

```
1 package com.example.demo;
2
3 import java.util.ArrayList;
4
5 public class Deer {
6     private int x;
7     private int y;
8     private int age;
9     private int months;
10    private int gender; //0: female 1: male
11    private int hunger;
12    private int thirst;
13    private long startTime;
14    private int sleep;
15    private Deer mother;
16    private boolean hasBaby;
17    private int myFlock;
18    private boolean hasFlock;
19    private boolean noticedPredator;
20    private Tiger predator;
21
22    public Deer(int x, int y, int a, int f){
23        this.x = x;
24        this.y = y;
25        age = a;
26        months = 0;
27        gender = (int)(Math.random() * 2);
28        hunger = 2;
29        thirst = 2;
30        sleep = 0;
31        startTime = System.nanoTime();
32        mother = null;
33        hasBaby = false;
34        myFlock = f;
35        hasFlock = true;
36    }
37
38    public void changeLoc(int[][][] gameGrid,
39                          ArrayList<ArrayList<Deer>> allFlock, ArrayList<
40                          Locations> water, ArrayList<Locations> bush){
41        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
46                 .length-1){
47                 tempx++;
48             }else if(tempx > 0){
49                 tempx--;
50             }
51             if(Math.random()>.5 && tempy < gameGrid
52                 [0].length-1){
53                 tempy++;
54             }else if(tempy > 0) {
55                 tempy--;
56             }
57             if (gameGrid[tempx][tempy]==0 && tempx
58             >= 0 && tempx <= gameGrid.length && tempy >= 0 &&
59             tempy <= gameGrid[0].length) {
60                 if(hasFlock && thirst < 20 &&
61                 hunger < 15 && myFlock >= 0){
62                     if(!this.equals(allFlock.get(
63                         myFlock).get(0))) {
64                         if (allFlock.get(myFlock).
65                             get(0).getX() >= tempx - 3 && allFlock.get(myFlock
66                             ).get(0).getX() <= tempx + 3 &&
67                             allFlock.get(
68                             myFlock).get(0).getY() >= tempy - 3 && allFlock.get
69                             (myFlock).get(0).getY() <= tempy + 3) {
70                             check = true;
71                             if (age <= 1) {
72                                 gameGrid[tempx][
73                                     tempy] = 4;
74                             } else {
75                                 gameGrid[tempx][
76                                     tempy] = gender + 5;
77                             }
78                             gameGrid[x][y] = 0;
79                             x = tempx;
80                             y = tempy;
81                         }
82                     }
83                 }
84             }
85         }
86     }
87 }
```

```

69
70
71
72
73
74
75
76
77
78
79
80
81 // mother.hasBaby = true;
82 } else {
83     if (hunger >= 15) {
84         if (this.placeDistance(
85             bush, tempx, tempy) < this.placeDistance(bush, x,
86             y)) {
87             check = true;
88             gameGrid[tempx][tempy]
89             ] = gender + 5;
90         }
91         }else if (thirst >= 20) {
92             if (this.placeDistance(
93                 water, tempx, tempy) < this.placeDistance(water, x,
94                 y)) {
95                 check = true;
96                 gameGrid[tempx][tempy]
97                 ] = gender + 5;
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<
136 Deer>> allFlock, ArrayList<Deer> tempGrid){
137         if(myFlock >= 0 && myFlock < allFlock.size
138         () ){
139             //checks if deer is atleast 5 away
140             from flock gorup leader
141             if(allFlock.get(myFlock).get(0).getX
142             () >=x-5 && allFlock.get(myFlock).get(0).getX()<=x
143             +5 &&
144                 allFlock.get(myFlock).get(0).
145                 getY() >=y-5 && allFlock.get(myFlock).get(0).getY
146                 ()<=y+5){
147                     hasFlock = true;
148                 } else {
149                     System.out.println("lost the fam
150 bro");
151                     hasFlock = false;
152                     //remove deer from flock array
153                     allFlock.get(myFlock).remove(this
154 );
155
156                     if(allFlock.get(myFlock).size
157                     () == 0){
158                         allFlock.remove(myFlock);
159                         System.out.println("removed
160 flock: " + myFlock);
161                         for (int j = 0; j < tempGrid.
162                         size(); j++) {
163                             if(tempGrid.get(j).
164                             isHasFlock() && tempGrid.get(j).getMyFlock()>
165                             myFlock){
166                                 tempGrid.get(j).
167                                 setMyFlock(tempGrid.get(j).getMyFlock()-1);
168                             }
169                         }
170                     }
171                 }
172             }
173         }
174     }
175 }
```

```

155         }
156
157         myFlock = -1;
158     }
159 } else {
160     hasFlock = false;
161     for (int i = 0; i < allFlock.size(); i
162         ++
163         if(allFlock.get(i).size() == 0){
164             allFlock.remove(i);
165             System.out.println("removed
166 flock: " + i);
167             for (int j = 0; j < tempGrid.
168             size(); j++) {
169                 if(tempGrid.get(j).
170                 isHasFlock() && tempGrid.get(j).getMyFlock()>i){
171                     tempGrid.get(j).
172                     setMyFlock(tempGrid.get(j).getMyFlock()-1);
173                 }
174             }
175         }
176         //if lone deer encounters new flock it
177         //will join as long as there is enough space
178         for (int i = 0; i < allFlock.size(); i
179         ++
180         if(allFlock.get(i).get(0).getX
181         () >=x-5 && allFlock.get(i).get(0).getX()<=x+5 &&
182         allFlock.get(i).get(0).
183         getY() >=y-5 && allFlock.get(i).get(0).getY()<=y+5
184         && allFlock.get(i).size() <=7 {
185             System.out.println("found
186             another flock besties");
187             hasFlock = true;
188             myFlock = i;
189         }
190     }
191
192     //if there are two lone deer they
193     //create their own lone flock
194     for (int i = 0; i < tempGrid.size(); i

```

```

183 ++ ) {
184             if(tempGrid.get(i).getX() >=x-5
185               && tempGrid.get(0).getX()<=x+5 &&
186               tempGrid.get(0).getY() >=y
187               -5 && tempGrid.get(0).getY()<=y+5 && tempGrid.get(
188               i).getMyFlock() < 0 && myFlock < 0){
189               allFlock.add(new ArrayList
190               <>());
191               allFlock.get(allFlock.size()-1
192               ).add(tempGrid.get(i));
193               tempGrid.get(i).setMyFlock(
194               allFlock.size()-1);
195               tempGrid.get(i).setHasFlock(
196               true);
197               allFlock.get(allFlock.size()-1
198               ).add(this);
199               hasFlock = true;
200               myFlock = allFlock.size()-1;
201               System.out.println("CREATED A
202               FLOCK");
203           }
204       }
205   }
206 }
207
208     public void checkPredator(ArrayList<Tiger>
209     tempGrid){
210         for (Tiger e: tempGrid) {
211             if(e.getX()>=x-3 && e.getX()<=x+3 && e
212             .getY() >=y-3 && e.getY()<=y+3 && e.getAge() >=
213             3){
214                 if(e.isHunting()){
215                     if(e.getPrey().equals(this
216                     ) && Math.random() <= .1{
217                         noticedPredator = true;
218                         predator = e;
219                     } else if(!e.getPrey().equals(
220                     this) && Math.random() <= .4{
221                         noticedPredator = true;
222                         predator = e;
223                     }
224                 }
225             }
226         }
227     }
228 }
```

```

210             } else if (Math.random() <= .6){
211                 noticedPredator = true;
212                 predator = e;
213             }
214         }
215     }
216 }
217
218     public void runningAway(int[][] gameGrid,
219     ArrayList<Tiger> allTiger){
220         ArrayList<Locations> tiger = new ArrayList
221         <>();
222         ArrayList<Locations> tempLocs = new
223         ArrayList<>();
224         double distance;
225         Locations which = new Locations(x, y);
226
227         //adding every location around the tiger
228         //to an array
229         if(allTiger.contains(predator) && predator
230             .getX()>=x-6 && predator.getX()<=x+6 && predator.
231             getY() >=y-6 && predator.getY()<=y+6){
232             tiger.add(new Locations(predator.getX()
233             , predator.getY()));
234             distance = this.placeDistance(tiger, x
235             , y);
236             for (int r = x-1; r <= x+1 && r <
237                 gameGrid.length && r > 0; r++) {
238                 for (int c = y-1; c <= y+1 && c <
239                     gameGrid[0].length && c > 0; c++) {
240                     if(checkEmptyAround(r, c,
241                         gameGrid)){
242                         tempLocs.add(new Locations
243                         (r, c));
244                     }
245                 }
246             }
247
248             //looking through every location
249             //around the tiger and determining which is the
250             //closest to the target

```

```

237         //if there is no closest location the
238         tiger remains in the same spot.
