```java
package com.example.demo;

import java.util.ArrayList;

public class Deer {
    private int x;
    private int y;
    private int age;
    private int months;
    private int gender; //0: female 1: male
    private int hunger;
    private int thirst;
    private long startTime;
    private int sleep;
    private Deer mother;
    private boolean hasBaby;
    private int myFlock;
    private boolean hasFlock;
    private boolean noticedPredator;
    private Tiger predator;

    public Deer(int x, int y, int a, int f){
        this.x = x;
        this.y = y;
        age = a;
        months = 0;
        gender = (int)(Math.random() * 2);
        hunger = 2;
        thirst = 2;
        sleep = 0;
        startTime = System.nanoTime();
        mother = null;
        hasBaby = false;
        myFlock = f;
        hasFlock = true;
    }

    public void changeLoc(int[][] gameGrid,
   ArrayList<ArrayList<Deer>> allFlock, ArrayList<
   Locations> water, ArrayList<Locations> bush){
        int timeout=0;
```

```
40          boolean check = false;
41          while(!check && timeout<1000){
42              timeout++;
43              int tempx = x;
44              int tempy = y;
45              if(Math.random()>.5 && tempx < gameGrid
   .length-1){
46                  tempx++;
47              }else if(tempx > 0){
48                  tempx--;
49              }
50              if(Math.random()>.5 && tempy < gameGrid
   [0].length-1){
51                  tempy++;
52              }else if(tempy > 0) {
53                  tempy--;
54              }
55              if (gameGrid[tempx][tempy]==0 && tempx
    >= 0 && tempx <= gameGrid.length && tempy >= 0 &&
   tempy <= gameGrid[0].length) {
56                  if(hasFlock && thirst < 20 &&
   hunger < 15 && myFlock >= 0){
57                      if(!this.equals(allFlock.get(
   myFlock).get(0))) {
58                          if (allFlock.get(myFlock).
   get(0).getX() >= tempx - 3 && allFlock.get(myFlock
   ).get(0).getX() <= tempx + 3 &&
59                                  allFlock.get(
   myFlock).get(0).getY() >= tempy - 3 && allFlock.get
   (myFlock).get(0).getY() <= tempy + 3) {
60                              check = true;
61                              if (age <= 1) {
62                                  gameGrid[tempx][
   tempy] = 4;
63                              } else {
64                                  gameGrid[tempx][
   tempy] = gender + 5;
65                              }
66                              gameGrid[x][y] = 0;
67                              x = tempx;
68                              y = tempy;
```

```
 69                          }
 70                      } else {
 71                          check = true;
 72                          if (age <= 1) {
 73                              gameGrid[tempx][tempy
    ] = 4;
 74                          } else {
 75                              gameGrid[tempx][tempy
    ] = gender + 5;
 76                          }
 77                          gameGrid[x][y] = 0;
 78                          x = tempx;
 79                          y = tempy;
 80                      }
 81 //                       mother.hasBaby = true;
 82                  } else {
 83                      if (hunger >= 15) {
 84                          if (this.placeDistance(
    bush, tempx, tempy) < this.placeDistance(bush, x,
    y)) {
 85                              check = true;
 86                              gameGrid[tempx][tempy
    ] = gender + 5;
 87                              gameGrid[x][y] = 0;
 88                              x = tempx;
 89                              y = tempy;
 90                          }
 91                      }else if (thirst >= 20) {
 92                          if (this.placeDistance(
    water, tempx, tempy) < this.placeDistance(water, x
    , y)) {
 93                              check = true;
 94                              gameGrid[tempx][tempy
    ] = gender + 5;
 95                              gameGrid[x][y] = 0;
 96                              x = tempx;
 97                              y = tempy;
 98                          }
 99                      } else{
100                          check = true;
101                          gameGrid[tempx][tempy] =
```

```
101 gender + 5;
102                         gameGrid[x][y] = 0;
103                         x = tempx;
104                         y = tempy;
105                     }
106                 }
107             }
108         }
109 //          System.out.println("x: " + x);
110     }
111 
112     public boolean checkingOpposite(ArrayList<Deer
    > tempGrid){
113         //check how many squares away.
114         for (int i = 0; i < tempGrid.size(); i
    ++) {
115             if(Math.abs(tempGrid.get(i).getX() - x
    ) == 1 && Math.abs(tempGrid.get(i).getY() - y) ==
    1 && gender!=tempGrid.get(i).getGender() &&
    tempGrid.get(i).getAge() >= 1 && age >= 1 && !
    hasBaby){
116                 if(tempGrid.get(i).getMother() !=
    null && !tempGrid.get(i).getMother().equals(this
    )){
117                     return true;
118                 } else if (tempGrid.get(i).
    getMother() == null){
119                     return true;
120                 }
121                 System.out.println("smth aint
    right");
122                 return false;
123             }
124         }
125 //        for (int r = x-1; r <= x+1 && r <
    tempGrid.size() && r>0; r++) {
126 //            for (int c = y-1; c <= y+1 && c <
    tempGrid[r].length && c>0; c++) {
127 //                if(tempGrid[r][c] != 0 && (r!=x
     && c!=y)){
128 //                    return true;
```

```
129 //                 }
130 //             }
131 //         }
132         return false;
133     }
134 
135     public void checkFlock(ArrayList<ArrayList<
    Deer>> allFlock, ArrayList<Deer> tempGrid){
136         if(myFlock >= 0 && myFlock < allFlock.size
    ()){
137             //checks if deer is atleast 5 away
    from flock gorup leader
138             if(allFlock.get(myFlock).get(0).getX
    () >=x-5 && allFlock.get(myFlock).get(0).getX()<=x
    +5 &&
139                     allFlock.get(myFlock).get(0).
    getY() >=y-5 && allFlock.get(myFlock).get(0).getY
    ()<=y+5){
140                 hasFlock = true;
141             } else {
142                 System.out.println("lost the fam
    bro");
143                 hasFlock = false;
144                 //remove deer from flock array
145                 allFlock.get(myFlock).remove(this
    );
146 
147                 if(allFlock.get(myFlock).size
    () == 0){
148                     allFlock.remove(myFlock);
149                     System.out.println("removed
    flock: " + myFlock);
150                     for (int j = 0; j < tempGrid.
    size(); j++) {
151                         if(tempGrid.get(j).
    isHasFlock() && tempGrid.get(j).getMyFlock()>
    myFlock){
152                             tempGrid.get(j).
    setMyFlock(tempGrid.get(j).getMyFlock()-1);
153                         }
154                     }
```

```
155                 }
156 
157                 myFlock = -1;
158             }
159         } else {
160             hasFlock = false;
161             for (int i = 0; i < allFlock.size(); i
    ++) {
162                 if(allFlock.get(i).size() == 0){
163                     allFlock.remove(i);
164                     System.out.println("removed
    flock: " + i);
165                     for (int j = 0; j < tempGrid.
