population of every species is recorded**
  - **The average canEat and canRunAway for each species is recorded**
- **Control trial (1 count)**
  - **300 cycles of this process (generations)**
- **Real trial (10 count)**
  - **100 generations, kill every Animal1, then 200 more generations**

Pictures of the Program

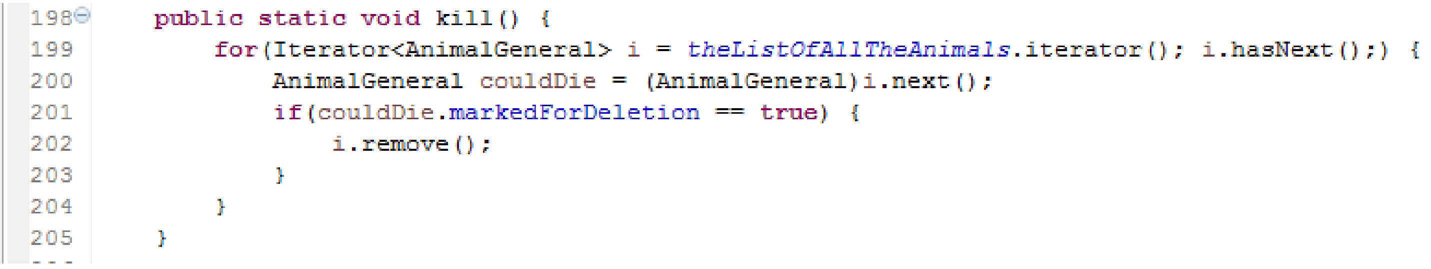
```
public class Excecutor {

    public static List<AnimalGeneral> theListOfAllTheAnimals = new ArrayList<AnimalGeneral>();
```



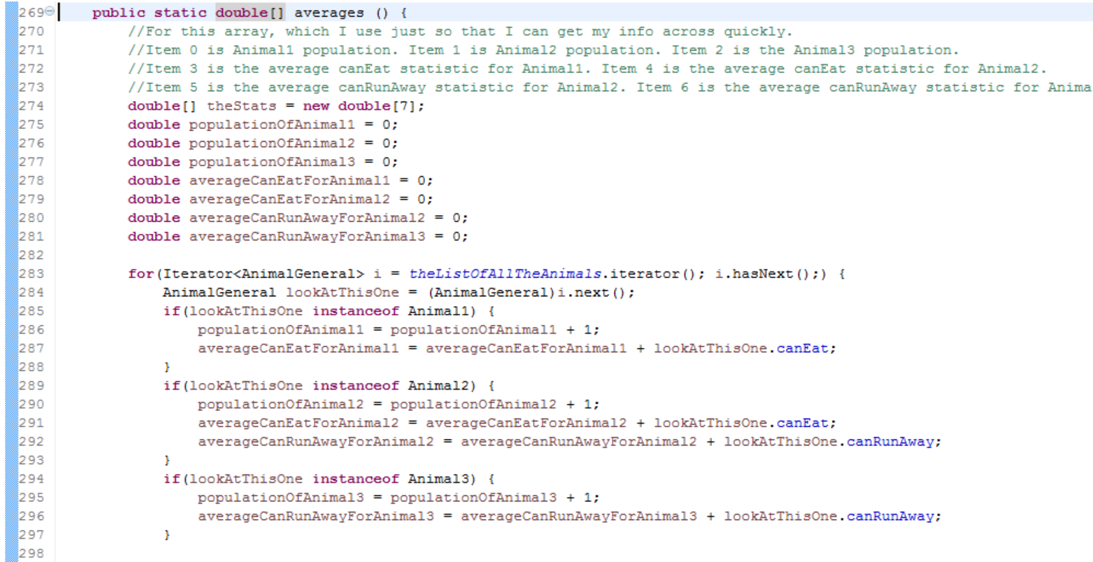
```
    public static void eat () {
        //      double[] currentStatistics = averages();
        //      double populationOfAnimal1 = currentStatistics[0];
        //      double populationOfAnimal2 = currentStatistics[1];
        //      double populationOfAnimal3 = currentStatistics[2];
        //      int animal1Dead = 0;
        //      int animal2Dead = 0;
        //      int animal3Dead = 0;


        for(Iterator<AnimalGeneral> i = theListOfAllTheAnimals.iterator(); i.hasNext();) {
            //eating++;
            //System.out.println("eating#:" + eating);

            AnimalGeneral eater = (AnimalGeneral)i.next();

            if(eater instanceof Animal1) {
                boolean foundOne = false;
                for(Iterator<AnimalGeneral> j = theListOfAllTheAnimals.iterator(); j.hasNext();) {
                    AnimalGeneral eaten = (AnimalGeneral)j.next();
                    if(eaten instanceof Animal2) {
                        //here's where the eating really happens
                        if(eaten.markedForDeletion == false && eaten.dealtWith == false) {
                            if(eater.canEat > eaten.canRunAway) {
                                eaten.markedForDeletion = true;
```



```
    public static void reproduce() {
        List<AnimalGeneral> theListOfAllAnimals2 = new ArrayList<AnimalGeneral>();
        int deaths = 0;
        for(Iterator<AnimalGeneral> i = theListOfAllTheAnimals.iterator(); i.hasNext();) {
            AnimalGeneral parent = (AnimalGeneral)i.next();
            if(parent instanceof Animal1) {
                theListOfAllAnimals2.add(new Animal1(0, (aNumberCloseToOurStartingNumber(parent.canEat)), 0, false, fal
                theListOfAllAnimals2.add(new Animal1(0, (aNumberCloseToOurStartingNumber(parent.canEat)), 0, false, fal
                parent.age = parent.age + 1;
                parent.dealtWith = false;
                if(parent.age >= 5) {
                    parent.markedForDeletion = true;
                }
            }
            if(parent instanceof Animal2) {
                theListOfAllAnimals2.add(new Animal2(aNumberCloseToOurStartingNumber(parent.canRunAway), aNumberCloseTo
                theListOfAllAnimals2.add(new Animal2(aNumberCloseToOurStartingNumber(parent.canRunAway), aNumberCloseTo
                parent.age = parent.age + 1;
                parent.dealtWith = false;
                if(parent.age >= 2) {
                    parent.markedForDeletion = true;
                }
            }
            if(parent instanceof Animal3) {
                Animal3 iNeedItToHaveAName1 = new Animal3(aNumberCloseToOurStartingNumber(parent.canRunAway), 0, 0, fal
                Animal3 iNeedItToHaveAName2 = new Animal3(aNumberCloseToOurStartingNumber(parent.canRunAway), 0, 0, fal

                parent.age = parent.age + 1;
                parent.dealtWith = false;
                if(parent.age >= 0) {
```