239         for(Locations e: tempLocs){
240             if(placeDistance(tiger, e.getX(),
241 e.getY()) > distance){
242                 distance = placeDistance(tiger
243 , e.getX(), e.getY());
244                 which.setX(e.getX());
245                 which.setY(e.getY());
246             }
247         } else {
248             predator = null;
249             noticedPredator = false;
250         }
251         if(age<=1){
252             gameGrid[which.getX()][which.getY()]=
253             4;
254         } else {
255             gameGrid[which.getX()][which.getY()]=
256             gender + 5;
257         }
258
259     public boolean willDie(){
260         if(hunger >= 15 && thirst >= 20){
261             if(age < 2 && Math.random() >= age * .
262             3){
263                 System.out.println("shawty so
264                 young she couldnt handle the pain bro");
265                 return true;
266             } else if (Math.random() < age * .1){
267                 System.out.println("this was kinda
268                 random bro");
269                 return true;
270             }
271         } else if (thirst >= 40){
272             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
```

```
1 package com.example.demo;
2
3 import java.util.ArrayList;
4
5 public class Tiger {
6     private int x;
7     private int y;
8     private int age;
9     private int months;
10    private int gender; //0: female 1: male
11    private int hunger;
12    private int thirst;
13    private long startTime;
14    private int sleep;
15    private Tiger mother;
16    private boolean hasBaby;
17    private Deer prey;
18    private boolean hunting;
19    private Tiger targetMate;
20
21    public Tiger(int x, int y, int a){
22        this.x = x;
23        this.y = y;
24        age = a;
25        months = 0;
26        gender = (int)(Math.random() * 2);
27        hunger = 2;
28        thirst = 2;
29        sleep = 0;
30        startTime = System.nanoTime();
31        mother = null;
32        hasBaby = false;
33        prey = null;
34        hunting = false;
35    }
36
37    public void changeLoc(int[][][] gameGrid,
38        ArrayList<Locations> water){
39        int timeout=0;
40        boolean check = false;
41        while(!check && timeout<1000){
```

```

41           timeout++;
42           int tempx = x;
43           int tempy = y;
44           if(Math.random()>.5 && tempx < gameGrid
45               .length-1){
46                   tempx++;
47           }else if(tempx > 0){
48               tempx--;
49           }
50           if(Math.random()>.5 && tempy < gameGrid
51               [0].length-1){
52                   tempy++;
53           }else if(tempy > 0) {
54               tempy--;
55           }
56           if (gameGrid[tempx][tempy]==0 && tempx
57               >= 0 && tempx <= gameGrid.length && tempy >= 0 &&
58               tempy <= gameGrid[0].length) {
59               if(age<3){
60                   if(mother.getX() >=tempx-2 &&
61                   mother.getX()<=tempx+2 &&
62                           mother.getY() >=tempy-2
63                           && mother.getY()<=tempy+2){
64                   check=true;
65                   gameGrid[tempx][tempy]= 1;
66                   gameGrid[x][y]=0;
67                   x=tempx;
68                   y=tempy;
69               }
70               mother.hasBaby = true;
71           } else {
72               //only moving to the spot if it
73               is closer to the water then the current spot
74               if(thirst>=20){
75                   if(this.placeDistance(water
76                   , tempx, tempy) < this.placeDistance(water, x, y)){
77                       System.out.println("tiger thirsty bro");
78                   }
79                   check=true;
80                   moved(gameGrid, tempx,
81                         tempy);
82               }
83           }
84       }
85   }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }
101 }
102 }
103 }
104 }
105 }
106 }
107 }
108 }
109 }
110 }
111 }
112 }
113 }
114 }
115 }
116 }
117 }
118 }
119 }
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
788 }
789 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
798 }
799 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
878 }
879 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
888 }
889 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
898 }
899 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
978 }
979 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
987 }
988 }
988 }
989 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
999 }
1000 }
1001 }
1002 }
1003 }
1004 }
1005 }
1006 }
1007 }
1008 }
1009 }
1009 }
1010 }
1011 }
1012 }
1013 }
1014 }
1015 }
1016 }
1017 }
1018 }
1019 }
1019 }
1020 }
1021 }
1022 }
1023 }
1024 }
1025 }
1026 }
1027 }
1028 }
1029 }
1029 }
1030 }
1031 }
1032 }
1033 }
1034 }
1035 }
1036 }
1037 }
1038 }
1039 }
1039 }
1040 }
1041 }
1042 }
1043 }
1044 }
1045 }
1046 }
1047 }
1048 }
1049 }
1049 }
1050 }
1051 }
1052 }
1053 }
1054 }
1055 }
1056 }
1057 }
1058 }
1059 }
1059 }
1060 }
1061 }
1062 }
1063 }
1064 }
1065 }
1066 }
1067 }
1068 }
1069 }
1069 }
1070 }
1071 }
1072 }
1073 }
1074 }
1075 }
1076 }
1077 }
1078 }
1078 }
1079 }
1079 }
1080 }
1081 }
1082 }
1083 }
1084 }
1085 }
1086 }
1086 }
1087 }
1087 }
1088 }
1088 }
1089 }
1089 }
1090 }
1091 }
1092 }
1093 }
1094 }
1095 }
1095 }
1096 }
1096 }
1097 }
1097 }
1098 }
1099 }
1099 }
1100 }
1101 }
1102 }
1103 }
1104 }
1105 }
1106 }
1107 }
1108 }
1109 }
1109 }
1110 }
1111 }
1112 }
1113 }
1114 }
1115 }
1116 }
1117 }
1118 }
1119 }
1119 }
1120 }
1121 }
1122 }
1123 }
1124 }
1125 }
1126 }
1127 }
1128 }
1129 }
1129 }
1130 }
1131 }
1132 }
1133 }
1134 }
1135 }
1136 }
1137 }
1138 }
1139 }
1139 }
1140 }
1141 }
1142 }
1143 }
1144 }
1145 }
1146 }
1147 }
1148 }
1149 }
1149 }
1150 }
1151 }
1152 }
1153 }
1154 }
1155 }
1156 }
1157 }
1158 }
1159 }
1159 }
1160 }
1161 }
1162 }
1163 }
1164 }
1165 }
1166 }
1167 }
1168 }
1169 }
1169 }
1170 }
1171 }
1172 }
1173 }
1174 }
1175 }
1176 }
1177 }
1178 }
1178 }
1179 }
1179 }
1180 }
1181 }
1182 }
1183 }
1184 }
1185 }
1186 }
1186 }
1187 }
1187 }
1188 }
1188 }
1189 }
1189 }
1190 }
1191 }
1192 }
1193 }
1194 }
1195 }
1195 }
1196 }
1196 }
1197 }
1197 }
1198 }
1198 }
1199 }
1199 }
1200 }
1201 }
1202 }
1203 }
1204 }
1205 }
1206 }
1207 }
1208 }
1209 }
1209 }
1210 }
1211 }
1212 }
1213 }
1214 }
1215 }
1216 }
1217 }
1218 }
1219 }
1219 }
1220 }
1221 }
1222 }
1223 }
1224 }
1225 }
1226 }
1227 }
1228 }
1229 }
1229 }
1230 }
1231 }
1232 }
1233 }
1234 }
1235 }
1236 }
1237 }
1238 }
1239 }
1239 }
1240 }
1241 }
1242 }
1243 }
1244 }
1245 }
1246 }
1247 }
1248 }
1249 }
1249 }
1250 }
1251 }
1252 }
1253 }
1254 }
1255 }
1256 }
1257 }
1258 }
1259 }
1259 }
1260 }
1261 }
1262 }
1263 }
1264 }
1265 }
1266 }
1267 }
1268 }
1269 }
1269 }
1270 }
1271 }
1272 }
1273 }
1274 }
1275 }
1276 }
1277 }
1278 }
1278 }
1279 }
1279 }
1280 }
1281 }
1282 }
1283 }
1284 }
1285 }
1286 }
1286 }
1287 }
1287 }
1288 }
1288 }
1289 }
1289 }
1290 }
1291 }
1292 }
1293 }
1294 }
1295 }
1295 }
1296 }
1296 }
1297 }
1297 }
1298 }
1298 }
1299 }
1299 }
1300 }
1301 }
1302 }
1303 }
1304 }
1305 }
1306 }
1307 }
1308 }
1309 }
1309 }