    size(); j++) {
166                         if(tempGrid.get(j).
    isHasFlock() && tempGrid.get(j).getMyFlock()>i){
167                             tempGrid.get(j).
    setMyFlock(tempGrid.get(j).getMyFlock()-1);
168                         }
169                     }
170                 }
171             }
172             //if lone deer encounters new flock it
     will join as long as there is enough space
173             for (int i = 0; i < allFlock.size(); i
    ++) {
174                 if(allFlock.get(i).get(0).getX
    () >=x-5 && allFlock.get(i).get(0).getX()<=x+5 &&
175                         allFlock.get(i).get(0).
    getY() >=y-5 && allFlock.get(i).get(0).getY()<=y+5
     && allFlock.get(i).size() <=7) {
176                     System.out.println("found
    anotha flock besties");
177                     hasFlock = true;
178                     myFlock = i;
179                 }
180             }
181 
182             //if there are two lone deer they
    create their own lone flock
183             for (int i = 0; i < tempGrid.size(); i
```

```
183 ++) {
184                 if(tempGrid.get(i).getX() >=x-5
     && tempGrid.get(0).getX()<=x+5 &&
185                         tempGrid.get(0).getY() >=y
    -5 && tempGrid.get(0).getY()<=y+5 && tempGrid.get(
    i).getMyFlock() < 0 && myFlock < 0){
186                     allFlock.add(new ArrayList
    <>());
187                     allFlock.get(allFlock.size()-1
    ).add(tempGrid.get(i));
188                     tempGrid.get(i).setMyFlock(
    allFlock.size()-1);
189                     tempGrid.get(i).setHasFlock(
    true);
190                     allFlock.get(allFlock.size()-1
    ).add(this);
191                     hasFlock = true;
192                     myFlock = allFlock.size()-1;
193                     System.out.println("CREATED A
    FLOCK");
194                 }
195             }
196         }
197     }
198
199     public void checkPredator(ArrayList<Tiger>
    tempGrid){
200         for (Tiger e: tempGrid) {
201             if(e.getX()>=x-3 && e.getX()<=x+3 && e
    .getY() >=y-3 && e.getY()<=y+3 && e.getAge() >= 3
    ){
202                 if(e.isHunting()){
203                     if(e.getPrey().equals(this
    ) && Math.random() <= .1){
204                         noticedPredator = true;
205                         predator = e;
206                     } else if(!e.getPrey().equals(
    this) && Math.random() <= .4){
207                         noticedPredator = true;
208                         predator = e;
209                     }
```

```
210                 } else if (Math.random() <= .6){
211                     noticedPredator = true;
212                     predator = e;
213                 }
214             }
215         }
216     }
217 
218     public void runningAway(int[][] gameGrid, ArrayList<Tiger> allTiger){
219         ArrayList<Locations> tiger = new ArrayList<>();
220         ArrayList<Locations> tempLocs = new ArrayList<>();
221         double distance;
222         Locations which = new Locations(x, y);
223 
224         //adding every location around the tiger to an array
225         if(allTiger.contains(predator) && predator.getX()>=x-6 && predator.getX()<=x+6 && predator.getY() >=y-6 && predator.getY()<=y+6){
226             tiger.add(new Locations(predator.getX(), predator.getY()));
227             distance = this.placeDistance(tiger, x, y);
228             for (int r = x-1; r <= x+1 && r < gameGrid.length && r > 0; r++) {
229                 for (int c = y-1; c <= y+1 && c < gameGrid[0].length && c > 0; c++) {
230                     if(checkEmptyAround(r, c, gameGrid)){
231                         tempLocs.add(new Locations(r, c));
232                     }
233                 }
234             }
235 
236             //looking through every location around the tiger and determining which is the closest to the target
```

```
237             //if there is no closest location the
    tiger remains in the same spot.
238             for(Locations e: tempLocs){
239                 if(placeDistance(tiger, e.getX(),
    e.getY()) > distance){
240                     distance = placeDistance(tiger
    , e.getX(), e.getY());
241                     which.setX(e.getX());
242                     which.setY(e.getY());
243                 }
244             }
245         } else {
246             predator = null;
247             noticedPredator = false;
248         }
249         if(age<=1){
250             gameGrid[which.getX()][which.getY()]=
    4;
251         } else {
252             gameGrid[which.getX()][which.getY()]=
    gender + 5;
253         }
254         gameGrid[x][y]=0;
255         x= which.getX();
256         y= which.getY();
257     }
258
259     public boolean willDie(){
260         if(hunger >= 15 && thirst >= 20){
261             if(age < 2 && Math.random() >= age * .
    3){
262                 System.out.println("shawty so
    young she couldnt handle the pain bro");
263                 return true;
264             } else if (Math.random() < age * .1){
265                 System.out.println("this was kinda
     random bro");
266                 return true;
267             }
268         } else if (thirst >= 40){
269             System.out.println("shawty dehydrated
```

```
269 bro");
270             return true;
271         } else if (hunger >= 30){
272             System.out.println("shawty starved bro
    ");
273             return true;
274         } else if (age >= 15 && Math.random() > .5
    ){
275             System.out.println("shawty old asf");
276             return true;
277         }
278         return false;
279     }
280 
281     public double placeDistance(ArrayList<
    Locations> place, int x, int y){
282         double distance = 100;
283         for(Locations e: place){
284             double temp = Math.hypot(x-e.getX(), y
    -e.getY());
285             if(temp < distance){
286                 distance = temp;
287             }
288         }
289         return distance;
290     }
291 
292     public boolean checkEmptyAround(int i, int j,
    int[][] gameGrid){
293         return gameGrid[i][j] == 0;
294     }
295 
296     public boolean inWater(int[][] envGrid){
297         return envGrid[x][y] == 1;
298     }
299 
300     public int getX() {
301         return x;
302     }
303 
304     public int getY() {
```

```
305         return y;
306     }
307
308     public int getAge() {
309         return age;
310     }
311
312     public int getGender() {
313         return gender;
314     }
315
316     public int getHunger() {
317         return hunger;
318     }
319
320     public int getThirst() {
321         return thirst;
322     }
323
324     public int getSleep() {
325         return sleep;
326     }
327
328     public Deer getMother() {
329         return mother;
330     }
331
332     public long getStartTime() {
333         return startTime;
334     }
335
336     public int getMonths() {
337         return months;
338     }
339
340     public boolean isHasBaby() {
341         return hasBaby;
342     }
343
344     public int getMyFlock() {
345         return myFlock;
```

```
346     }
347
348     public boolean isHasFlock() {
349         return hasFlock;
350     }
351
352     public boolean isNoticedPredator() {
353         return noticedPredator;
354     }
355
356     public void setAge(int age) {
357         this.age = age;
358     }
359
360     public void setGender(int gender) {
361         this.gender = gender;
362     }
363
364     public void setHunger(int hunger) {
365         this.hunger = hunger;
366     }
367
368     public void setThirst(int thirst) {
369         this.thirst = thirst;