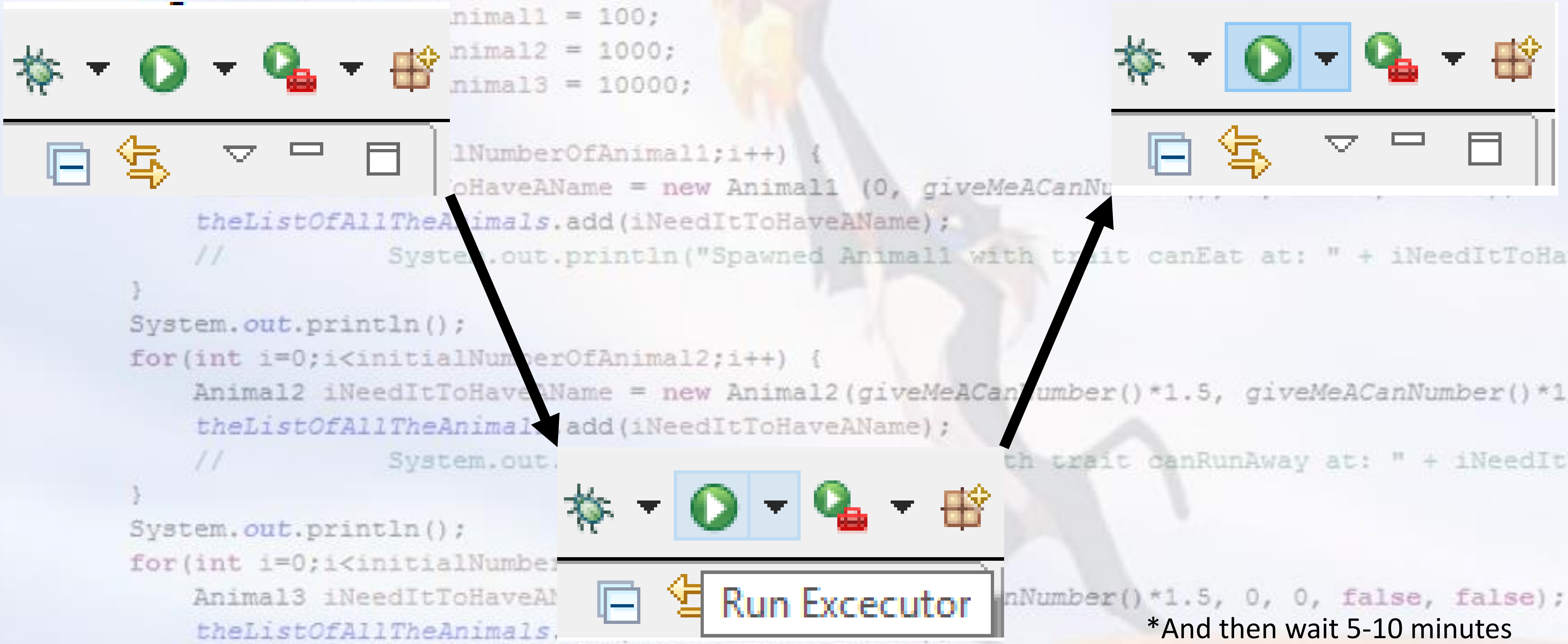
```
    public static void kill() {
        for(Iterator<AnimalGeneral> i = theListOfAllTheAnimals.iterator(); i.hasNext();) {
            AnimalGeneral couldDie = (AnimalGeneral)i.next();
            if(couldDie.markedForDeletion == true) {
                i.remove();
            }
        }
    }
```



```
    public static double[] averages () {
        //For this array, which I use just so that I can get my info across quickly.
        //Item 0 is Animal1 population. Item 1 is Animal2 population. Item 2 is the Animal3 population.
        //Item 3 is the average canEat statistic for Animal1. Item 4 is the average canEat statistic for Animal2.
        //Item 5 is the average canRunAway statistic for Animal2. Item 6 is the average canRunAway statistic for Anima
        double[] theStats = new double[7];
        double populationOfAnimal1 = 0;
        double populationOfAnimal2 = 0;
        double populationOfAnimal3 = 0;
        double averageCanEatForAnimal1 = 0;
        double averageCanEatForAnimal2 = 0;
        double averageCanRunAwayForAnimal2 = 0;
        double averageCanRunAwayForAnimal3 = 0;

        for(Iterator<AnimalGeneral> i = theListOfAllTheAnimals.iterator(); i.hasNext();) {
            AnimalGeneral lookAtThisOne = (AnimalGeneral)i.next();
            if(lookAtThisOne instanceof Animal1) {
                populationOfAnimal1 = populationOfAnimal1 + 1;
                averageCanEatForAnimal1 = averageCanEatForAnimal1 + lookAtThisOne.canEat;
            }
            if(lookAtThisOne instanceof Animal2) {
                populationOfAnimal2 = populationOfAnimal2 + 1;
                averageCanEatForAnimal2 = averageCanEatForAnimal2 + lookAtThisOne.canEat;
                averageCanRunAwayForAnimal2 = averageCanRunAwayForAnimal2 + lookAtThisOne.canRunAway;
            }
            if(lookAtThisOne instanceof Animal3) {
                populationOfAnimal3 = populationOfAnimal3 + 1;
                averageCanRunAwayForAnimal3 = averageCanRunAwayForAnimal3 + lookAtThisOne.canRunAway;
            }

```

Experiment: Execution

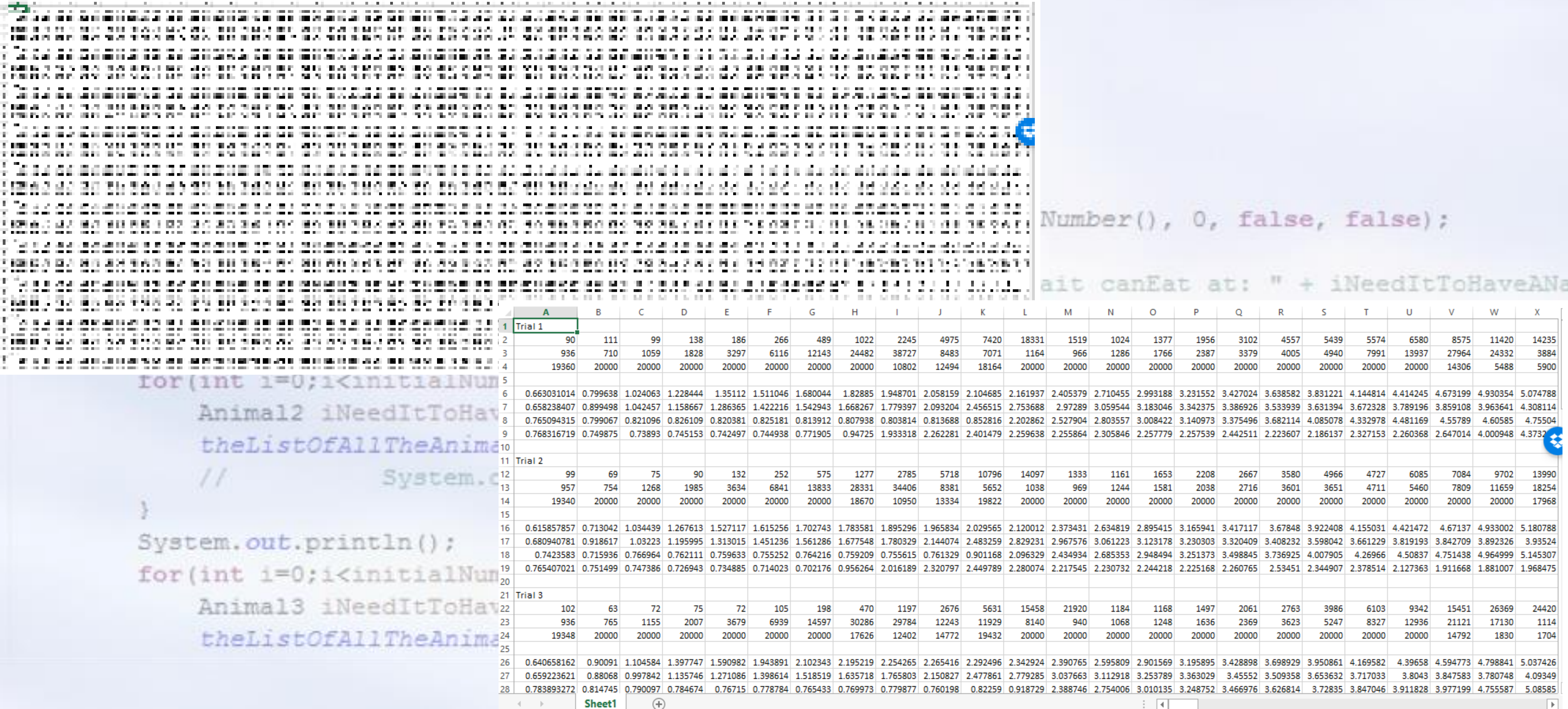
```
public class Excecutor {

    public static List<AnimalGeneral> theListOfAllTheAnimals = new ArrayList<AnimalGeneral>();

    public static void main(String[] args) {
        //Start
                              nimal1 = 100;
                              nimal2 = 1000;
                              nimal3 = 10000;

                              lNumberOfAnimal1;i++) {
                              oHaveAName = new Animal1 (0, giveMeACanNu
            theListOfAllTheAnimals.add(iNeedItToHaveAName);
            //          System.out.println("Spawned Animal1 with trait canEat at: " + iNeedItToHaveANa
        }
        System.out.println();
        for(int i=0;i<initialNumberOfAnimal2;i++) {
            Animal2 iNeedItToHaveAName = new Animal2(giveMeACanNumber()*1.5, giveMeACanNumber()*1, 0,
            theListOfAllTheAnimals.add(iNeedItToHaveAName);
            //          System.out.                               th trait canRunAway at: " + iNeedItToHav
        }
        System.out.println();
        for(int i=0;i<initialNumbe
            Animal3 iNeedItToHaveAN                                nNumber()*1.5, 0, 0, false, false);
            theListOfAllTheAnimals.
```