1310 }
1311 }
1312 }
1313 }
1314 }
1315 }
1316 }
1317 }
1318 }
1319 }
1319 }
1320 }
1321 }
1322 }
1323 }
1324 }
1325 }
1326 }
1327 }
1328 }
1329 }
1329 }
1330 }
1331 }
1332 }
1333 }
1334 }
1335 }
1336 }
1337 }
1338 }
1339 }
1339 }
1340 }
1341 }
1342 }
1343 }
1344 }
1345 }
1346 }
1347 }
1348 }
1349 }
1349 }
1350 }
1351 }
1352 }
1353 }
1354 }
1355 }
1356 }
1357 }
1358 }
1359 }
1359 }
1360 }
1361 }
1362 }
1363 }
1364 }
1365 }
1366 }
1367 }
1368 }
1369 }
1369 }
1370 }
1371 }
1372 }
1373 }
1374 }
1375 }
1376 }
1377 }
1378 }
1378 }
1379 }
1379 }
1380 }
1381 }
1382 }
1383 }
1384 }
1385 }
1386 }
1386 }
1387 }
1387 }
1388 }
1388 }
1389 }
1389 }
1390 }
1391 }
1392 }
1393 }
1394 }
1395 }
1395 }
1396 }
1396 }
1397 }
1397 }
1398 }
1398 }
1399 }
1399 }
1400 }
1401 }
1402 }
1403 }
1404 }
1405 }
1406 }
1407 }
1408 }
1409 }
1409 }
1410 }
1411 }
1412 }
1413 }
1414 }
1415 }
1416 }
1417 }
1418 }
1419 }
1419 }
1420 }
1421 }
1422 }
1423 }
1424 }
1425 }
1426 }
1427 }
1428 }
1429 }
1429 }
1430 }
1431 }
1432 }
1433 }
1434 }
1435 }
1436 }
1437 }
1438 }
1439 }
1439 }
1440 }
1441 }
1442 }
1443 }
1444 }
1445 }
1446 }
1447 }
1448 }
1449 }
1449 }
1450 }
1451 }
1452 }
1453 }
1454 }
1455 }
1456 }
1457 }
1458 }
1459 }
1459 }
1460 }
1461 }
1462 }
1463 }
1464 }
1465 }
1466 }
1467 }
1468 }
1469 }
1469 }
1470 }
1471 }
1472 }
1473 }
1474 }
1475 }
1476 }
1477 }
1478 }
1478 }
1479 }
1479 }
1480 }
1481 }
1482 }
1483 }
1484 }
1485 }
1486 }
1486 }
1487 }
1487 }
1488 }
1488 }
1489 }
1489 }
1490 }
1491 }
1492 }
1493 }
1494 }
1495 }
1495 }
1496 }
1496 }
1497 }
1497 }
1498 }
1498 }
1499 }
1499 }
1500 }
1501 }
1502 }
1503 }
1504 }
1505 }
1506 }
1507 }
1508 }
1509 }
1509 }
1510 }
1511 }
1512 }
1513 }
1514 }
1515 }
1516 }
1517 }
1518 }
1519 }
1519 }
1520 }
1521 }
1522 }
1523 }
1524 }
1525 }
1526 }
1527 }
1528 }
1529 }
1529 }
1530 }
1531 }
1532 }
1533 }
1534 }
1535 }
1536 }
1537 }
1538 }
1539 }
1539 }
1540 }
1541 }
1542 }
1543 }
1544 }
1545 }
1546 }
1547 }
1548 }
1549 }
1549 }
1550 }
1551 }
1552 }
1553 }
1554 }
1555 }
1556 }
1557 }
1558 }
1558 }
1559 }
1559 }
1560 }
1561 }
1562 }
1563 }
1564 }
1565 }
1566 }
1567 }
1568 }
1569 }
1569 }
1570 }
1571 }
1572 }
1573 }
1574 }
1575 }
1576 }
1577 }
1578 }
1578 }
1579 }
1579 }
1580 }
1581 }
1582 }
1583 }
1584 }
1585 }
1586 }
1586 }
1587 }
1587 }
1588 }
1588 }
1589 }
1589 }
1590 }
1591 }
1592 }
1593 }
1594 }
1595 }
1595 }
1596 }
1596 }
1597 }
1597 }
1598 }
1598 }
1599 }
1599 }
1600 }
1601 }
1602 }
1603 }
1604 }
1605 }
1606 }
1607 }
1608 }
1609 }
1609 }
1610 }
1611 }
1612 }
1613 }
1614 }
1615 }
1616 }
1617 }
1618 }
1619 }
1619 }
1620 }
1621 }
1622 }
1623 }
1624 }
1625 }
1626 }
1627 }
1628 }
1629 }
1629 }
1630 }
1631 }
1632 }
1633 }
1634 }
1635 }
1636 }
1637 }
1638 }
1639 }
1639 }
1640 }
1641 }
1642 }
1643 }
1644 }
1645 }
1646 }
1647 }
1648 }
1649 }
1649 }
1650 }
1651 }
1652 }
1653 }
1654 }
1655 }
1656 }
1657 }
1658 }
1658 }
1659 }
1659 }
1660 }
1661 }
1662 }
1663 }
1664 }
1665 }
1666 }
1667 }
1668 }
1669 }
1669 }
1670 }
1671 }
1672 }
1673 }
1674 }
1675 }
1676 }
1677 }
1678 }
1678 }
1679 }
1679 }
1680 }
1681 }
1682 }
1683 }
1684 }
1685 }
1686 }
1686 }
1687 }
1687 }
1688 }
1688 }
1689 }
1689 }
1690 }
1691 }
1692 }
1693 }
1694 }
1695 }
1695 }
1696 }
1696 }
1697 }
1697 }
1698 }
1698 }
1699 }
1699 }
1700 }
1701 }
1702 }
1703 }
1704 }
1705 }
1706 }
1707 }
1708 }
1709 }
1709 }
1710 }
1711 }
1712 }
1713 }
1714 }
1715 }
1716 }
1717 }
1718 }
1719 }
1719 }
1720 }
1721 }
1722 }
1723 }
1724 }
1725 }
1726 }
1727 }
1728 }
1729 }
1729 }
1730 }
1731 }
1732 }
1733 }
1734 }
1735 }
1736 }
1737 }
1738 }
1739 }
1739 }
1740 }
1741 }
1742 }
1743 }
1744 }
1745 }
1746 }
1747 }
1748 }
1749 }
1749 }
1750 }
1751 }
1752 }
1753 }
1754 }
1755 }
1756 }
1757 }
1758 }
1758 }
1759 }
1759 }
1760 }
1761 }
1762 }
1763 }
1764 }
1765 }
1766 }
1767 }
1768 }
1769 }
1769 }
1770 }
1771 }
1772 }
1773 }
1774 }
1775 }
1776 }
1777 }
1778 }
1778 }
1779 }
1779 }
1780 }
1781 }
1782 }
1783 }
1784 }
1785 }
1786 }
1786 }
1787 }
1787 }
1788 }
1788 }
1789 }
1789 }
1790 }
1791 }
1792 }
1793 }
1794 }
1795 }
1795 }
1796 }
1796 }
1797 }
1797 }
1798 }
1798 }
1799 }
1799 }
1800 }
1801 }
1802 }
1803 }
1804 }
1805 }
1806 }
1807 }
1808 }
1809 }
1809 }
1810 }
1811 }
1812 }
1813 }
1814 }
1815 }
1816 }
1817 }
1818 }
1819 }
1819 }
1820 }
1821 }
1822 }
1823 }
1824 }
1825 }
1826 }
1827 }
1828 }
1829 }
1829 }
1830 }
1831 }
1832 }
1833 }
1834 }
1835 }
1836 }
1837 }
1838 }
1839 }
1839 }
1840 }
1841 }
1842 }
1843 }
1844 }
1845 }
1846 }
1847 }
1848 }
1849 }
1849 }
1850 }
1851 }
1852 }
1853 }
1854 }
1855 }
1856 }
1857 }
1858 }
1858 }
1859 }
1859 }
1860 }
1861 }
1862 }
1863 }
1864 }
1865 }
1866 }
1867 }
1868 }
1869 }
1869 }
1870 }
1871 }
1872 }
1873 }
1874 }
1875 }
1876 }
1877 }
1878 }
1878 }
1879 }
1879 }
1880 }
1881 }
1882 }
1883 }
1884 }
1885 }
1886 }
1886 }
1887 }
1887 }
1888 }
1888 }
1889 }
1889 }
1890 }
1891 }
1892 }
1893 }
1894 }
1895 }
1895 }
1896 }
1896 }
1897 }
1897 }
1898 }
1898 }
1899 }
1899 }
1900 }
1901 }
1902 }
1903 }
1904 }
1905 }
1906 }
1907 }
1908 }
1909 }
1909 }
1910 }
1911 }
1912 }
1913 }
1914 }
1915 }
1916 }
1917 }
1918 }
1919 }
1919 }
1920 }
1921 }
1922 }
1923 }
1924 }
1925 }
1926 }
1927 }
1928 }
1929 }
1929 }
1930 }
1931 }
1932 }
1933 }
1934 }
1935 }
1936 }
1937 }
1938 }
1939 }
1939 }
1940 }
1941 }
1942 }
1943 }
1944 }
1945 }
1946 }
1947 }
1948 }
1949 }
1949 }
1950 }
1951 }
1952 }
1953 }
1954 }
1955 }
1956 }
1957 }
1958 }
1958 }
1959 }
1959 }
1960 }
1961 }
1962 }
1963 }
1964 }
1965 }
1966 }
1967 }
1968 }
1969 }
1969 }
1970 }
1971 }
1972 }
1973 }
1974 }
1975 }
1976 }
1977 }
1978 }
1978 }
1979 }
1979 }
1980 }
1981 }
1982 }
1983 }
1984 }
1985 }
1986 }
1986 }
1987 }
1987 }
1988 }
1988 }
1989 }
1989 }
1990 }
1991 }
1992 }
1993 }
1994 }
1995 }
1995 }
1996 }
1996 }
1997 }
1997 }
1998 }
1998 }
1999 }
1999 }
2000 }
2001 }
2002 }
2003 }
2004 }
2005 }
2006 }
2007 }
2008 }
2009 }
2009 }
2010 }
2011 }
2012 }
2013 }
2014 }
2015 }
2016 }
2017 }
2018 }
2019 }
2019 }
2020 }
2021 }
2022 }
2023 }
2024 }
2025 }
2026 }
2027 }
2028 }
2029 }
2029 }
2030 }
2031 }
2032 }
2033 }
2034 }
2035 }
2036 }
2037 }
2038 }
2039 }
2039 }
2040 }
2041 }
2042 }
2043 }
2044 }
2045 }
2046 }
2047 }
2048 }
2049 }
2049 }
2050 }
2051 }
2052 }
2053 }
2054 }
2055 }
2056 }
2057 }
2058 }
2058 }
2059 }
2059 }
2060 }
2061 }
2062 }
2063 }
2064 }
2065 }
2066 }
2067 }
2068 }
2069 }
2069 }
2070 }
2071 }
2072 }
2073 }
2074 }
2075 }
2076 }
2077 }
2078 }
2078 }
2079 }
2079 }
2080 }
2081 }
2082 }
2083 }
2084 }
2085 }
2086 }
2086 }
2087 }
2087 }
2088 }
2088 }
2089 }
2089 }
2090 }
2091 }
2092 }
2093 }
2094 }
2095 }
2095 }
2096 }
2096 }
2097 }
2097 }
2098 }
2098 }
2099 }
2099 }
2100 }
2101 }
2102 }
2103 }
2104 }
2105 }
2106 }
2107 }
2108 }
2109 }
2109 }
2110 }
2111 }
2112 }
2113 }
2114 }
2115 }
2116 }
2117 }
2118 }
2119 }
2119 }
2120 }
2121 }
2122 }
2123 }
2124 }
2125 }
2126 }
2127 }
2128 }
2129 }
2129 }
2130 }
2131 }
2132 }
2133 }
2134 }
2135 }
2136 }
2137 }
2138 }
2139 }
2139 }
2140 }
2141 }
2142 }
2143 }

```

```

72                     }
73             } else {
74                 check=true;
75                 moved(gameGrid, tempx,
76                         tempy);
77             }
78         }
79     }
80 //           System.out.println("x: " + x);
81 }
82