370     }
371
372     public void resetStartTime() {
373         this.startTime = System.nanoTime();
374     }
375
376     public void setSleep(int sleep) {
377         this.sleep = sleep;
378     }
379
380     public void setMother(Deer mother) {
381         this.mother = mother;
382     }
383
384     public void setMyFlock(int myFlock) {
385         this.myFlock = myFlock;
386     }
```

```
387
388     public void setHasFlock(boolean hasFlock) {
389         this.hasFlock = hasFlock;
390     }
391
392     public void setHasBaby(boolean hasBaby) {
393         this.hasBaby = hasBaby;
394     }
395
396     public void setNoticedPredator(boolean
    noticedPredator) {
397         this.noticedPredator = noticedPredator;
398     }
399
400     public void increaseMonths(){
401         months += 2;
402         if(months >= 12){
403             age ++;
404             months = 0;
405         }
406         if (age >= 1 && mother != null) {
407             mother.hasBaby = false;
408         }
409     }
410 }
411
```

```java
package com.example.demo;

import java.util.ArrayList;

public class Tiger {
    private int x;
    private int y;
    private int age;
    private int months;
    private int gender; //0: female 1: male
    private int hunger;
    private int thirst;
    private long startTime;
    private int sleep;
    private Tiger mother;
    private boolean hasBaby;
    private Deer prey;
    private boolean hunting;
    private Tiger targetMate;

    public Tiger(int x, int y, int a){
        this.x = x;
        this.y = y;
        age = a;
        months = 0;
        gender = (int)(Math.random() * 2);
        hunger = 2;
        thirst = 2;
        sleep = 0;
        startTime = System.nanoTime();
        mother = null;
        hasBaby = false;
        prey = null;
        hunting = false;
    }

    public void changeLoc(int[][] gameGrid,
ArrayList<Locations> water){
        int timeout=0;
        boolean check = false;
        while(!check && timeout<1000){
```

```
41              timeout++;
42              int tempx = x;
43              int tempy = y;
44              if(Math.random()>.5 && tempx < gameGrid
   .length-1){
45                  tempx++;
46              }else if(tempx > 0){
47                  tempx--;
48              }
49              if(Math.random()>.5 && tempy < gameGrid
   [0].length-1){
50                  tempy++;
51              }else if(tempy > 0) {
52                  tempy--;
53              }
54              if (gameGrid[tempx][tempy]==0 && tempx
    >= 0 && tempx <= gameGrid.length && tempy >= 0 &&
   tempy <= gameGrid[0].length) {
55                  if(age<3){
56                      if(mother.getX() >=tempx-2 &&
   mother.getX()<=tempx+2 &&
57                                  mother.getY() >=tempy-2
    && mother.getY()<=tempy+2){
58                          check=true;
59                          gameGrid[tempx][tempy]= 1;
60                          gameGrid[x][y]=0;
61                          x=tempx;
62                          y=tempy;
63                      }
64                      mother.hasBaby = true;
65                  } else {
66                      //only moving to the spot if it
    is closer to the water then the current spot
67                      if(thirst>=20){
68                          if(this.placeDistance(water
   , tempx, tempy) < this.placeDistance(water, x, y)){
69                              System.out.println("
   tiger thirsty bro");
70                              check=true;
71                              moved(gameGrid, tempx,
   tempy);
```

```
72                          }
73                      } else {
74                          check=true;
75                          moved(gameGrid, tempx,
   tempy);
76                      }
77                  }
78              }
79          }
80 //          System.out.println("x: " + x);
81      }
82
83      public void moved(int[][] gameGrid, int tempx
   , int tempy){
84          gameGrid[tempx][tempy]= gender + 2;
85          gameGrid[x][y]=0;
86          x=tempx;
87          y=tempy;
88      }
89
90      public boolean checkingOpposite(ArrayList<
   Tiger> tempGrid){
91          //check how many squares away.
92          for (int i = 0; i < tempGrid.size(); i
   ++) {
93              if(Math.abs(tempGrid.get(i).getX() - x
   ) == 1 && Math.abs(tempGrid.get(i).getY() - y) ==
   1 && gender!=tempGrid.get(i).getGender() &&
   tempGrid.get(i).getAge() >= 3 && age >= 3 && !
   hasBaby){
94                  if(tempGrid.get(i).getMother() !=
   null && !tempGrid.get(i).getMother().equals(this
   )){
95                      return true;
96                  } else if (tempGrid.get(i).
   getMother() == null){
97                      return true;
98                  }
99                  System.out.println("smth aint
   right");
100                 return false;
```

```
101             }
102         }
103 //         for (int r = x-1; r <= x+1 && r <
    tempGrid.size() && r>0; r++) {
104 //             for (int c = y-1; c <= y+1 && c <
    tempGrid[r].length && c>0; c++) {
105 //                 if(tempGrid[r][c] != 0 && (r!=x
     && c!=y)){
106 //                     return true;
107 //                 }
108 //             }
109 //         }
110         return false;
111     }
112 
113     public void findPrey(ArrayList<Deer> deer, int
    [] wind) {
114         for (Deer d : deer) {
115             //if deer is 15 blocks away from tiger
     than tiger will chase deer
116             if (d.getX() >= x - 15 + wind[0] && d.
    getX() <= x + 15 + wind[0] &&
117                     d.getY() >= y - 15 + wind[1
    ] && d.getY() <= y + 15 + wind[1] && prey == null
    ) {
118                 prey = d;
119                 hunting = true;
120                 //if there is another deer that is
     less than 2 blcoks away and the deer that is
    currently
121                 //being chased is further away
    than the deer will switch targets.
122             } else if(hunting && d.getX() >= x - 2
     + wind[0] && d.getX() <= x + 2 + wind[0] &&
123                     d.getY() >= y - 2 + wind[0
    ] && d.getY() <= y + 2 + wind[0]){
124                 if(prey.getX() <= x - 2 + wind[0
    ] && prey.getX() >= x + 2 + wind[0] &&
125                         prey.getY() <= y - 2 +
    wind[0] && prey.getY() >= y + 2 + wind[0]){
126                     prey = d;
```

```
127                     System.out.println("tiger
    changed prey");
128                 }
129             }
130         }
131     }
132
133     public void chasingPrey(int[][] gameGrid,
    ArrayList<Deer> allDeer){
134         ArrayList<Locations> deer = new ArrayList
    <>();
135         ArrayList<Locations> tempLocs = new
    ArrayList<>();
136         double distance;
137         Locations which = new Locations(x, y);
138
139         //adding every location around the tiger
    to an array
140         if(allDeer.contains(prey)){
141             deer.add(new Locations(prey.getX(),
    prey.getY()));
142             distance = this.placeDistance(deer, x
    , y);
143             for (int r = x-1; r <= x+1 && r <
    gameGrid.length && r > 0; r++) {
144                 for (int c = y-1; c <= y+1 && c <
    gameGrid[0].length && c > 0; c++) {
145                     if(checkEmptyAround(r, c,
    gameGrid)){
146                         tempLocs.add(new Locations
    (r, c));
147                     }
148                 }
149             }
150
151             //looking through every location
    around the tiger and determining which is the
    closest to the target
152             //if there is no closest location the
    tiger remains in the same spot.