*And then wait 5-10 minutes

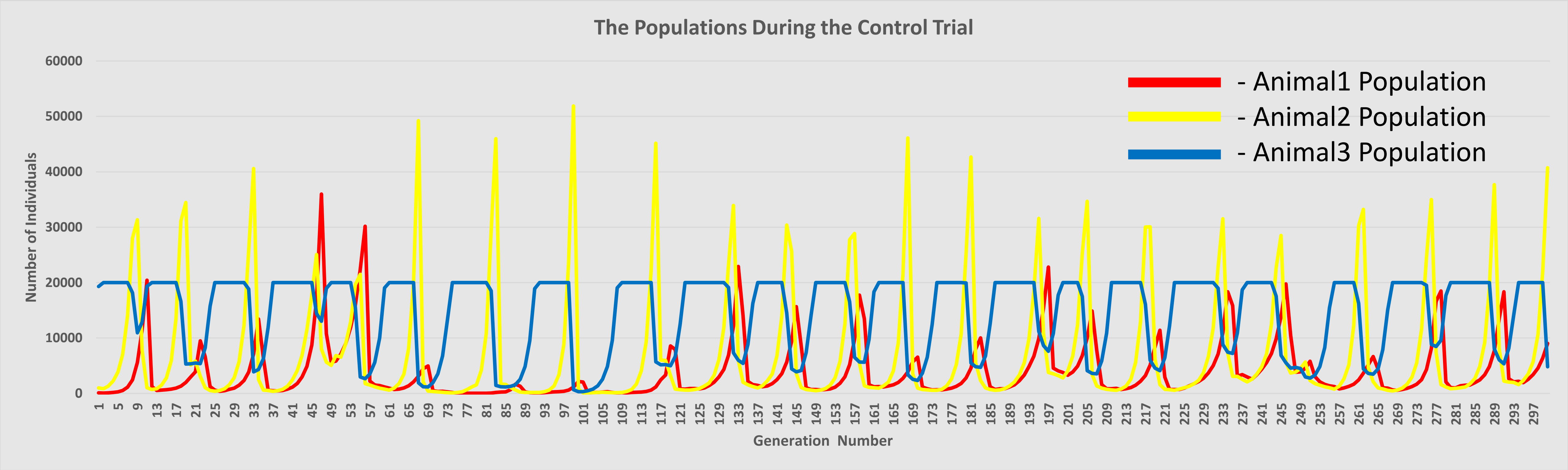
Results: The Scale

```
public class Excecutor {

    public static List<AnimalG                                      ArrayList<AnimalGeneral>();

    public static void main(String[] args) {

                                                  Number(), 0, false, false);

                                                  ait canEat at: " + iNeedItToHaveANa

        for(int i=0;i<initialNum
            Animal2 iNeedItToHav
            theListOfAllTheAnima
            //             System.c
        }
        System.out.println();
        for(int i=0;i<initialNum
            Animal3 iNeedItToHav
            theListOfAllTheAnima
```



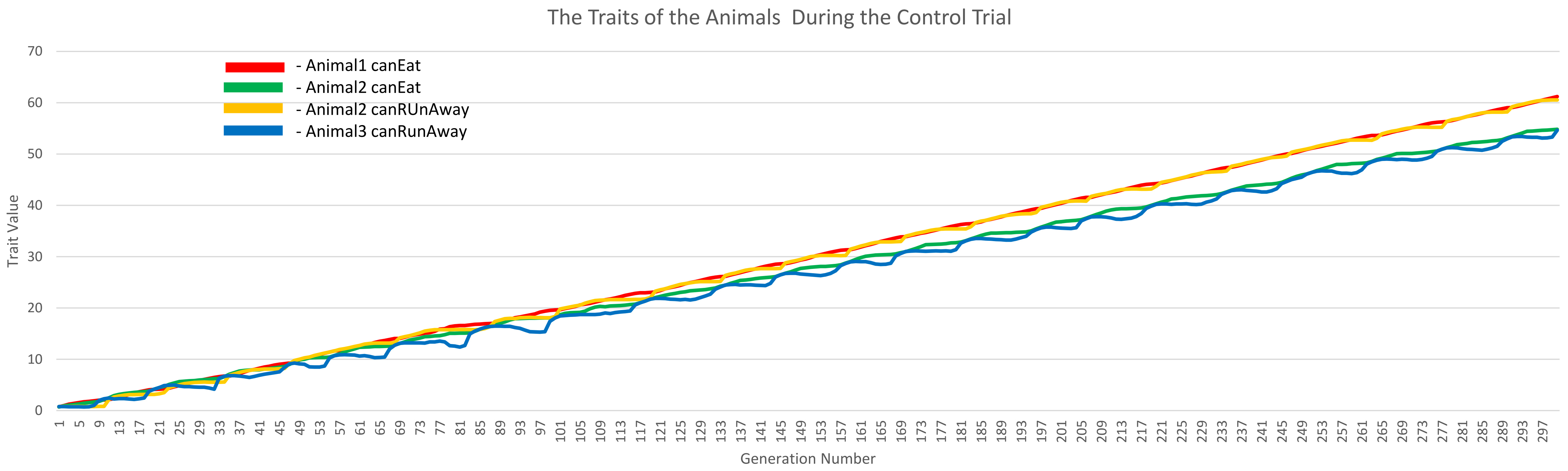
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
1	Trial 1																							
2	90	111	99	138	186	266	489	1022	2245	4975	7420	18331	1519	1024	1377	1956	3102	4557	5439	5574	6580	8575	11420	14235
3	936	710	1059	1828	3297	6116	12143	24482	38727	8483	7071	1164	966	1286	1766	2387	3379	4005	4940	7991	13937	27964	24332	3884
4	19360	20000	20000	20000	20000	20000	20000	20000	10802	12494	18164	20000	20000	20000	20000	20000	20000	20000	20000	20000	20000	14306	5488	5900
5																								
6	0.663031014	0.799638	1.024063	1.228444	1.35112	1.511046	1.680044	1.82885	1.948701	2.058159	2.104685	2.161937	2.405379	2.710455	2.993188	3.231552	3.427024	3.638582	3.831221	4.144814	4.414245	4.673199	4.930354	5.074788
7	0.658238407	0.899498	1.042457	1.158667	1.286365	1.422216	1.542943	1.668267	1.779397	2.093204	2.456515	2.753688	2.97289	3.059544	3.183046	3.342375	3.386926	3.533939	3.631394	3.672328	3.789196	3.859108	3.963641	4.308114
8	0.765094315	0.799067	0.821096	0.826109	0.820381	0.825181	0.813912	0.807938	0.803814	0.813688	0.852816	2.202862	2.527904	2.803557	3.008422	3.140973	3.375496	3.682114	4.085078	4.332978	4.481169	4.55789	4.60585	4.75504
9	0.768316719	0.749875	0.73893	0.745153	0.742497	0.744938	0.771905	0.94725	1.933318	2.262281	2.401479	2.259638	2.255864	2.305846	2.257779	2.257539	2.442511	2.223607	2.186137	2.327153	2.260368	2.647014	4.000948	4.3732
10																								
11	Trial 2																							
12	99	69	75	90	132	252	575	1277	2785	5718	10796	14097	1333	1161	1653	2208	2667	3580	4966	4727	6085	7084	9702	13990
13	957	754	1268	1985	3634	6841	13833	28331	34406	8381	5652	1038	969	1244	1581	2038	2716	3601	3651	4711	5460	7809	11659	18254
14	19340	20000	20000	20000	20000	20000	20000	18670	10950	13334	19822	20000	20000	20000	20000	20000	20000	20000	20000	20000	20000	20000	20000	17968
15																								
16	0.615857857	0.713042	1.034439	1.267613	1.527117	1.615256	1.702743	1.783581	1.895296	1.965834	2.029565	2.120012	2.373431	2.634819	2.895415	3.165941	3.417117	3.67848	3.922408	4.155031	4.421472	4.67137	4.933002	5.180788
17	0.680940781	0.918617	1.03223	1.195995	1.313015	1.451236	1.561286	1.677548	1.780329	2.144074	2.483259	2.829231	2.967576	3.061223	3.123178	3.230303	3.320409	3.408232	3.598042	3.661229	3.819193	3.842709	3.892326	3.93524
18	0.7423583	0.715936	0.766964	0.762111	0.759633	0.755252	0.764216	0.759209	0.755615	0.761329	0.901168	2.096329	2.434934	2.685353	2.948494	3.251373	3.498845	3.736925	4.007905	4.26966	4.50837	4.751438	4.964999	5.145307
19	0.765407021	0.751499	0.747386	0.726943	0.734885	0.714023	0.702176	0.956264	2.016189	2.320797	2.449789	2.280074	2.217545	2.230732	2.244218	2.225168	2.260765	2.53451	2.344907	2.378514	2.127363	1.911668	1.881007	1.968475
20																								
21	Trial 3																							
22	102	63	72	75	72	105	198	470	1197	2676	5631	15458	21920	1184	1168	1497	2061	2763	3986	6103	9342	15451	26369	24420
23	936	765	1155	2007	3679	6939	14597	30286	29784	12243	11929	8140	940	1068	1248	1636	2369	3623	5247	8327	12936	21121	17130	1114
24	19348	20000	20000	20000	20000	20000	20000	17626	12402	14772	19432	20000	20000	20000	20000	20000	20000	20000	20000	20000	20000	14792	1830	1704
25																								
26	0.640658162	0.90091	1.104584	1.397747	1.590982	1.943891	2.102343	2.195219	2.254265	2.265416	2.292496	2.342924	2.390765	2.595809	2.901569	3.195895	3.428898	3.698929	3.950861	4.169582	4.39658	4.594773	4.798841	5.037426
27	0.659223621	0.88068	0.997842	1.135746	1.271086	1.398614	1.518519	1.635718	1.765803	2.150827	2.477861	2.779285	3.037663	3.112918	3.253789	3.363029	3.45552	3.509358	3.653632	3.717033	3.8043	3.847583	3.780748	4.09349
28	0.783893272	0.814745	0.790097	0.784674	0.76715	0.778784	0.765433	0.769973	0.779877	0.760198	0.82259	0.918729	2.388746	2.754006	3.010135	3.248752	3.466976	3.626814	3.72835	3.847046	3.911828	3.977199	4.755587	5.08585