83     public void moved(int[][] gameGrid, int tempx
84 , int tempy){
85         gameGrid[tempx][tempy]= gender + 2;
86         gameGrid[x][y]=0;
87         x=tempx;
88         y=tempy;
89     }
90
91     public boolean checkingOpposite(ArrayList<
Tiger> tempGrid){
92         //check how many squares away.
93         for (int i = 0; i < tempGrid.size(); i
94 ++
95         {
96             if(Math.abs(tempGrid.get(i).getX() - x
97 ) == 1 && Math.abs(tempGrid.get(i).getY() - y) ==
98 1 && gender!=tempGrid.get(i).getGender() &&
99 tempGrid.get(i).getAge() >= 3 && age >= 3 && !
hasBaby){
100                 if(tempGrid.get(i).getMother() !=
null && !tempGrid.get(i).getMother().equals(this
))
{
101                     return true;
102                 } else if (tempGrid.get(i).
103 getMother() == null){
104                     return true;
105                 }
106                 System.out.println("smth aint
right");
107             }
108         }
109         return false;

```

```

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

```

```

127                     System.out.println("tiger
128                         changed prey");
129                     }
130                 }
131             }
132
133     public void chasingPrey(int[][] gameGrid,
134         ArrayList<Deer> allDeer){
135         ArrayList<Locations> deer = new ArrayList
136             <>();
137         ArrayList<Locations> tempLocs = new
138             ArrayList<>();
139         double distance;
140         Locations which = new Locations(x, y);
141
142         //adding every location around the tiger
143         //to an array
144         if(allDeer.contains(prey)){
145             deer.add(new Locations(prey.getX(),
146                 prey.getY()));
147             distance = this.placeDistance(deer, x
148                 , y);
149             for (int r = x-1; r <= x+1 && r <
150                 gameGrid.length && r > 0; r++) {
151                 for (int c = y-1; c <= y+1 && c <
152                     gameGrid[0].length && c > 0; c++) {
153                     if(checkEmptyAround(r, c,
154                         gameGrid)){
155                         tempLocs.add(new Locations
156                             (r, c));
157                     }
158                 }
159             }
160
161             //looking through every location
162             //around the tiger and determining which is the
163             //closest to the target
164             //if there is no closest location the
165             //tiger remains in the same spot.
166             for(Locations e: tempLocs){

```

```

154             if(placeDistance(deer, e.getX(), e
155               .getY()) < distance){
156                 distance = placeDistance(deer
157               , e.getX(), e.getY());
158                 which.setX(e.getX());
159                 which.setY(e.getY());
160             }
161         } else {
162             prey = null;
163             hunting = false;
164         }
165     moved(gameGrid, which.getX(), which.getY
166   ());
167
168     public void foundPrey(ArrayList<Deer> deer,
169       int[][][] tempGrid, ArrayList<ArrayList<Deer>>
170       allFlock, ArrayList<Tiger> tigers, Deer prey){
171       if(prey.getX() >= x - 1 && prey.getX() <=
172         x + 1 &&
173           prey.getY() >= y - 1 && prey.getY
174             () <= y + 1){
175               tempGrid[prey.getX()][prey.getY()] = 0
176 ;
177               if(prey.isHasFlock()){
178                 allFlock.get(prey.getMyFlock()).
179                   remove(prey);
180               }
181               deer.remove(prey);
182               hunger = 0;
183               thirst -= 1;
184               hunting = false;
185               this.prey = null;
186               for (Tiger a: tigers) {
187                 if(a.getAge() < 3 && a.getMother
188                   () == this){
189                   hunger = 0;
190                   thirst -= 1;
191               }
192             }
193           }
194         }
195       }
196     }
197   }
198 }

```

```

185          }
186          System.out.println("she munching bro"
187      );
188  }
189
190  public void chasingMate(int[][] gameGrid,
191      ArrayList<Tiger> allTiger){
192      ArrayList<Locations> mate = new ArrayList
193      <>();
194      ArrayList<Locations> tempLocs = new
195      ArrayList<>();
196      double distance;
197      Locations which = new Locations(x, y);
198
199      //adding every location around the tiger
200      //to an array
201      if(allTiger.contains(targetMate)){
202          mate.add(new Locations(targetMate.getX()
203          , targetMate.getY()));
204          distance = this.placeDistance(mate, x
205          , y);
206          for (int r = x-1; r <= x+1 && r <
207          gameGrid.length && r > 0; r++) {
208              for (int c = y-1; c <= y+1 && c <
209              gameGrid[0].length && c > 0; c++) {
210                  if(checkEmptyAround(r, c,
211                  gameGrid)){
212                      tempLocs.add(new Locations
213                      (r, c));
214                  }
215              }
216          }
217
218          //looking through every location
219          //around the tiger and determining which is the
220          //closest to the target
221          //if there is no closest location the
222          //tiger remains in the same spot.
223          for(Locations e: tempLocs){
224              if(placeDistance(mate, e.getX(), e

```

```

211 .getY() < distance){
212             System.out.println("THE TIGERS
213                 ARE MATING AND CHASING EACH OTHER");
214             distance = placeDistance(mate
215                 , e.getX(), e.getY());
216                 which.setX(e.getX());
217                 which.setY(e.getY());
218             }
219         } else {
220             targetMate = null;
221         }
222     moved(gameGrid, which.getX(), which.getY
223     ());
224 }
225
226     public boolean willDie(){
227         if(hunger >= 20 && thirst >= 25){
228             if(age < 3 && Math.random() >= age * .
229             3){
230                 System.out.println("deer so young
231                 it died");
232                 return true;
233             } else if (Math.random() < age * .1){
234                 System.out.println("this was kinda
235                 random bro");
236                 return true;
237             }
238         } else if (thirst >= 45){
239             System.out.println("shawty dehydrated
240                 bro");
241             return true;
242         } else if (hunger >= 35){
243             System.out.println("shawty starved bro
244                 ");
245             return true;
246         } else if (age >= 15 && Math.random() > .5
247         ){
248             System.out.println("deer very old");
249             return true;
250         }

```

```
243         return false;
244     }
245
246     public boolean checkDistance(Tiger check, int
247         dist){
248         return (check.getX() >= x - dist && check.
249             getX() <= x + dist &&
250                 check.getY() >= y - dist && check.