153             for(Locations e: tempLocs){
```

```
154                 if(placeDistance(deer, e.getX(), e
    .getY()) < distance){
155                     distance = placeDistance(deer
    , e.getX(), e.getY());
156                     which.setX(e.getX());
157                     which.setY(e.getY());
158                 }
159             }
160         } else {
161             prey = null;
162             hunting = false;
163         }
164         moved(gameGrid, which.getX(), which.getY
    ());
165     }
166 
167 
168     public void foundPrey(ArrayList<Deer> deer,
    int[][] tempGrid, ArrayList<ArrayList<Deer>>
    allFlock, ArrayList<Tiger> tigers, Deer prey){
169         if(prey.getX() >= x - 1 && prey.getX() <=
    x + 1 &&
170                 prey.getY() >= y - 1 && prey.getY
    () <= y + 1){
171             tempGrid[prey.getX()][prey.getY()] = 0
    ;
172             if(prey.isHasFlock()){
173                 allFlock.get(prey.getMyFlock()).
    remove(prey);
174             }
175             deer.remove(prey);
176             hunger = 0;
177             thirst -= 1;
178             hunting = false;
179             this.prey = null;
180             for (Tiger a: tigers) {
181                 if(a.getAge() < 3 && a.getMother
    () == this){
182                     hunger = 0;
183                     thirst -= 1;
184                 }
```

```
185             }
186             System.out.println("she munching bro"
    );
187         }
188     }
189 
190     public void chasingMate(int[][] gameGrid,
    ArrayList<Tiger> allTiger){
191         ArrayList<Locations> mate = new ArrayList
    <>();
192         ArrayList<Locations> tempLocs = new
    ArrayList<>();
193         double distance;
194         Locations which = new Locations(x, y);
195 
196         //adding every location around the tiger
    to an array
197         if(allTiger.contains(targetMate)){
198             mate.add(new Locations(targetMate.getX
    (), targetMate.getY()));
199             distance = this.placeDistance(mate, x
    , y);
200             for (int r = x-1; r <= x+1 && r <
    gameGrid.length && r > 0; r++) {
201                 for (int c = y-1; c <= y+1 && c <
    gameGrid[0].length && c > 0; c++) {
202                     if(checkEmptyAround(r, c,
    gameGrid)){
203                         tempLocs.add(new Locations
    (r, c));
204                     }
205                 }
206             }
207 
208             //looking through every location
    around the tiger and determining which is the
    closest to the target
209             //if there is no closest location the
    tiger remains in the same spot.
210             for(Locations e: tempLocs){
211                 if(placeDistance(mate, e.getX(), e
```

```
211 .getY()) < distance){
212                     System.out.println("THE TIGERS
     ARE MATING AND CHASING EACH OTHER");
213                     distance = placeDistance(mate
    , e.getX(), e.getY());
214                     which.setX(e.getX());
215                     which.setY(e.getY());
216                 }
217             }
218         } else {
219             targetMate = null;
220         }
221         moved(gameGrid, which.getX(), which.getY
    ());
222     }
223 
224     public boolean willDie(){
225         if(hunger >= 20 && thirst >= 25){
226             if(age < 3 && Math.random() >= age * .
    3){
227                 System.out.println("deer so young
    it died");
228                 return true;
229             } else if (Math.random() < age * .1){
230                 System.out.println("this was kinda
     random bro");
231                 return true;
232             }
233         } else if (thirst >= 45){
234             System.out.println("shawty dehydrated
    bro");
235             return true;
236         } else if (hunger >= 35){
237             System.out.println("shawty starved bro
    ");
238             return true;
239         } else if (age >= 15 && Math.random() > .5
    ){
240             System.out.println("deer very old");
241             return true;
242         }
```

```
243         return false;
244     }
245 
246     public boolean checkDistance(Tiger check, int
    dist){
247         return (check.getX() >= x - dist && check.
    getX() <= x + dist &&
248                 check.getY() >= y - dist && check.
    getY() <= y + dist);
249     }
250 
251     public double placeDistance(ArrayList<
    Locations> places, int x, int y){
252         double distance = 100;
253         for(Locations e: places){
254             double temp = Math.hypot(x-e.getX(), y
    -e.getY());
255             if(temp < distance){
256                 distance = temp;
257             }
258         }
259         return distance;
260     }
261 
262     public boolean inWater(int[][] envGrid){
263         return envGrid[x][y] == 1;
264     }
265 
266     public boolean checkEmptyAround(int i, int j,
    int[][] gameGrid){
267         return gameGrid[i][j] == 0;
268     }
269 
270     public int getX() {
271         return x;
272     }
273 
274     public int getY() {
275         return y;
276     }
277 
```

```
278     public int getAge() {
279         return age;
280     }
281 
282     public int getGender() {
283         return gender;
284     }
285 
286     public int getHunger() {
287         return hunger;
288     }
289 
290     public int getThirst() {
291         return thirst;
292     }
293 
294     public int getSleep() {
295         return sleep;
296     }
297 
298     public Tiger getMother() {
299         return mother;
300     }
301 
302     public long getStartTime() {
303         return startTime;
304     }
305 
306     public int getMonths() {
307         return months;
308     }
309 
310     public Deer getPrey() {
311         return prey;
312     }
313 
314     public boolean isHasBaby() {
315         return hasBaby;
316     }
317 
318     public boolean isHunting() {
```

```
319         return hunting;
320     }
321 
322     public void setAge(int age) {
323         this.age = age;
324     }
325 
326     public void setGender(int gender) {
327         this.gender = gender;
328     }
329 
330     public void setHunger(int hunger) {
331         this.hunger = hunger;
332     }
333 
334     public void setThirst(int thirst) {
335         this.thirst = thirst;
336     }
337 
338     public void resetStartTime() {
339         this.startTime = System.nanoTime();
340     }
341 
342     public void setSleep(int sleep) {
343         this.sleep = sleep;
344     }
345 
346     public void setMother(Tiger mother) {
347         this.mother = mother;
348     }
349 
350 //    public void setMonths{int m}{
351 //        this.months = m;
352 //    }
353 
354 
355     public void setHasBaby(boolean hasBaby) {
356         this.hasBaby = hasBaby;
357     }
358 
359     public void increaseMonths(){