Results: Control Trial Populations



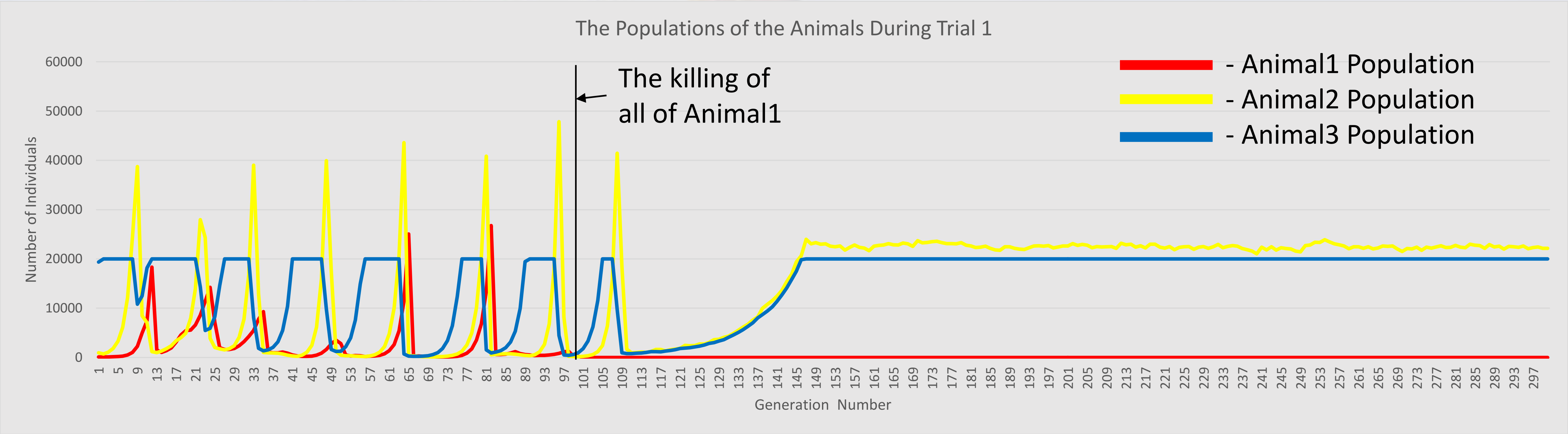
- **A regular cycle of spike, major decline, slow rebound in each species**
- **Animal3’s population cap is visible**
- **Animal2’s spikes are much higher than the other ones**

Results: Control Trial Traits



- Animal1’s canEat is almost exactly Animal2’s canRunAway
- Animal2’s canEat is almost exactly Animal3’s canRunAway
  - Animal3’s CanRunAway goes down slowly and then suddenly spikes regularly (Animal3’s population goes down)

Results: Trial 1 Populations



- **The regular cycle occurred before the killing of all of Animal1**
- **The populations exponentially rose from there**
- **Animal2's population hovered between 20,000 and 24,000 constantly**