251             getY() <= y + dist);
252     }
253
254     public double placeDistance(ArrayList<
255         Locations> places, int x, int y){
256         double distance = 100;
257         for(Locations e: places){
258             double temp = Math.hypot(x-e.getX(), y
259             -e.getY());
260             if(temp < distance){
261                 distance = temp;
262             }
263         }
264         return distance;
265     }
266
267     public boolean inWater(int[][] envGrid){
268         return envGrid[x][y] == 1;
269     }
270
271     public boolean checkEmptyAround(int i, int j,
272         int[][][] gameGrid){
273         return gameGrid[i][j] == 0;
274     }
275
276     public int getX() {
277         return x;
278     }
279
280     public int getY() {
281         return y;
282     }
283 }
```

```
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 }
```

```
1 package com.example.demo;
2
3 public class Locations {
4     private int x;
5     private int y;
6
7     public Locations(int xx, int yy){
8         x = xx;
9         y = yy;
10    }
11
12    public int getX() {
13        return x;
14    }
15
16    public int getY() {
17        return y;
18    }
19
20    public void setX(int x) {
21        this.x = x;
22    }
23
24    public void setY(int y) {
25        this.y = y;
26    }
27 }
28
```

```
1 package com.example.demo;
2
3 import javafx.fxml.FXML;
4
5
6 //import com.sun.prism.paint.Color;
7
8 import java.util.ArrayList;
9
10 import javafx.animation.AnimationTimer;
11 import javafx.event.ActionEvent;
12 import javafx.scene.chart.LineChart;
13 import javafx.scene.chart.XYChart;
14 import javafx.scene.control.*;
15 import javafx.scene.layout.AnchorPane;
16 import javafx.scene.layout.GridPane;
17
18 public class HelloController {
19     int x = 44;
20     int y = 59;
21     Button[][] btn = new Button[x][y];
22     int[][] gameGrid = new int[x][y];
23     int[][] environmentGrid = new int[x][y];
24     int[] wind = new int[2];
25     long gameStart = System.nanoTime();
26     boolean matingSeason = false;
27     int salinityLvl = 0;
28     int riverWidth = 6;
29     ArrayList<Locations> water = new ArrayList<>();
30     ArrayList<Locations> bush = new ArrayList<>();
31     ArrayList<Tiger> allTigers = new ArrayList<>();
32     ArrayList<ArrayList<Deer>> allFlocks = new
ArrayList<>();
33     ArrayList<Deer> allDeer = new ArrayList<>();
34     //GridPane gPane = new GridPane();
35 //    Image k = new Image("resources/Koala.jpg");
36     @FXML
37     private AnchorPane aPane, keyPane;
38     @FXML
39     private GridPane gPane;
40     @FXML
```

```
41     private LineChart lChart;
42     @FXML
43     private Button startButton, tigerButton,
44         deerButton;
45     @FXML
46     private CheckBox matingCheck;
47     @FXML
48     private Slider climateBar, humanBar;
49     @FXML
50     private Label climate, human;
51
52     ArrayList<Integer> monthlyDeer = new ArrayList
53     <>();
54     ArrayList<Integer> monthlyTiger = new ArrayList
55     <>();
56     XYChart.Series deerPop = new XYChart.Series();
57     XYChart.Series tigerPop = new XYChart.Series();
58
59     @FXML
60     private void handleStart(ActionEvent event) {
61         startButton.setVisible(false);
62         tigerButton.setVisible(true);
63         deerButton.setVisible(true);
64         matingCheck.setVisible(true);
65         climateBar.setVisible(true);
66         humanBar.setVisible(true);
67         climate.setVisible(true);
68         human.setVisible(true);
69         keyPane.setVisible(true);
70         lChart.setVisible(true);
71         lChart.getYAxis().setLabel("POPULATION");
72         lChart.getXAxis().setLabel("MONTHS");
73         //after adding the grpipane in scene
74         // builder, modify the fxml manually to
75         // eliminate
76         // rows and columns
77         // gPane.setMinSize(0,0);
78         //gPane.setPadding(new Insets(btn[i][j]));
79         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
86                 //values i,j
87                 btn[i][j] = new Button();
88                 btn[i][j].setStyle("-fx-background
89                 -color:#d3d3d3");
90
91                 btn[i][j].setMinWidth(20);
92                 btn[i][j].setMinHeight(20);
93
94                 btn[i][j].setMaxWidth(20);
95                 btn[i][j].setMaxHeight(20);
96
97                 //btn[i][j].setPrefSize(25, 5);
98                 //Paramters: object, columns,
99                 //rows
100
101
102             }
103         }
104
105         gPane.setGridLinesVisible(true);
106
107         gPane.setVisible(true);
108
109         for (int i = 0; i < environmentGrid.length
110 ; i++) {
111             for (int j = 0; j < environmentGrid[0
112 ].length; j++) {
113                 environmentGrid[i][j]=2;
114             }
115         }
116
117     }
118
119     @Override
120     public void start(Stage stage) throws IOException {
121
122         stage.setTitle("Hello JavaFX");
123
124         stage.setScene(scene);
125
126         stage.show();
127     }
128
129     @Override
130     public void stop() {
131
132         Platform.exit();
133     }
134
135     @Override
136     public void handleEvent(Event event) {
137
138         if (event.getSource() == button) {
139
140             System.out.println("Button Clicked!");
141
142             for (int i = 0; i < environmentGrid.length;
143                 i++) {
144                 for (int j = 0; j < environmentGrid[0].length;
145                     j++) {
146
147                     if (environmentGrid[i][j] == 1) {
148
149                         environmentGrid[i][j] = 2;
150
151                         button.fire();
152
153                         environmentGrid[i][j] = 1;
154
155                     }
156
157                 }
158             }
159
160         }
161
162     }
163
164     @Override
165     public void handleEvent(Event event) {
166
167         if (event.getSource() == button) {
168
169             System.out.println("Button Clicked!");
170
171             for (int i = 0; i < environmentGrid.length;
172                 i++) {
173                 for (int j = 0; j < environmentGrid[0].length;
174                     j++) {
175
176                     if (environmentGrid[i][j] == 1) {
177
178                         environmentGrid[i][j] = 2;
179
180                         button.fire();
181
182                         environmentGrid[i][j] = 1;
183
184                     }
185
186                 }
187             }
188
189         }
190
191     }
192
193     @Override
194     public void handleEvent(Event event) {
195
196         if (event.getSource() == button) {
197
198             System.out.println("Button Clicked!");
199
200             for (int i = 0; i < environmentGrid.length;
201                 i++) {
202                 for (int j = 0; j < environmentGrid[0].length;
203                     j++) {
204
205                     if (environmentGrid[i][j] == 1) {
206
207                         environmentGrid[i][j] = 2;
208
209                         button.fire();
210
211                         environmentGrid[i][j] = 1;
212
213                     }
214
215                 }
216             }
217
218         }
219
220     }
221
222     @Override
223     public void handleEvent(Event event) {
224
225         if (event.getSource() == button) {
226
227             System.out.println("Button Clicked!");
228
229             for (int i = 0; i < environmentGrid.length;
230                 i++) {
231                 for (int j = 0; j < environmentGrid[0].length;
232                     j++) {
233
234                     if (environmentGrid[i][j] == 1) {
235
236                         environmentGrid[i][j] = 2;
237
238                         button.fire();
239
240                         environmentGrid[i][j] = 1;
241
242                     }
243
244                 }
245             }
246
247         }
248
249     }
250
251     @Override
252     public void handleEvent(Event event) {
253
254         if (event.getSource() == button) {
255
256             System.out.println("Button Clicked!");
257
258             for (int i = 0; i < environmentGrid.length;
259                 i++) {
260                 for (int j = 0; j < environmentGrid[0].length;
261                     j++) {
262
263                     if (environmentGrid[i][j] == 1) {
264
265                         environmentGrid[i][j] = 2;
266
267                         button.fire();
268
269                         environmentGrid[i][j] = 1;
270
271                     }
272
273                 }
274             }
275
276         }
277
278     }
279
280     @Override
281     public void handleEvent(Event event) {
282
283         if (event.getSource() == button) {
284
285             System.out.println("Button Clicked!");
286
287             for (int i = 0; i < environmentGrid.length;
288                 i++) {
289                 for (int j = 0; j < environmentGrid[0].length;
290                     j++) {
291
292                     if (environmentGrid[i][j] == 1) {
293
294                         environmentGrid[i][j] = 2;
295
296                         button.fire();
297
298                         environmentGrid[i][j] = 1;
299
300                     }
301
302                 }
303             }
304
305         }
306
307     }
308
309     @Override
310     public void handleEvent(Event event) {
311
312         if (event.getSource() == button) {
313
314             System.out.println("Button Clicked!");
315
316             for (int i = 0; i < environmentGrid.length;
317                 i++) {
318                 for (int j = 0; j < environmentGrid[0].length;
319                     j++) {
320
321                     if (environmentGrid[i][j] == 1) {
322
323                         environmentGrid[i][j] = 2;
324
325                         button.fire();
326
327                         environmentGrid[i][j] = 1;
328
329                     }
330
331                 }
332             }
333
334         }
335
336     }
337
338     @Override
339     public void handleEvent(Event event) {
340
341         if (event.getSource() == button) {
342
343             System.out.println("Button Clicked!");
344
345             for (int i = 0; i < environmentGrid.length;
346                 i++) {
347                 for (int j = 0; j < environmentGrid[0].length;
348                     j++) {
349
350                     if (environmentGrid[i][j] == 1) {
351
352                         environmentGrid[i][j] = 2;
353
354                         button.fire();
355