```

```
360             months += 2;
361             if(months >= 12){
362                 age ++;
363                 months = 0;
364             }
365             if (age >= 3 && mother != null) {
366                 mother.hasBaby = false;
367             }
368         }
369 
370         public Tiger getTargetMate() {
371             return targetMate;
372         }
373 
374         public void setTargetMate(Tiger targetMate) {
375             this.targetMate = targetMate;
376         }
377 }
378 
```

```java
package com.example.demo;

public class Locations {
    private int x;
    private int y;

    public Locations(int xx, int yy){
        x = xx;
        y = yy;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }

    public void setX(int x) {
        this.x = x;
    }

    public void setY(int y) {
        this.y = y;
    }
}

```

```java
package com.example.demo;

import javafx.fxml.FXML;


//import com.sun.prism.paint.Color;

import java.util.ArrayList;

import javafx.animation.AnimationTimer;
import javafx.event.ActionEvent;
import javafx.scene.chart.LineChart;
import javafx.scene.chart.XYChart;
import javafx.scene.control.*;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.GridPane;

public class HelloController {
    int x = 44;
    int y = 59;
    Button[][] btn = new Button[x][y];
    int[][] gameGrid = new int[x][y];
    int[][] environmentGrid = new int[x][y];
    int[] wind = new int[2];
    long gameStart = System.nanoTime();
    boolean matingSeason = false;
    int salinityLvl = 0;
    int riverWidth = 6;
    ArrayList<Locations> water = new ArrayList<>();
    ArrayList<Locations> bush = new ArrayList<>();
    ArrayList<Tiger> allTigers = new ArrayList<>();
    ArrayList<ArrayList<Deer>> allFlocks = new
ArrayList<>();
    ArrayList<Deer> allDeer = new ArrayList<>();
    //GridPane gPane = new GridPane();
//    Image k = new Image("resources/Koala.jpg");
    @FXML
    private AnchorPane aPane, keyPane;
    @FXML
    private GridPane gPane;
    @FXML
```

```
41      private LineChart lChart;
42      @FXML
43      private Button startButton, tigerButton,
    deerButton;
44      @FXML
45      private CheckBox matingCheck;
46      @FXML
47      private Slider climateBar, humanBar;
48      @FXML
49      private Label climate, human;
50
51      ArrayList<Integer> monthlyDeer = new ArrayList
    <>();
52      ArrayList<Integer> monthlyTiger = new ArrayList
    <>();
53      XYChart.Series deerPop = new XYChart.Series();
54      XYChart.Series tigerPop = new XYChart.Series();
55
56      @FXML
57      private void handleStart(ActionEvent event) {
58          startButton.setVisible(false);
59          tigerButton.setVisible(true);
60          deerButton.setVisible(true);
61          matingCheck.setVisible(true);
62          climateBar.setVisible(true);
63          humanBar.setVisible(true);
64          climate.setVisible(true);
65          human.setVisible(true);
66          keyPane.setVisible(true);
67          lChart.setVisible(true);
68          lChart.getYAxis().setLabel("POPULATION");
69          lChart.getXAxis().setLabel("MONTHS");
70          //after adding the grdipane in scene
71          // builder, modify the fxml manually to
    eliminate
72          // rows and columns
73
74
75  //          gPane.setMinSize(0,0);
76          //gPane.setPadding(new Insets(btn[i][j]));
77          gPane.setHgap(0);
```

```
 78          gPane.setVgap(0);
 79          //gPane.setGridLinesVisible(true);
 80          //gPane.setAlignment(Pos.CENTER);
 81
 82          for (int i = 0; i < btn.length; i++) {
 83              for (int j = 0; j < btn[0].length; j
    ++) {
 84
 85                  //Initializing 2D buttons with
    values i,j
 86                  btn[i][j] = new Button();
 87                  btn[i][j].setStyle("-fx-background
    -color:#d3d3d3");
 88
 89                  btn[i][j].setMinWidth(20);
 90                  btn[i][j].setMinHeight(20);
 91
 92                  btn[i][j].setMaxWidth(20);
 93                  btn[i][j].setMaxHeight(20);
 94
 95
 96 //                  btn[i][j].setPrefSize(25, 5);
 97                  //Paramters:  object, columns,
    rows
 98                  gPane.add(btn[i][j], j, i);
 99                  gameGrid[i][j] = 0;
100
101
102              }
103          }
104
105          gPane.setGridLinesVisible(true);
106
107          gPane.setVisible(true);
108
109          for (int i = 0; i < environmentGrid.length
    ; i++) {
110              for (int j = 0; j < environmentGrid[0
    ].length; j++) {
111                  environmentGrid[i][j]=2;
112              }
```

```
113         }
114 
115         for (int i = 0; i < 10; i++) {
116             int x = (int)(Math.random()*(
    environmentGrid.length - 10)) + 3;
117             int y = (int)(Math.random()*(
    environmentGrid[0].length - 10)) + 3;
118             boolean tooClose = true;
119 
120             //making sure that the grass aren't
    too close to each other
121             while(tooClose){
122                 tooClose = false;
123                 for (int r = x-10; r <= x+10 && r
     < gameGrid.length && r > 0; r++) {
124                     for (int c = y - 10; c <= y +
    10 && c < gameGrid[0].length && c > 0; c++) {
125                         if(environmentGrid[r][c
    ] == 3){
126                             tooClose = true;
127                         }
128                     }
129                 }
130                 if(tooClose){
131                     x = (int)(Math.random()*(
    environmentGrid.length - 10)) + 3;
132                     y = (int)(Math.random()*(
    environmentGrid[0].length - 10)) + 3;
133                 }
134             }
135 
136             //coloring the blcoks in
137             for (int j = 0; j < (int)(Math.random
    ()*4)+3; j++) {
138                 environmentGrid[x-1][y+j] = 3;
139                 environmentGrid[x][y+j] = 3;
140                 environmentGrid[x+1][y+j] = 3;
141             }
142         }
143 
144         setRiver(riverWidth);
```

```
145         start();
146         updateScreen();
147     }
148 
149     public void setRiver(int width){
150         water.clear();
151         bush.clear();
152         //setting certain spots on the grid to
    water
153         int j = (int)((Math.random() * 5) + 23);
154         for(int i = 0; i < environmentGrid.length
    ; i++){
155             environmentGrid[i][j]=1;
156             for (int k = j; k < j+width; k++) {
157                 environmentGrid[i][k]=1;
158             }
159             int chance = (int)(Math.random() * 5);
160             if(chance == 4){
161                 j--;
162             } else if (chance == 3) {
163                 j++;
164             }
165         }
166         int a = (int)((Math.random() * 5) + 16);
167         for (int i = 0; i < environmentGrid[0].