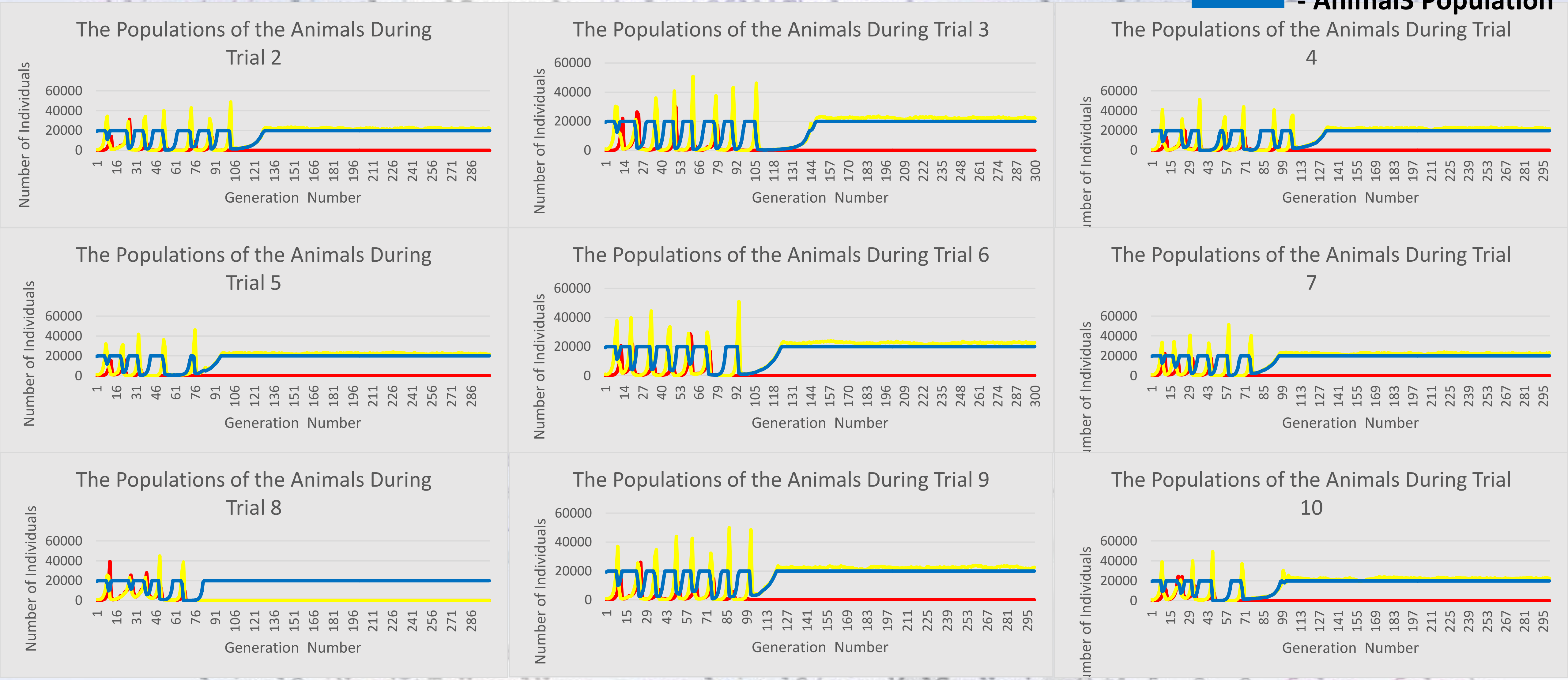
```
public class Executor {
```

The Rest of the Trials’ Populations

🟥 - Animal1 Population

🟨 - Animal2 Population

🟦 - Animal3 Population



The Populations of the Animals During Trial 2

The Populations of the Animals During Trial 3

The Populations of the Animals During Trial 4

The Populations of the Animals During Trial 5

The Populations of the Animals During Trial 6

The Populations of the Animals During Trial 7

The Populations of the Animals During Trial 8

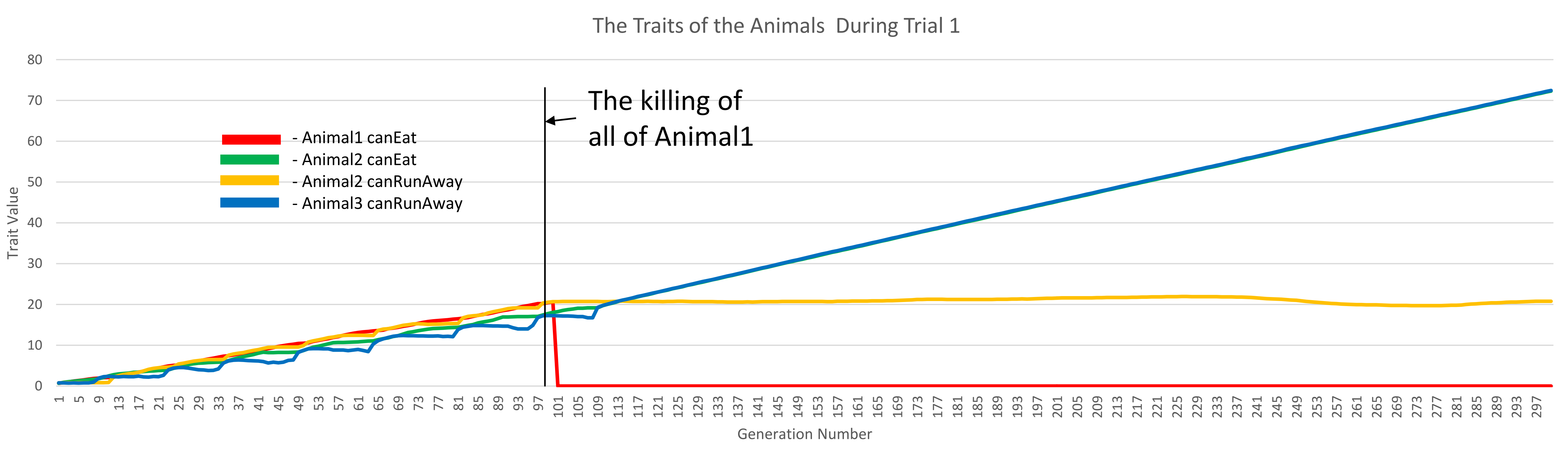
The Populations of the Animals During Trial 9

The Populations of the Animals During Trial 10

```
Animal3 iNeedItToHaveAName = new Animal3(giveMeACanNumber()*1.5, 0, 0, false, false);
theListOfAllTheAnimals.add(iNeedItToHaveAName);
```

- Very consistently the same thing
- Occasional Fails like trial 8 (the animals die off naturally)

Results: Trial 1 Traits


The Traits of the Animals During Trial 1

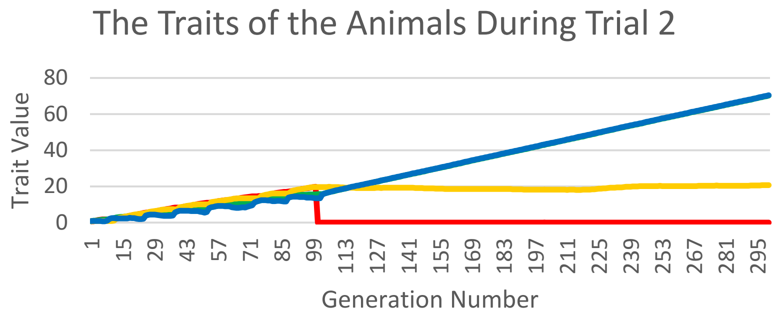
The killing of all of Animal1

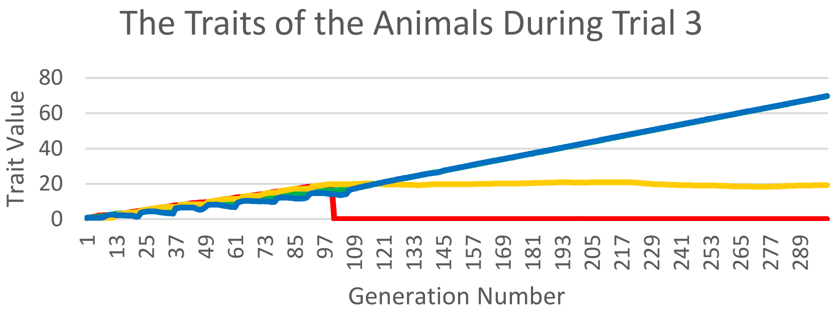
- **Everything is as normal before the killing of all of Animal1**
- **Animal2's canRunAway slowly declines after it**
- **Animal3's canRunAway stops going down then spiking after it**
- **Animal2's canEat increases sharply after it**

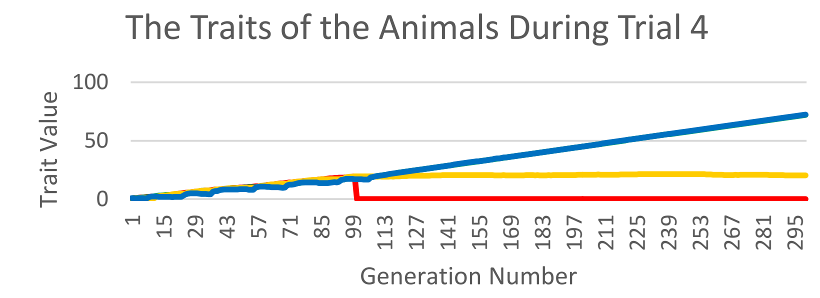
```
public class Excecuter {
                                          rrayList<AnimalGeneral>();
    Animal3 iNeedItToHaveAName = new Animal3(giveMeACanNumber()*1.5, 0, 0, false, false);
    theListOfAllTheAnimals.add(iNeedItToHaveAName);
```

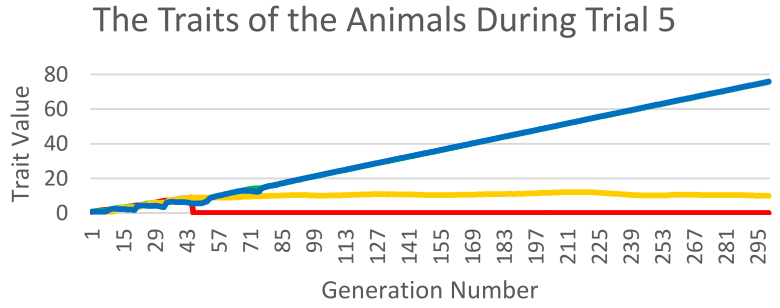
The Rest of the Trials' Traits

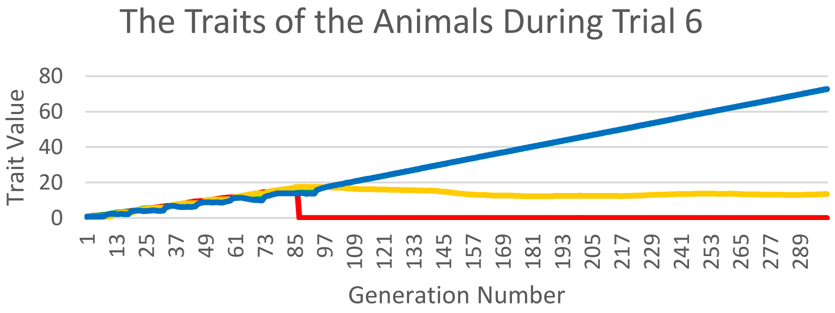
🟥 - Animal1 canEat
🟩 - Animal2 canEat
🟨 - Animal2 canRunAway
🟦 - Animal3 canRunAway