356                         environmentGrid[i][j] = 1;
357
358                     }
359
360                 }
361             }
362
363         }
364
365     }
366
367     @Override
368     public void handleEvent(Event event) {
369
370         if (event.getSource() == button) {
371
372             System.out.println("Button Clicked!");
373
374             for (int i = 0; i < environmentGrid.length;
375                 i++) {
376                 for (int j = 0; j < environmentGrid[0].length;
377                     j++) {
378
379                     if (environmentGrid[i][j] == 1) {
380
381                         environmentGrid[i][j] = 2;
382
383                         button.fire();
384
385                         environmentGrid[i][j] = 1;
386
387                     }
388
389                 }
390             }
391
392         }
393
394     }
395
396     @Override
397     public void handleEvent(Event event) {
398
399         if (event.getSource() == button) {
400
401             System.out.println("Button Clicked!");
402
403             for (int i = 0; i < environmentGrid.length;
404                 i++) {
405                 for (int j = 0; j < environmentGrid[0].length;
406                     j++) {
407
408                     if (environmentGrid[i][j] == 1) {
409
410                         environmentGrid[i][j] = 2;
411
412                         button.fire();
413
414                         environmentGrid[i][j] = 1;
415
416                     }
417
418                 }
419             }
420
421         }
422
423     }
424
425     @Override
426     public void handleEvent(Event event) {
427
428         if (event.getSource() == button) {
429
430             System.out.println("Button Clicked!");
431
432             for (int i = 0; i < environmentGrid.length;
433                 i++) {
434                 for (int j = 0; j < environmentGrid[0].length;
435                     j++) {
436
437                     if (environmentGrid[i][j] == 1) {
438
439                         environmentGrid[i][j] = 2;
440
441                         button.fire();
442
443                         environmentGrid[i][j] = 1;
444
445                     }
446
447                 }
448             }
449
450         }
451
452     }
453
454     @Override
455     public void handleEvent(Event event) {
456
457         if (event.getSource() == button) {
458
459             System.out.println("Button Clicked!");
460
461             for (int i = 0; i < environmentGrid.length;
462                 i++) {
463                 for (int j = 0; j < environmentGrid[0].length;
464                     j++) {
465
466                     if (environmentGrid[i][j] == 1) {
467
468                         environmentGrid[i][j] = 2;
469
470                         button.fire();
471
472                         environmentGrid[i][j] = 1;
473
474                     }
475
476                 }
477             }
478
479         }
480
481     }
482
483     @Override
484     public void handleEvent(Event event) {
485
486         if (event.getSource() == button) {
487
488             System.out.println("Button Clicked!");
489
490             for (int i = 0; i < environmentGrid.length;
491                 i++) {
492                 for (int j = 0; j < environmentGrid[0].length;
493                     j++) {
494
495                     if (environmentGrid[i][j] == 1) {
496
497                         environmentGrid[i][j] = 2;
498
499                         button.fire();
500
501                         environmentGrid[i][j] = 1;
502
503                     }
504
505                 }
506             }
507
508         }
509
510     }
511
512     @Override
513     public void handleEvent(Event event) {
514
515         if (event.getSource() == button) {
516
517             System.out.println("Button Clicked!");
518
519             for (int i = 0; i < environmentGrid.length;
520                 i++) {
521                 for (int j = 0; j < environmentGrid[0].length;
522                     j++) {
523
524                     if (environmentGrid[i][j] == 1) {
525
526                         environmentGrid[i][j] = 2;
527
528                         button.fire();
529
530                         environmentGrid[i][j] = 1;
531
532                     }
533
534                 }
535             }
536
537         }
538
539     }
540
541     @Override
542     public void handleEvent(Event event) {
543
544         if (event.getSource() == button) {
545
546             System.out.println("Button Clicked!");
547
548             for (int i = 0; i < environmentGrid.length;
549                 i++) {
550                 for (int j = 0; j < environmentGrid[0].length;
551                     j++) {
552
553                     if (environmentGrid[i][j] == 1) {
554
555                         environmentGrid[i][j] = 2;
556
557                         button.fire();
558
559                         environmentGrid[i][j] = 1;
560
561                     }
562
563                 }
564             }
565
566         }
567
568     }
569
570     @Override
571     public void handleEvent(Event event) {
572
573         if (event.getSource() == button) {
574
575             System.out.println("Button Clicked!");
576
577             for (int i = 0; i < environmentGrid.length;
578                 i++) {
579                 for (int j = 0; j < environmentGrid[0].length;
580                     j++) {
581
582                     if (environmentGrid[i][j] == 1) {
583
584                         environmentGrid[i][j] = 2;
585
586                         button.fire();
587
588                         environmentGrid[i][j] = 1;
589
590                     }
591
592                 }
593             }
594
595         }
596
597     }
598
599     @Override
600     public void handleEvent(Event event) {
601
602         if (event.getSource() == button) {
603
604             System.out.println("Button Clicked!");
605
606             for (int i = 0; i < environmentGrid.length;
607                 i++) {
608                 for (int j = 0; j < environmentGrid[0].length;
609                     j++) {
610
611                     if (environmentGrid[i][j] == 1) {
612
613                         environmentGrid[i][j] = 2;
614
615                         button.fire();
616
617                         environmentGrid[i][j] = 1;
618
619                     }
620
621                 }
622             }
623
624         }
625
626     }
627
628     @Override
629     public void handleEvent(Event event) {
630
631         if (event.getSource() == button) {
632
633             System.out.println("Button Clicked!");
634
635             for (int i = 0; i < environmentGrid.length;
636                 i++) {
637                 for (int j = 0; j < environmentGrid[0].length;
638                     j++) {
639
640                     if (environmentGrid[i][j] == 1) {
641
642                         environmentGrid[i][j] = 2;
643
644                         button.fire();
645
646                         environmentGrid[i][j] = 1;
647
648                     }
649
650                 }
651             }
652
653         }
654
655     }
656
657     @Override
658     public void handleEvent(Event event) {
659
660         if (event.getSource() == button) {
661
662             System.out.println("Button Clicked!");
663
664             for (int i = 0; i < environmentGrid.length;
665                 i++) {
666                 for (int j = 0; j < environmentGrid[0].length;
667                     j++) {
668
669                     if (environmentGrid[i][j] == 1) {
670
671                         environmentGrid[i][j] = 2;
672
673                         button.fire();
674
675                         environmentGrid[i][j] = 1;
676
677                     }
678
679                 }
680             }
681
682         }
683
684     }
685
686     @Override
687     public void handleEvent(Event event) {
688
689         if (event.getSource() == button) {
690
691             System.out.println("Button Clicked!");
692
693             for (int i = 0; i < environmentGrid.length;
694                 i++) {
695                 for (int j = 0; j < environmentGrid[0].length;
696                     j++) {
697
698                     if (environmentGrid[i][j] == 1) {
699
700                         environmentGrid[i][j] = 2;
701
702                         button.fire();
703
704                         environmentGrid[i][j] = 1;
705
706                     }
707
708                 }
709             }
710
711         }
712
713     }
714
715     @Override
716     public void handleEvent(Event event) {
717
718         if (event.getSource() == button) {
719
720             System.out.println("Button Clicked!");
721
722             for (int i = 0; i < environmentGrid.length;
723                 i++) {
724                 for (int j = 0; j < environmentGrid[0].length;
725                     j++) {
726
727                     if (environmentGrid[i][j] == 1) {
728
729                         environmentGrid[i][j] = 2;
730
731                         button.fire();
732
733                         environmentGrid[i][j] = 1;
734
735                     }
736
737                 }
738             }
739
740         }
741
742     }
743
744     @Override
745     public void handleEvent(Event event) {
746
747         if (event.getSource() == button) {
748
749             System.out.println("Button Clicked!");
750
751             for (int i = 0; i < environmentGrid.length;
752                 i++) {
753                 for (int j = 0; j < environmentGrid[0].length;
754                     j++) {
755
756                     if (environmentGrid[i][j] == 1) {
757
758                         environmentGrid[i][j] = 2;
759
760                         button.fire();
761
762                         environmentGrid[i][j] = 1;
763
764                     }
765
766                 }
767             }
768
769         }
770
771     }
772
773     @Override
774     public void handleEvent(Event event) {
775
776         if (event.getSource() == button) {
777
778             System.out.println("Button Clicked!");
779
780             for (int i = 0; i < environmentGrid.length;
781                 i++) {
782                 for (int j = 0; j < environmentGrid[0].length;
783                     j++) {
784
785                     if (environmentGrid[i][j] == 1) {
786
787                         environmentGrid[i][j] = 2;
788
789                         button.fire();
790
791                         environmentGrid[i][j] = 1;
792
793                     }
794
795                 }