    length; i++) {
168             environmentGrid[a][i]=1;
169             for (int k = a; k < a+width; k++) {
170                 environmentGrid[k][i]=1;
171             }
172             int chance = (int)(Math.random() * 5);
173             if(chance == 4){
174                 a--;
175             } else if (chance == 3) {
176                 a++;
177             }
178         }
179 
180         //adding which locations on the grid have
    water
181         for (int r = 0; r < environmentGrid.length
```

```
181 ; r++) {
182             for (int c = 0; c < environmentGrid[0
    ].length; c++) {
183                 if(environmentGrid[r][c] == 1){
184                     water.add(new Locations(r,c));
185                 }
186             }
187         }
188 
189         //adding the bush locations
190         for (int i = 0; i < environmentGrid.length
    ; i++) {
191             for (int h = 0; h < environmentGrid[0
    ].length; h++) {
192                 if(environmentGrid[i][h] == 3){
193                     bush.add(new Locations(i, h));
194                 }
195             }
196         }
197     }
198 
199     public void updateScreen() {
200         //environment grid first, then overide
    with objects
201         //1=river, 2 = grass
202         for (int i = 0; i < btn.length; i++) {
203             for (int j = 0; j < btn[0].length; j
    ++) {
204                 if (environmentGrid[i][j] == 0) {
205                     btn[i][j].setStyle("-fx-
    background-color:#d3d3d3");
206                 } else if (environmentGrid[i][j
    ] == 1) {
207                     btn[i][j].setStyle("-fx-
    background-color: #ADD8E6");
208                 } else if (environmentGrid[i][j
    ] == 2) {
209                     btn[i][j].setStyle("-fx-
    background-color:#86c892");
210                 } else if (environmentGrid[i][j
    ] == 3) {
```

```
211                     btn[i][j].setStyle("-fx-
    background-color:#5d8861");
212                 }
213             }
214         }
215 
216         // 0 = empty(already set in environment,
217         // 1 = tiger child, 2 = tiger female 3 =
    tiger male
218         // 4 = deer child, 5 = deer female, 6 =
    deer male
219         for(int i=0; i<btn.length; i++) {
220             for (int j = 0; j < btn[0].length; j
    ++) {
221                 if (gameGrid[i][j]==1){
222                     btn[i][j].setStyle("-fx-
    background-color:#fa8128");
223                 } else if (gameGrid[i][j]==2){
224                     btn[i][j].setStyle("-fx-
    background-color:#dd571c");
225                 } else if (gameGrid[i][j]==3){
226                     btn[i][j].setStyle("-fx-
    background-color:#8a3008");
227                 } else if (gameGrid[i][j]==4){
228                     btn[i][j].setStyle("-fx-
    background-color:#d4b37f");
229                 } else if (gameGrid[i][j]==5){
230                     btn[i][j].setStyle("-fx-
    background-color:#bd9168");
231                 } else if (gameGrid[i][j]==6){
232                     btn[i][j].setStyle("-fx-
    background-color:#a3866a");
233                 }
234             }
235         }
236     }
237 
238     @FXML
239     private void addTiger(ActionEvent event) {
240         int x = (int)(Math.random()*(
    environmentGrid.length));
```

```
241         int y = (int)(Math.random()*(
    environmentGrid[0].length));
242         allTigers.add(new Tiger(x,y, 3));
243         //setting the game grid to a certain
    number depending on the age and gender of the
    tiger
244         if(allTigers.get(allTigers.size()-1).
    getAge() < 3){
245             gameGrid[allTigers.get(allTigers.size
    ()-1).getX()][allTigers.get(allTigers.size()-1).
    getY()] = 1;
246         } else {
247
248             if(allTigers.get(allTigers.size()-1).
    getGender() == 0){
249                 gameGrid[allTigers.get(allTigers.
    size()-1).getX()][allTigers.get(allTigers.size()-1
    ).getY()] = 2;
250             } else {
251                 gameGrid[allTigers.get(allTigers.
    size()-1).getX()][allTigers.get(allTigers.size()-1
    ).getY()] = 3;
252             }
253
254         }
255         updateScreen();
256     }
257
258     @FXML
259     private void addFlock(ActionEvent event){
260         allFlocks.add(new ArrayList<>());
261         int x = (int)(Math.random()*(
    environmentGrid.length - 12)) + 6;
262         int y = (int)(Math.random()*(
    environmentGrid[0].length - 12)) + 6;
263
264         //creates the leader of the flock and then
     every other deer in the flock will be surrounded
    around the leader
265         allFlocks.get(allFlocks.size()-1).add(new
    Deer(x, y, 2, allFlocks.size()-1));
```

```
266         allFlocks.get(allFlocks.size()-1).get(0).
    setGender(1);
267 
268         for (int j = 0; j < (int)(Math.random()*3
    ) + 2; j++) {
269             int r = (int)(Math.random()*7) + (x-3
    );
270             int c = (int)(Math.random()*7) + (y-3
    );
271             allFlocks.get(allFlocks.size()-1).add(
    new Deer(r, c, 2, allFlocks.size()-1));
272         }
273 
274         for(int i = 0; i < allFlocks.get(allFlocks
    .size()-1).size(); i++){
275             gameGrid[allFlocks.get(allFlocks.size
    ()-1).get(i).getX()][allFlocks.get(allFlocks.size
    ()-1).get(i).getY()] = allFlocks.get(allFlocks.
    size()-1).get(i).getGender() + 5;
276         }
277 
278         allDeer.addAll(allFlocks.get(allFlocks.
    size() - 1));
279         System.out.println(allFlocks.get(allFlocks
    .size() - 1).size());
280         updateScreen();
281     }
282 
283     @FXML
284     private void handleMating(){
285         if(matingCheck.isSelected()){
286             matingSeason = true;
287         } else {
288             matingSeason = false;
289         }
290     }
291 
292     public void start() {
293         lChart.getData().add(deerPop);
294         lChart.getData().add(tigerPop);
295         new AnimationTimer() {
```

```
296             @Override
297             public void handle(long now) {
298 //                System.out.println(now);
299                 if(now - gameStart > 2000000000.0
    ){
300 //                  lChart.getData().clear();
301                     monthlyDeer.add(allDeer.size
    ());
302                     monthlyTiger.add(allTigers.
    size());
303                     deerPop.setName("Deer
    Population");
304                     tigerPop.setName("Tiger
    Population");
305                     deerPop.getData().clear();
306                     tigerPop.getData().clear();
307                     for (int i = 0; i <
    monthlyTiger.size(); i++) {
308                         deerPop.getData().add(new
    XYChart.Data(i, monthlyDeer.get(i)));
309                         tigerPop.getData().add(new
     XYChart.Data(i, monthlyTiger.get(i)));
310                     }
311                     gameStart = System.nanoTime();
312                     humanInteraction();
313                     climateChange();
314                 }
315                 if (allTigers.size() > 0) {
316                     for (int i = 0; i < allTigers.
    size(); i++) {
317                         if(allTigers.get(i).
    isHunting()){
318                             //checking if tiger is
     next to prey
319                             allTigers.get(i).
    foundPrey(allDeer, gameGrid, allFlocks, allTigers
    , allTigers.get(i).getPrey());
320                             //checking if tiger is
     next to any other deer
321                             for (int j = 0; j <
    allDeer.size(); j++) {
```

```
322                                 allTigers.get(i).