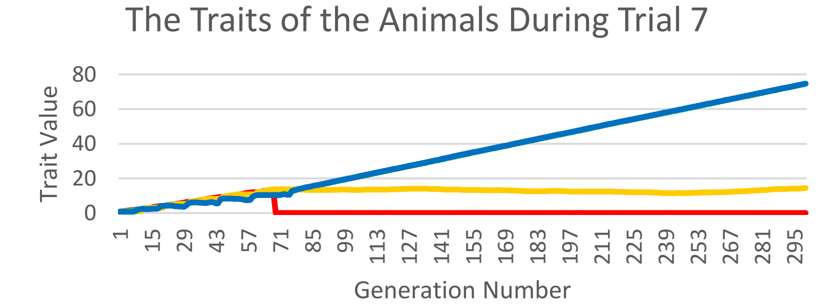

The Traits of the Animals During Trial 2

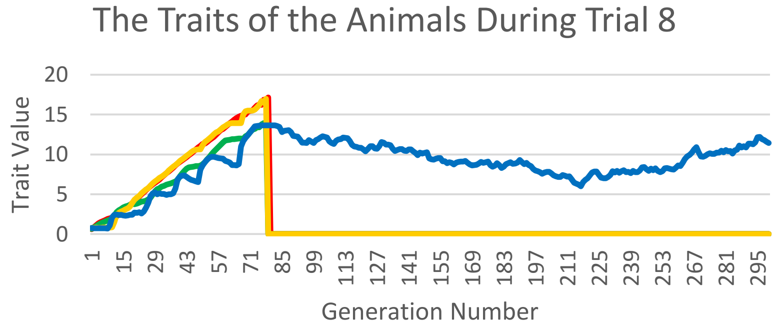

The Traits of the Animals During Trial 3

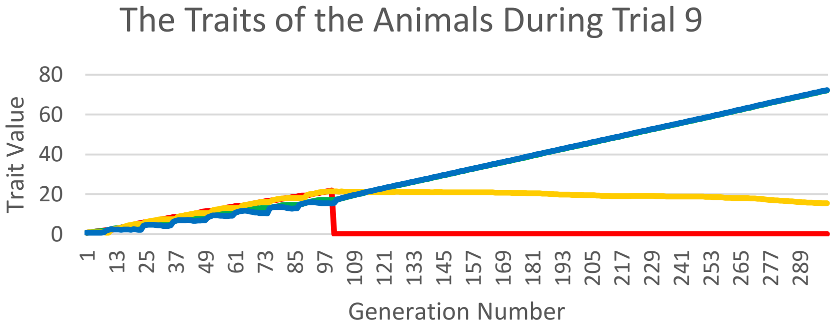

The Traits of the Animals During Trial 4

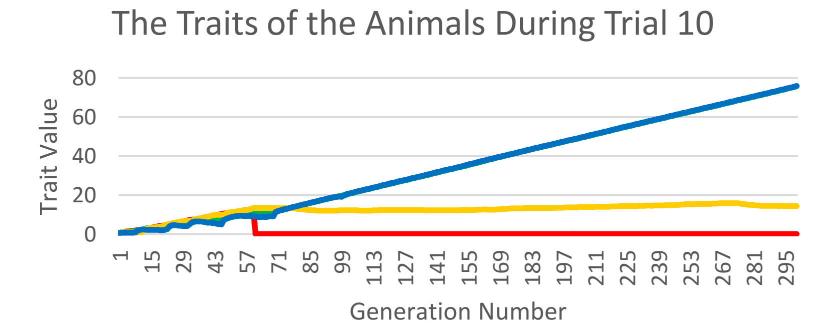

The Traits of the Animals During Trial 5


The Traits of the Animals During Trial 6


The Traits of the Animals During Trial 7


The Traits of the Animals During Trial 8


The Traits of the Animals During Trial 9

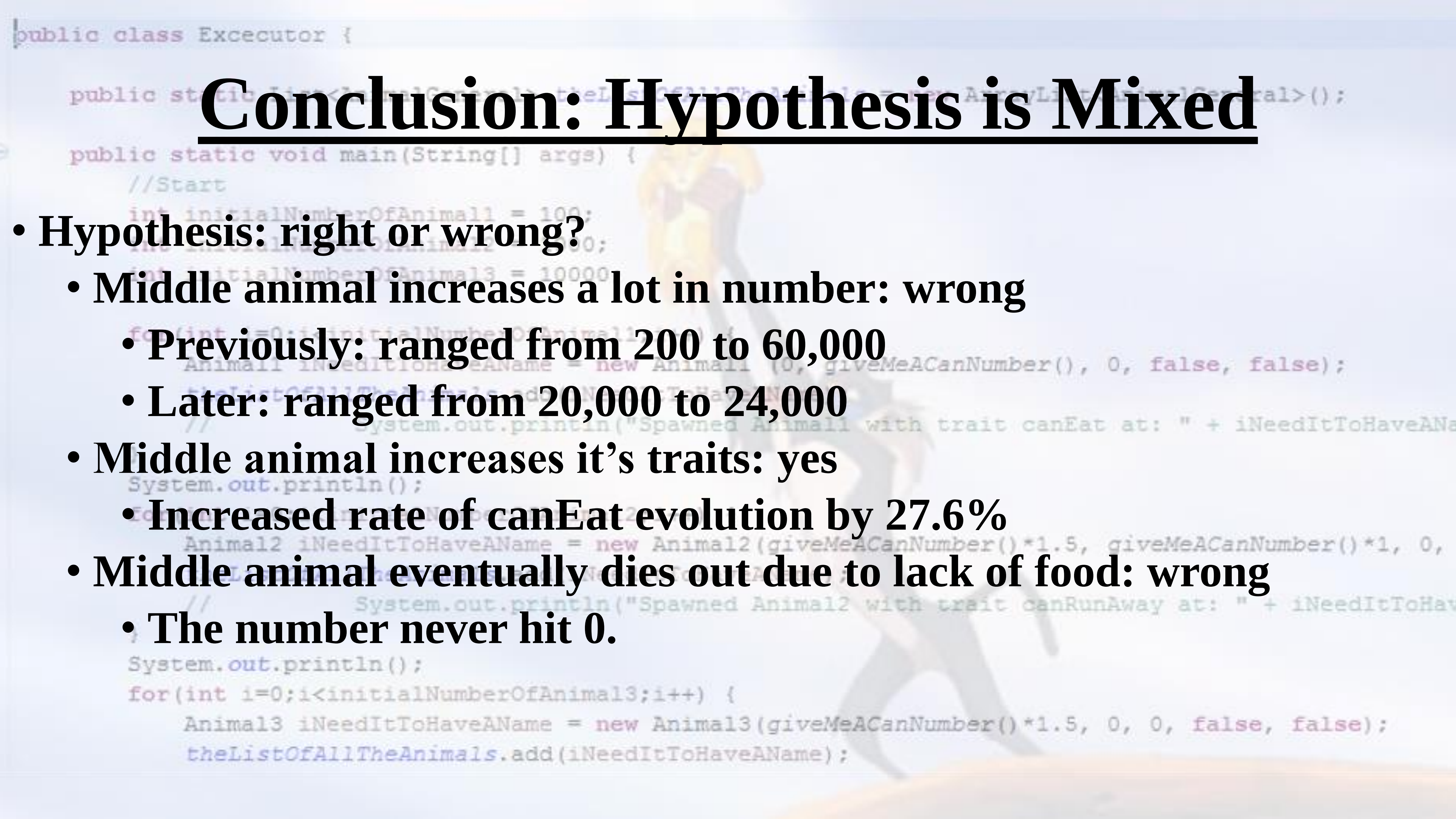

The Traits of the Animals During Trial 10

- **Very consistently the same thing**
- **Occasional Fails like trial 8 (the animals die off naturally)**