796             }
797
798         }
799
800     }
801
802     @Override
803     public void handleEvent(Event event) {
804
805         if (event.getSource() == button) {
806
807             System.out.println("Button Clicked!");
808
809             for (int i = 0; i < environmentGrid.length;
810                 i++) {
811                 for (int j = 0; j < environmentGrid[0].length;
812                     j++) {
813
814                     if (environmentGrid[i][j] == 1) {
815
816                         environmentGrid[i][j] = 2;
817
818                         button.fire();
819
820                         environmentGrid[i][j] = 1;
821
822                     }
823
824                 }
825             }
826
827         }
828
829     }
830
831     @Override
832     public void handleEvent(Event event) {
833
834         if (event.getSource() == button) {
835
836             System.out.println("Button Clicked!");
837
838             for (int i = 0; i < environmentGrid.length;
839                 i++) {
840                 for (int j = 0; j < environmentGrid[0].length;
841                     j++) {
842
843                     if (environmentGrid[i][j] == 1) {
844
845                         environmentGrid[i][j] = 2;
846
847                         button.fire();
848
849                         environmentGrid[i][j] = 1;
850
851                     }
852
853                 }
854             }
855
856         }
857
858     }
859
860     @Override
861     public void handleEvent(Event event) {
862
863         if (event.getSource() == button) {
864
865             System.out.println("Button Clicked!");
866
867             for (int i = 0; i < environmentGrid.length;
868                 i++) {
869                 for (int j = 0; j < environmentGrid[0].length;
870                     j++) {
871
872                     if (environmentGrid[i][j] == 1) {
873
874                         environmentGrid[i][j] = 2;
875
876                         button.fire();
877
878                         environmentGrid[i][j] = 1;
879
880                     }
881
882                 }
883             }
884
885         }
886
887     }
888
889     @Override
890     public void handleEvent(Event event) {
891
892         if (event.getSource() == button) {
893
894             System.out.println("Button Clicked!");
895
896             for (int i = 0; i < environmentGrid.length;
897                 i++) {
898                 for (int j = 0; j < environmentGrid[0].length;
899                     j++) {
900
901                     if (environmentGrid[i][j] == 1) {
902
903                         environmentGrid[i][j] = 2;
904
905                         button.fire();
906
907                         environmentGrid[i][j] = 1;
908
909                     }
910
911                 }
912             }
913
914         }
915
916     }
917
918     @Override
919     public void handleEvent(Event event) {
920
921         if (event.getSource() == button) {
922
923             System.out.println("Button Clicked!");
924
925             for (int i = 0; i < environmentGrid.length;
926                 i++) {
927                 for (int j = 0; j < environmentGrid[0].length;
928                     j++) {
929
930                     if (environmentGrid[i][j] == 1) {
931
932                         environmentGrid[i][j] = 2;
933
934                         button.fire();
935
936                         environmentGrid[i][j] = 1;
937
938                     }
939
940                 }
941             }
942
943         }
944
945     }
946
947     @Override
948     public void handleEvent(Event event) {
949
950         if (event.getSource() == button) {
951
952             System.out.println("Button Clicked!");
953
954             for (int i = 0; i < environmentGrid.length;
955                 i++) {
956                 for (int j = 0; j < environmentGrid[0].length;
957                     j++) {
958
959                     if (environmentGrid[i][j] == 1) {
960
961                         environmentGrid[i][j] = 2;
962
963                         button.fire();
964
965                         environmentGrid[i][j] = 1;
966
967                     }
968
969                 }
970             }
971
972         }
973
974     }
975
976     @Override
977     public void handleEvent(Event event) {
978
979         if (event.getSource() == button) {
980
981             System.out.println("Button Clicked!");
982
983             for (int i = 0; i < environmentGrid.length;
984                 i++) {
985                 for (int j = 0; j < environmentGrid[0].length;
986                     j++) {
987
988                     if (environmentGrid[i][j] == 1) {
989
990                         environmentGrid[i][j] = 2;
991
992                         button.fire();
993
994                         environmentGrid[i][j] = 1;
995
996                     }
997
998                 }
999             }
1000
1001         }
1002
1003     }
1004
1005     @Override
1006     public void handleEvent(Event event) {
1007
1008         if (event.getSource() == button) {
1009
1010             System.out.println("Button Clicked!");
1011
1012             for (int i = 0; i < environmentGrid.length;
1013                 i++) {
1014                 for (int j = 0; j < environmentGrid[0].length;
1015                     j++) {
1016
1017                     if (environmentGrid[i][j] == 1) {
1018
1019                         environmentGrid[i][j] = 2;
1020
1021                         button.fire();
1022
1023                         environmentGrid[i][j] = 1;
1024
1025                     }
1026
1027                 }
1028             }
1029
1030         }
1031
1032     }
1033
1034     @Override
1035     public void handleEvent(Event event) {
1036
1037         if (event.getSource() == button) {
1038
1039             System.out.println("Button Clicked!");
1040
1041             for (int i = 0; i < environmentGrid.length;
1042                 i++) {
1043                 for (int j = 0; j < environmentGrid[0].length;
1044                     j++) {
1045
1046                     if (environmentGrid[i][j] == 1) {
1047
1048                         environmentGrid[i][j] = 2;
1049
1050                         button.fire();
1051
1052                         environmentGrid[i][j] = 1;
1053
1054                     }
1055
1056                 }
1057             }
1058
1059         }
1060
1061     }
1062
1063     @Override
1064     public void handleEvent(Event event) {
1065
1066         if (event.getSource() == button) {
1067
1068             System.out.println("Button Clicked!");
1069
1070             for (int i = 0; i < environmentGrid.length;
1071                 i++) {
1072                 for (int j = 0; j < environmentGrid[0].length;
1073                     j++) {
1074
1075                     if (environmentGrid[i][j] == 1) {
1076
1077                         environmentGrid[i][j] = 2;
1078
1079                         button.fire();
1080
1081                         environmentGrid[i][j] = 1;
1082
1083                     }
1084
1085                 }
1086             }
1087
1088         }
1089
1090     }
1091
1092     @Override
1093     public void handleEvent(Event event) {
1094
1095         if (event.getSource() == button) {
1096
1097             System.out.println("Button Clicked!");
1098
1099             for (int i = 0; i < environmentGrid.length;
1100                 i++) {
1101                 for (int j = 0; j < environmentGrid[0].length;
1102                     j++) {
1103
1104                     if (environmentGrid[i][j] == 1) {
1105
1106                         environmentGrid[i][j] = 2;
1107
1108                         button.fire();
1109
1110                         environmentGrid[i][j] = 1;
1111
1112                     }
1113
1114                 }
1115             }
1116
1117         }
1118
1119     }
1120
1121     @Override
1122     public void handleEvent(Event event) {
1123
1124         if (event.getSource() == button) {
1125
1126             System.out.println("Button Clicked!");
1127
1128             for (int i = 0; i < environmentGrid.length;
1129                 i++) {
1130                 for (int j = 0; j < environmentGrid[0].length;
1131                     j++) {
1132
1133                     if (environmentGrid[i][j] == 1) {
1134
1135                         environmentGrid[i][j] = 2;
1136
1137                         button.fire();
1138
1139                         environmentGrid[i][j] = 1;
1140
1141                     }
1142
1143                 }
1144             }
1145
1146         }
1147
1148     }
1149
1150     @Override
1151     public void handleEvent(Event event) {
1152
1153         if (event.getSource() == button) {
1154
1155             System.out.println("Button Clicked!");
1156
1157             for (int i = 0; i < environmentGrid.length;
1158                 i++) {
1159                 for (int j = 0; j < environmentGrid[0].length;
1160                     j++) {
1161
1162                     if (environmentGrid[i][j] == 1) {
1163
1164                         environmentGrid[i][j] = 2;
1165
1166                         button.fire();
1167
1168                         environmentGrid[i][j] = 1;
1169
1170                     }
1171
1172                 }
1173             }
1174
1175         }
1176
1177     }
1178
1179     @Override
1180     public void handleEvent(Event event) {
1181
1182         if (event.getSource() == button) {
1183
1184             System.out.println("Button Clicked!");
1185
1186             for (int i = 0; i < environmentGrid.length;
1187                 i++) {
1188                 for (int j = 0; j < environmentGrid[0].length;
1189                     j++) {
1190
1191                     if (environmentGrid[i][j] == 1) {
1192
1193                         environmentGrid[i][j] = 2;
1194
1195                         button.fire();
1196
1197                         environmentGrid[i][j] = 1;
1198
1199                     }
1200
1201                 }
1202             }
1203
1204         }
1205
1206     }
1207
1208     @Override
1209     public void handleEvent(Event event) {
1210
1211         if (event.getSource() == button) {
1212
1213             System.out.println("Button Clicked!");
1214
1215             for (int i = 0; i < environmentGrid.length;
1216                 i++) {
1217                 for (int j = 0; j < environmentGrid[0].length;
1218                     j++) {
1219
1220                     if (environmentGrid[i][j] == 1) {
1221
1222                         environmentGrid[i][j] = 2;
1223
1224                         button.fire();