    foundPrey(allDeer, gameGrid, allFlocks, allTigers
    , allDeer.get(j));
323                             }
324                         }
325
326                         if(Math.random() <= 0.2){
327                             windy();
328                         }
329
330                         if (now - allTigers.get(i
    ).getStartTime() > 1000000000.0) {
331
332                             if(allTigers.get(i).
    checkingOpposite(allTigers) && allTigers.get(i).
    getGender()==0 && (!allTigers.get(i).inWater(
    environmentGrid))){
333                                 System.out.println
    ("female tiger found a male tiger");
334                                 for (int j = 0; j
     < (int)((Math.random()*3) + 1); j++) {
335                                     System.out.
    println("female reproduced");
336                                     reproduceTiger
    (i);
337                                 }
338                             }
339
340                             if(allTigers.get(i).
    isHunting()){
341                                 allTigers.get(i).
    chasingPrey(gameGrid, allDeer);
342                             } else if(matingSeason
     && allTigers.get(i).getThirst() <= 15 &&
    allTigers.get(i).getAge() < 3 && !allTigers.get(i
    ).isHasBaby()) {
343                                 //running through
    all tigers array and checking which one is close,
    diff gender and doesn't already have a target mate
344                                 for (int j = 0; j
     < allTigers.size() && allTigers.get(i).
```

```
344 getTargetMate() == null; j++) {
345                                     if(allTigers.
    get(i).checkDistance(allTigers.get(j), 10) && (
    allTigers.get(j).getGender() != allTigers.get(i).
    getGender()) && allTigers.get(j).getTargetMate
    () == null && allTigers.get(j).getAge() < 3 && !
    allTigers.get(j).isHasBaby()){
346                                         allTigers.
    get(i).setTargetMate(allTigers.get(j));
347                                         allTigers.
    get(j).setTargetMate(allTigers.get(i));
348                                     }
349                                 }
350
351                                 if(allTigers.get(i
    ).getTargetMate() != null){
352                                     allTigers.get(
    i).chasingMate(gameGrid, allTigers);
353                                 }
354
355                             } else {
356                                 allTigers.get(i).
    changeLoc(gameGrid, water);
357                             }
358
359                             //if the tiger
    consumes food or water the stats are reset to 0.
360                             if(allTigers.get(i).
    getHunger() >= 15){
361                                 allTigers.get(i).
    findPrey(allDeer, wind);
362                                 System.out.println
    ("tiger found prey: " + allTigers.get(i).isHunting
    ());
363                             }
364
365                             if(allTigers.get(i).
    inWater(environmentGrid)){
366                                 if(salinityLvl >=
    10){
367                                     allTigers.get(
```

```
367 i).setThirst(salinityLvl);
368                                 }else{
369                                     allTigers.get(
    i).setThirst(0);
370                                 }
371                                 System.out.println
    ("tiger found water");
372                             }
373 
374                             allTigers.get(i).
    increaseMonths();
375                             allTigers.get(i).
    setHunger(allTigers.get(i).getHunger()+1);
376                             allTigers.get(i).
    resetStartTime();
377 
378                             //there is a
    randomized probability of the thirst increasing in
     order to contrast the need of water for deers and
     tigers
379                             if(Math.random()>.3
     && allTigers.get(i).getAge() >= 3){
380                                 allTigers.get(i).
    setThirst(allTigers.get(i).getThirst()+1);
381                             }
382 
383                             //if the target mates
    are over 20 blocks away from each other than they
    will not be target mates no more
384                             if(allTigers.get(i).
    getTargetMate() != null){
385                                 if(!allTigers.get(
    i).checkDistance(allTigers.get(i).getTargetMate
    (), 20)){
386                                     allTigers.get(
    i).getTargetMate().setTargetMate(null);
387                                     allTigers.get(
    i).setTargetMate(null);
388                                 }
389                             }
390 
```

```
391                                 //if the baby is too
    far awar from the mom it will die because it has
    no support
392                                 if(allTigers.get(i).
    getAge() < 3 && !allTigers.get(i).checkDistance(
    allTigers.get(i).getMother(), 4)){
393                                     gameGrid[allTigers
    .get(i).getX()][allTigers.get(i).getY()] = 0;
394                                     allTigers.remove(i
    );
395                                     System.out.println
    ("the baby tiger lost the mother");
396                                 } else if(allTigers.
    get(i).willDie()){
397                                     //based on the
    health and age the probabilities of death are
    increased
398                                     gameGrid[allTigers
    .get(i).getX()][allTigers.get(i).getY()] = 0;
399                                     if(allTigers.get(i
    ).getTargetMate() != null){
400                                         allTigers.get(
    i).getTargetMate().setTargetMate(null);
401                                     }
402                                     allTigers.remove(i
    );
403                                     System.out.println
    ("tiger died bro");
404                                 }
405                             }
406                         }
407                 }
408
409                 if(allDeer.size() > 0){
410                     for(int i = 0; i < allDeer.
    size(); i++){
411                         if (now - allDeer.get(i).
    getStartTime() > 1000000000.0) {
412                             if(allDeer.get(i).
    getAge() <= 1){
413                                 allDeer.get(i).
```

```
413 getMother().setHasBaby(true);
414                                 }
415                                 if(allDeer.get(i).
    checkingOpposite(allDeer) && allDeer.get(i).
    getGender()==0 && (!allDeer.get(i).inWater(
    environmentGrid))){
416                                     System.out.println
    ("female deer found a male deer");
417                                     reproduceDeer(i);
418                                 }
419 
420                                 allDeer.get(i).
    checkPredator(allTigers);
421 
422                                 //if noticed predator
    uses special algorithim to run away else just uses
     randomized change location
423                                 if(allDeer.get(i).
    isNoticedPredator()){
424                                     allDeer.get(i).
    runningAway(gameGrid, allTigers);
425                                     System.out.println
    ("deer noticed hungry tiger");
426                                 } else {
427                                     allDeer.get(i).
    changeLoc(gameGrid, allFlocks, water, bush);
428                                 }
429 
430                                 //setting hunger and
    thirst to 0 if in respective places
431                                 if(allDeer.get(i).
    inWater(environmentGrid)){
432                                     if(salinityLvl >=
    7){
433                                         allDeer.get(i
    ).setThirst(salinityLvl);
434                                     } else {
435                                         allDeer.get(i
    ).setThirst(0);
436                                     }
437                                 }
```

```
438                                 if(environmentGrid[
    allDeer.get(i).getX()][allDeer.get(i).getY()]==3){
439                                     allDeer.get(i).
    setHunger(0);
440                                 }
441
442                                 allDeer.get(i).
    checkFlock(allFlocks, allDeer);
443                                 allDeer.get(i).
    increaseMonths();
444                                 allDeer.get(i).