Statistical Analysis of the Traits

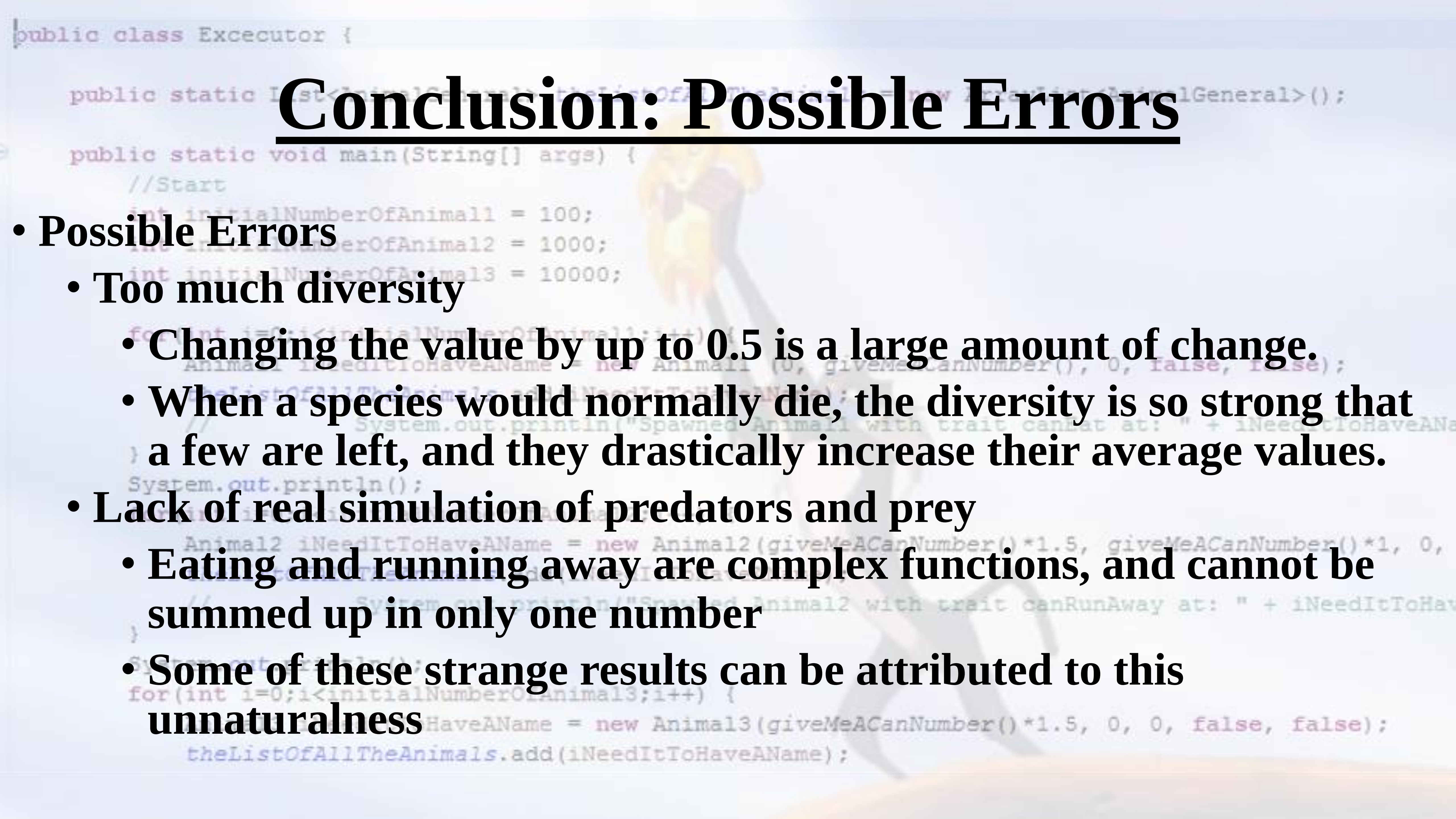
Did Animal2’s rate of canEat increase by much?

- Control Trial
  - If graphed, the slope of the increase in Animal2 canEat generations 0-99 was 0.1826.
  - In generations 100-300, the slope was 0.1852
  - This is a 1.42% increase, statistically insignificant
- Real Trials
  - If graphed, the average slope of the increase in Animal2 canEat generations 0-99 was 0.1550.
  - In generations 100-300, the slope was 0.2203
  - This is a 27.6% increase, statistically significant (more than 5%)



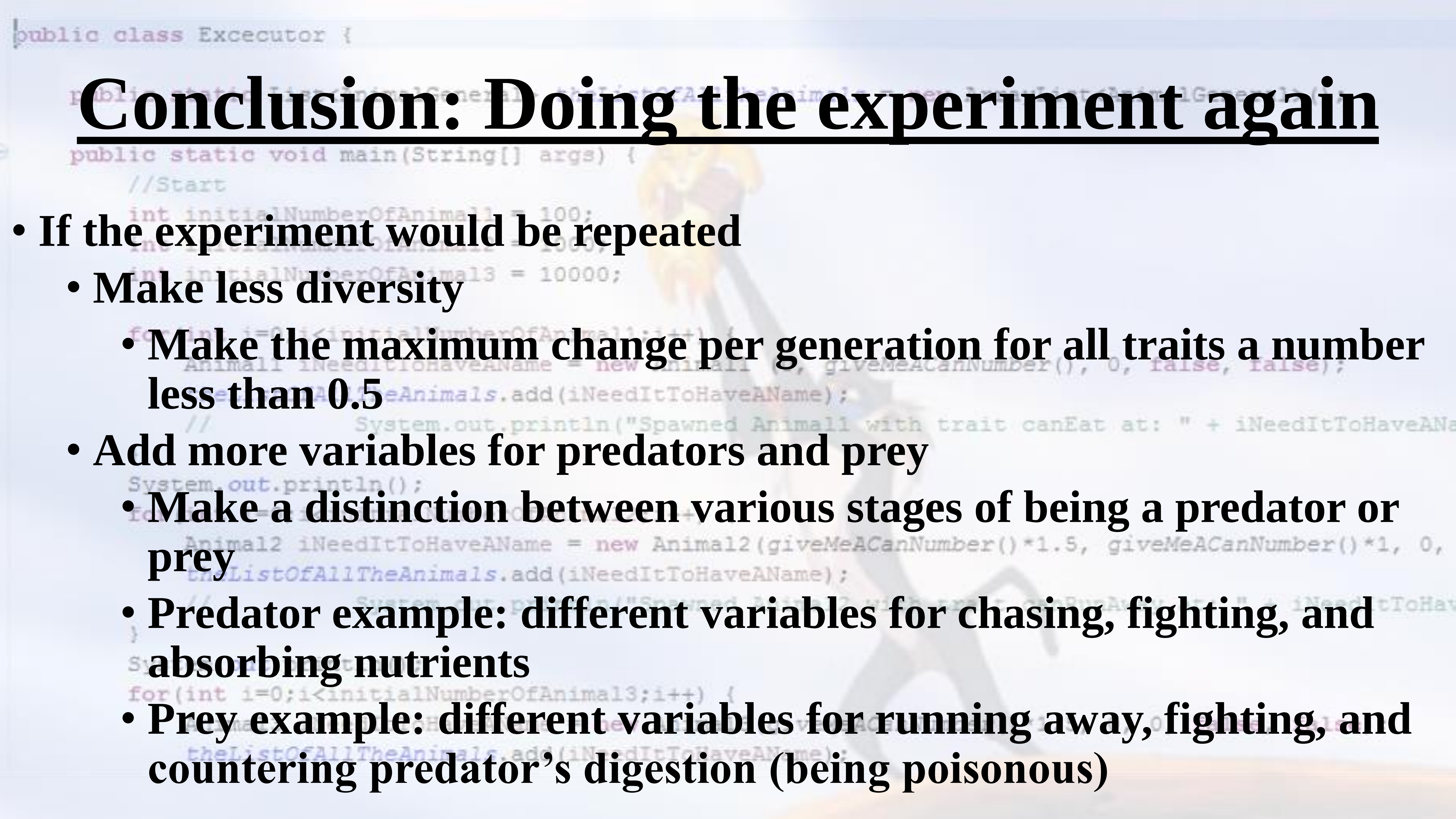
Conclusion: Hypothesis is Mixed

- **Hypothesis: right or wrong?**
  - **Middle animal increases a lot in number: wrong**
    - **Previously: ranged from 200 to 60,000**
    - **Later: ranged from 20,000 to 24,000**
  - **Middle animal increases it’s traits: yes**
    - **Increased rate of canEat evolution by 27.6%**
  - **Middle animal eventually dies out due to lack of food: wrong**
    - **The number never hit 0.**



Conclusion: Possible Errors

- **Possible Errors**
  - **Too much diversity**
    - **Changing the value by up to 0.5 is a large amount of change.**
    - **When a species would normally die, the diversity is so strong that a few are left, and they drastically increase their average values.**
  - **Lack of real simulation of predators and prey**
    - **Eating and running away are complex functions, and cannot be summed up in only one number**
    - **Some of these strange results can be attributed to this unnaturalness**