    resetStartTime();
445                                 allDeer.get(i).
    setHunger(allDeer.get(i).getHunger()+1);
446
447                                 if(Math.random()>.5){
448                                     allDeer.get(i).
    setThirst(allDeer.get(i).getThirst()+1);
449                                 }
450
451                                 if(allDeer.get(i).
    getAge() < 1 && !allDeer.get(i).isHasFlock()){
452                                     gameGrid[allDeer.
    get(i).getX()][allDeer.get(i).getY()] = 0;
453                                     allDeer.remove(i);
454                                     System.out.println
    ("baby deer died bro");
455                                 } else if(allDeer.get(
    i).willDie()){
456                                     gameGrid[allDeer.
    get(i).getX()][allDeer.get(i).getY()] = 0;
457                                     if(allDeer.get(i).
    isHasFlock()) {
458                                         allFlocks.get(
    allDeer.get(i).getMyFlock()).remove(allDeer.get(i
    ));
459                                         if (allFlocks.
    get(allDeer.get(i).getMyFlock()).size() == 0) {
460                                             allFlocks.
    remove(allDeer.get(i).getMyFlock());
461                                             System.out
```

```
461 .println("removed flock: " + i);
462                                         for (int j
     = 0; j < allDeer.size(); j++) {
463                                             if (
    allDeer.get(j).isHasFlock() && allDeer.get(j).
    getMyFlock() > allDeer.get(i).getMyFlock()) {
464 
    allDeer.get(j).setMyFlock(allDeer.get(j).
    getMyFlock() - 1);
465                                             }
466                                         }
467                                     }
468                                     allDeer.remove
    (i);
469                                     System.out.
    println("deer died bro");
470                                 }
471                             }
472                         }
473                     }
474                 }
475                 updateScreen();
476             }
477         }.start();
478     }
479 
480     ArrayList<Locations> tempLocs = new ArrayList
    <>();
481     public void reproduceTiger(int which){
482         tempLocs.clear();
483         double prob;
484         if(matingSeason){
485             prob = 1;
486         } else {
487             prob = 0.7;
488         }
489         //this code is running through every
    location around the ant.
490         for (int r = allTigers.get(which).getX()-1
    ; r <= allTigers.get(which).getX()+1 && r <
    gameGrid.length && r > 0; r++) {
```

```
491             for (int c = allTigers.get(which).getY
    ()-1; c <= allTigers.get(which).getY()+1 && c <
    gameGrid[0].length && c > 0; c++) {
492                 if(checkEmptyAround(r, c) && !(
    environmentGrid[r][c] == 1)){
493                     tempLocs.add(new Locations(r,
    c));
494                 }
495             }
496         }
497         if(tempLocs.size()>0 && allTigers.get(
    which).getAge() >= 3 && Math.random() <= prob){
498             int num = (int)(Math.random() *
    tempLocs.size());
499             allTigers.add(new Tiger(tempLocs.get(
    num).getX(), tempLocs.get(num).getY(), 0));
500             gameGrid[tempLocs.get(num).getX()][
    tempLocs.get(num).getY()] = 1;
501             allTigers.get(allTigers.size()-1).
    setMother(allTigers.get(which));
502             System.out.println("new baby tiger");
503         } else {
504             System.out.println("no space for baby
    tiger :(");
505 
506         }
507     }
508 
509     ArrayList<Locations> tempLocs2 = new ArrayList
    <>();
510     public void reproduceDeer(int which) {
511         tempLocs2.clear();
512         double prob;
513         if(matingSeason){
514             prob = 1;
515         } else {
516             prob = 0.7;
517         }
518         //this code is running through every
    location around the ant.
519         for (int r = allDeer.get(which).getX()-1;
```

```
519 r <= allDeer.get(which).getX()+1 && r < gameGrid.
    length && r > 0; r++) {
520             for (int c = allDeer.get(which).getY
    ()-1; c <= allDeer.get(which).getY()+1 && c <
    gameGrid[0].length && c > 0; c++) {
521                 if(checkEmptyAround(r, c) && !(
    environmentGrid[r][c] == 1)){
522                     tempLocs.add(new Locations(r,
    c));
523                 }
524             }
525         }
526         if(tempLocs.size()>0 && allDeer.get(which
    ).getAge() >= 1 && Math.random() <= prob){
527             int num = (int)(Math.random() *
    tempLocs.size());
528             allDeer.add(new Deer(tempLocs.get(num
    ).getX(), tempLocs.get(num).getY(), 0, allDeer.get
    (which).getMyFlock()));
529             gameGrid[tempLocs.get(num).getX()][
    tempLocs.get(num).getY()] = 4;
530             allDeer.get(allDeer.size()-1).
    setMother(allDeer.get(which));
531             System.out.println("new baby deer");
532         } else {
533             System.out.println("no space for baby
    deer :(");
534
535         }
536     }
537
538     int count = 0;
539     public void humanInteraction(){
540         int rate = (int)(humanBar.getValue()/10);
541         count++;
542         if(count==7){
543             for (int i = 0; i < rate; i++) {
544                 int random = (int)(Math.random()*
    bush.size());
545                 environmentGrid[bush.get(random).
    getX()][bush.get(random).getY()] = 2;
```

```
546                 bush.remove(random);
547             }
548             count = 0;
549         }
550 
551     }
552 
553     int count2 = 0;
554     public void climateChange(){
555         int rate = (int)(climateBar.getValue()/10
    );
556         System.out.println(rate);
557         count2++;
558         if(count2 == 13-rate && rate > 0){
559             salinityLvl += 1;
560             System.out.println("salinity: " +
    salinityLvl);
561             riverWidth ++;
562             setRiver(riverWidth);
563             count2 = 0;
564         }
565     }
566 
567     public void windy(){
568         int chance = (int)(Math.random() * 4);
569         if(chance == 2){
570             wind[0] = (int)(Math.random() * 3) + 1
    ;
571         } else if (chance == 3){
572             wind[0] = -1 * (int)(Math.random() * 3
    ) + 1;
573         }
574         chance = (int)(Math.random() * 4);
575         if(chance == 2){
576             wind[1] = (int)(Math.random() * 3) + 1
    ;
577         } else if (chance == 3){
578             wind[1] = -1 * (int)(Math.random() * 3
    ) + 1;
579         }
580     }
```

```
581
582     public boolean checkEmptyAround(int i, int j){
583         return gameGrid[i][j] == 0;
584     }
585 }
586
587
```

```java
package com.example.demo;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.stage.Stage;

import java.io.IOException;

public class HelloApplication extends Application {
    @Override
    public void start(Stage stage) throws IOException {
        FXMLLoader fxmlLoader = new FXMLLoader(HelloApplication.class.getResource("hello-view.fxml"));
        Scene scene = new Scene(fxmlLoader.load(), 600, 600);
        stage.setTitle("Hello!");
        stage.setScene(scene);
        stage.show();
    }

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