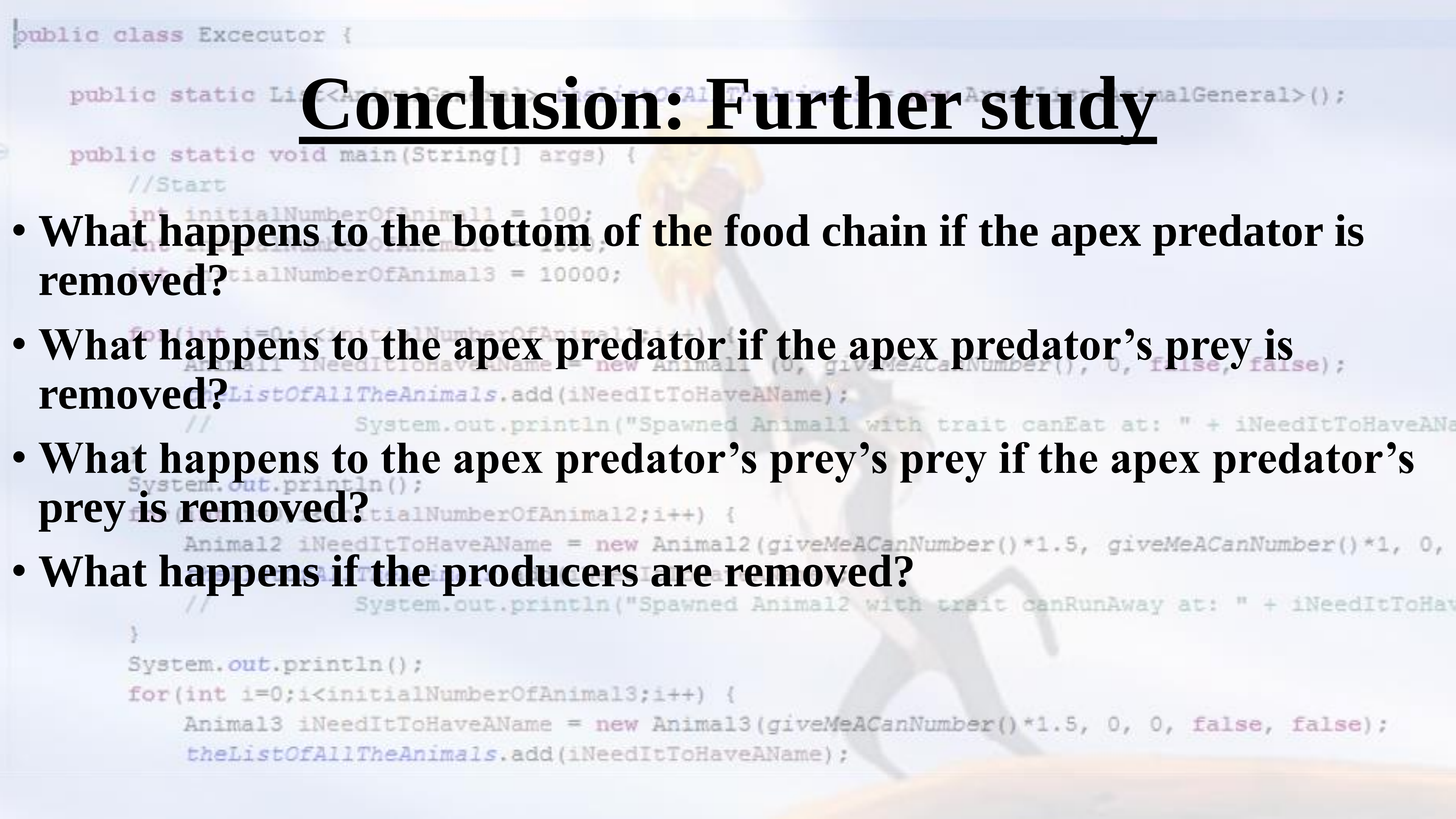


Conclusion: Doing the experiment again

- If the experiment would be repeated
  - Make less diversity
    - Make the maximum change per generation for all traits a number less than 0.5
  - Add more variables for predators and prey
    - Make a distinction between various stages of being a predator or prey
    - Predator example: different variables for chasing, fighting, and absorbing nutrients
    - Prey example: different variables for running away, fighting, and countering predator’s digestion (being poisonous)

Conclusion: Further study

- What happens to the bottom of the food chain if the apex predator is removed?
- What happens to the apex predator if the apex predator’s prey is removed?
- What happens to the apex predator’s prey’s prey if the apex predator’s prey is removed?
- What happens if the producers are removed?



```
public class Excecutor {

    public static List<AnimalGeneral> theListOfAllTheAnimals = new ArrayList<AnimalGeneral>();

    public static void main(String[] args) {
        //Start
        int initialNumberOfAnimal1 = 100;
        int initialNumberOfAnimal2 = 1000;
        int initialNumberOfAnimal3 = 10000;

        for(int i=0;i<initialNumberOfAnimal1;i++) {
            Animal1 iNeedItToHaveAName = new Animal1 (0, giveMeACanNumber(), 0, false, false);
            theListOfAllTheAnimals.add(iNeedItToHaveAName);
//                  System.out.println("Spawned Animal1 with trait canEat at: " + iNeedItToHaveANa
        }
        System.out.println();
        for(int i=0;i<initialNumberOfAnimal2;i++) {
            Animal2 iNeedItToHaveAName = new Animal2(giveMeACanNumber()*1.5, giveMeACanNumber()*1, 0,
            theListOfAllTheAnimals.add(iNeedItToHaveAName);
//                  System.out.println("Spawned Animal2 with trait canRunAway at: " + iNeedItToHav
        }
        System.out.println();
        for(int i=0;i<initialNumberOfAnimal3;i++) {
            Animal3 iNeedItToHaveAName = new Animal3(giveMeACanNumber()*1.5, 0, 0, false, false);
            theListOfAllTheAnimals.add(iNeedItToHaveAName);
```

Conclusion: Real World Application

- **Real world application**
  - **Do not let a species go extinct.**
  - **Many changes can occur in an ecosystem if it happens.**

```
public class Excecutor {

    public static List<AnimalGeneral> theListOfAllTheAnimals = new ArrayList<AnimalGeneral>();

    public static void main(String[] args) {
        //Start
        int initialNumberOfAnimal1 = 100;
        int initialNumberOfAnimal2 = 1000;
        int initialNumberOfAnimal3 = 10000;

        for(int i=0;i<initialNumberOfAnimal1;i++) {
            Animal1 iNeedItToHaveAName = new Animal1 (0, giveMeACanNumber(), 0, false, false);
            theListOfAllTheAnimals.add(iNeedItToHaveAName);
            //          System.out.println("Spawned Animal1 with trait canEat at: " + iNeedItToHaveANa
        }
        System.out.println();
        for(int i=0;i<initialNumberOfAnimal2;i++) {
            Animal2 iNeedItToHaveAName = new Animal2(giveMeACanNumber()*1.5, giveMeACanNumber()*1, 0,
            theListOfAllTheAnimals.add(iNeedItToHaveAName);
            //          System.out.println("Spawned Animal2 with trait canRunAway at: " + iNeedItToHav
        }
        System.out.println();
        for(int i=0;i<initialNumberOfAnimal3;i++) {
            Animal3 iNeedItToHaveAName = new Animal3(giveMeACanNumber()*1.5, 0, 0, false, false);
            theListOfAllTheAnimals.add(iNeedItToHaveAName);
```

Citations

Dowd, Mary. "What Happens When Something in a Food Chain Goes Extinct?" *Synonym*. Demand Media, n.d. Web. 11 Nov. 2015. <http://classroom.synonym.com/happens-something-food-chain-goes-extinct-18214.html>.

Lee, Kevin. "What Happens When Something in a Food Chain Goes Extinct?" *Seattle Pi*. Hearst Seattle Media, n.d. Web. 11 Nov. 2015. <http://education.seattlepi.com/happens-something-food-chain-goes-extinct-4656.html>.

"Loss of Large Predators Disrupting Multiple Plant, Animal and Human Ecosystems." *Science Daily*. ScienceDaily, 14 July 2011. Web. 11 Nov. 2015. <http://www.sciencedaily.com/releases/2011/07/110714142135.htm>.

"Loss of Large Predators Has Caused Widespread Disruption of Ecosystems." *Science Daily*. ScienceDaily, 14 July 2011. Web. 11 Nov. 2015. <http://www.sciencedaily.com/releases/2011/07/110714142128.htm>.

"Loss of Top Animal Predators Has Massive Ecological Effects." *Science Daily*. ScienceDaily, 14 July 2011. Web. 11 Nov. 2015. <http://www.sciencedaily.com/releases/2011/07/110714142133.htm>.

Palkovacs, Eric P., Ben A. Wasserman, and Michael T. Kinnison. *Eco-Evolutionary Trophic Dynamics: Loss of Top Predators Drives Trophic Evolution and Ecology of Prey*. Ed. Howard Browman. *National Center for Biotechnology Information*. National Center for Biotechnology Information, U.S. National Library of Medicine, 3 Dec. 2010. Web. 11 Nov. 2015. <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3079755/#s4title>.

Sieben, Katrin, Anneke D. Rippen, and Britas Klemens Eriksson. *Cascading Effects from Predator Removal Depend on Resource Availability in a Benthic Food Web*. *Springer Link*. Springer International Publishing AG, 3 Nov. 2010. Web. 11 Nov. 2015. <http://link.springer.com/article/10.1007/s00227-010-1567-5/fulltext.html#copyrightInformation>.

Wills, Christopher. *Green Equilibrium: The Vital Balance of Humans and Nature*. Oxford: OUP Oxford, 2013. *Google Books*. 28 Mar. 2013. Web. 11 Nov. 2015. <https://books.google.com/books?id=_JWx72qu5OEC&dq=how+many+secondary+consumers+are+there+%22ten%22&source=gbs_navlinks_s>.

THE

END