1225
1226                         environmentGrid[i][j] = 1;
1227
1228                     }
1229
1230                 }
1231             }
1232
1233         }
1234
1235     }
1236
1237     @Override
1238     public void handleEvent(Event event) {
1239
1240         if (event.getSource() == button) {
1241
1242             System.out.println("Button Clicked!");
1243
1244             for (int i = 0; i < environmentGrid.length;
1245                 i++) {
1246                 for (int j = 0; j < environmentGrid[0].length;
1247                     j++) {
1248
1249                     if (environmentGrid[i][j] == 1) {
1250
1251                         environmentGrid[i][j] = 2;
1252
1253                         button.fire();
1254
1255                         environmentGrid[i][j] = 1;
1256
1257                     }
1258
1259                 }
1260             }
1261
1262         }
1263
1264     }
1265
1266     @Override
1267     public void handleEvent(Event event) {
1268
1269         if (event.getSource() == button) {
1270
1271             System.out.println("Button Clicked!");
1272
1273             for (int i = 0; i < environmentGrid.length;
1274                 i++) {
1275                 for (int j = 0; j < environmentGrid[0].length;
1276                     j++) {
1277
1278                     if (environmentGrid[i][j] == 1) {
1279
1280                         environmentGrid[i][j] = 2;
1281
1282                         button.fire();
1283
1284                         environmentGrid[i][j] = 1;
1285
1286                     }
1287
1288                 }
1289             }
1290
1291         }
1292
1293     }
1294
1295     @Override
1296     public void handleEvent(Event event) {
1297
1298         if (event.getSource() == button) {
1299
1300             System.out.println("Button Clicked!");
1301
1302             for (int i = 0; i < environmentGrid.length;
1303                 i++) {
1304                 for (int j = 0; j < environmentGrid[0].length;
1305                     j++) {
1306
1307                     if (environmentGrid[i][j] == 1) {
1308
1309                         environmentGrid[i][j] = 2;
1310
1311                         button.fire();
1312
1313                         environmentGrid[i][j] = 1;
1314
1315                     }
1316
1317                 }
1318             }
1319
1320         }
1321
1322     }
1323
1324     @Override
1325     public void handleEvent(Event event) {
1326
1327         if (event.getSource() == button) {
1328
1329             System.out.println("Button Clicked!");
1330
1331             for (int i = 0; i < environmentGrid.length;
1332                 i++) {
1333                 for (int j = 0; j < environmentGrid[0].length;
1334                     j++) {
1335
1336                     if (environmentGrid[i][j] == 1) {
1337
1338                         environmentGrid[i][j] = 2;
1339
1340                         button.fire();
1341
1342                         environmentGrid[i][j] = 1;
1343
1344                     }
1345
1346                 }
1347             }
1348
1349         }
1350
1351     }
1352
1353     @Override
1354     public void handleEvent(Event event) {
1355
1356         if (event.getSource() == button) {
1357
1358             System.out.println("Button Clicked!");
1359
1360             for (int i = 0; i < environmentGrid.length;
1361                 i++) {
1362                 for (int j = 0; j < environmentGrid[0].length;
1363                     j++) {
1364
1365                     if (environmentGrid[i][j] == 1) {
1366
1367                         environmentGrid[i][j] = 2;
1368
1369                         button.fire();
1370
1371                         environmentGrid[i][j] = 1;
1372
1373                     }
1374
1375                 }
1376             }
1377
1378         }
1379
1380     }
1381
1382     @Override
1383     public void handleEvent(Event event) {
1384
1385         if (event.getSource() == button) {
1386
1387             System.out.println("Button Clicked!");
1388
1389             for (int i = 0; i < environmentGrid.length;
1390                 i++) {
1391                 for (int j = 0; j < environmentGrid[0].length;
1392                     j++) {
1393
1394                     if (environmentGrid[i][j] == 1) {
1395
1396                         environmentGrid[i][j] = 2;
1397
1398                         button.fire();
1399
1400                         environmentGrid[i][j] = 1;
1401
1402                     }
1403
1404                 }
1405             }
1406
1407         }
1408
1409     }
1410
1411     @Override
1412     public void handleEvent(Event event) {
1413
1414         if (event.getSource() == button) {
1415
1416             System.out.println("Button Clicked!");
1417
1418             for (int i = 0; i < environmentGrid.length;
1419                 i++) {
1420                 for (int j = 0; j < environmentGrid[0].length;
1421                     j++) {
1422
1423                     if (environmentGrid[i][j] == 1) {
1424
1425                         environmentGrid[i][j] = 2;
1426
1427                         button.fire();
1428
1429                         environmentGrid[i][j] = 1;
1430
1431                     }
1432
1433                 }
1434             }
1435
1436         }
1437
1438     }
1
```

```

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
121          //too close to each other
122          while(tooClose){
123              tooClose = false;
124              for (int r = x-10; r <= x+10 && r
125                  < gameGrid.length && r > 0; r++) {
126                  for (int c = y - 10; c <= y +
127                      10 && c < gameGrid[0].length && c > 0; c++) {
128                      if(environmentGrid[r][c]
129                          ] == 3){
130                          tooClose = true;
131                      }
132                  }
133              }
134          }
135
136          //coloring the blocks in
137          for (int j = 0; j < (int)(Math.random()
138              ()*4)+3; j++) {
139                  environmentGrid[x-1][y+j] = 3;
140                  environmentGrid[x][y+j] = 3;
141                  environmentGrid[x+1][y+j] = 3;
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 override  
        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:#ffccbc");  
212                 }  
213             }  
214         }  
215     }  
216 }
```

```
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()*(  

242             environmentGrid[0].length));  

243             allTigers.add(new Tiger(x,y, 3));  

244             //setting the game grid to a certain  

245             number depending on the age and gender of the  

246             tiger  

247             if(allTigers.get(allTigers.size()-1).  

248                 getAge() < 3){  

249                     gameGrid[allTigers.get(allTigers.size()  

250                     ()-1).getX()][allTigers.get(allTigers.size()-1).  

251                     getY()] = 1;  

252             } else {  

253  

254             if(allTigers.get(allTigers.size()-1).  

255                 getGender() == 0){  

256                     gameGrid[allTigers.get(allTigers.  

257                     size()-1).getX()][allTigers.get(allTigers.size()-1)  

258                     .getY()] = 2;  

259             } else {  

260                     gameGrid[allTigers.get(allTigers.  

261                     size()-1).getX()][allTigers.get(allTigers.size()-1)  

262                     .getY()] = 3;  

263             }  

264             updateScreen();  

265         }
266
267         @FXML
268         private void addFlock(ActionEvent event){
269             allFlocks.add(new ArrayList<>());
270             int x = (int)(Math.random()*(  

271                 environmentGrid.length - 12)) + 6;
272             int y = (int)(Math.random()*(  

273                 environmentGrid[0].length - 12)) + 6;
274
275             //creates the leader of the flock and then
276             //every other deer in the flock will be surrounded
277             //around the leader
278             allFlocks.get(allFlocks.size()-1).add(new
279             Deer(x, y, 2, allFlocks.size()-1));

```

```

266         allFlocks.get(allFlocks.size()-1).get(0).
267             setGender(1);
268         for (int j = 0; j < (int)(Math.random()*3
269             ) + 2; j++) {
270             int r = (int)(Math.random()*7) + (x-3
271             );
272             int c = (int)(Math.random()*7) + (y-3
273             );
274             allFlocks.get(allFlocks.size()-1).add(
275                 new Deer(r, c, 2, allFlocks.size()-1));
276         }
277
278         for(int i = 0; i < allFlocks.get(allFlocks
279             .size()-1).size(); i++){
280             gameGrid[allFlocks.get(allFlocks.size
281             ()-1).get(i).getX()][allFlocks.get(allFlocks.size
282             ()-1).get(i).getY()] = allFlocks.get(allFlocks.
283             size()-1).get(i).getGender() + 5;
284         }
285
286         allDeer.addAll(allFlocks.get(allFlocks.
287             size() - 1));
288         System.out.println(allFlocks.get(allFlocks
289             .size() - 1).size());
290         updateScreen();
291
292     @FXML
293     private void handleMating(){
294         if(matingCheck.isSelected()){
295             matingSeason = true;
296         } else {
297             matingSeason = false;
298         }
299     }
300
301     public void start() {
302         lChart.getData().add(deerPop);
303         lChart.getData().add(tigerPop);
304         new AnimationTimer() {

```

```

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

```

```

322                               allTigers.get(i).
323                     foundPrey(allDeer, gameGrid, allFlocks, allTigers
324                     , allDeer.get(j));
325                         }
326                         }
327                         if(Math.random() <= 0.2){
328                             windy();
329                         }
330                         if (now - allTigers.get(i
331 ).getStartTime() > 1000000000.0) {
332                             if(allTigers.get(i).
333                               checkingOpposite(allTigers) && allTigers.get(i).
334                               getGender()==0 && (!allTigers.get(i).inWater(
335                                 environmentGrid))){
336                                 System.out.println
337                                 ("female tiger found a male tiger");
338                                 for (int j = 0; j
339                                   < (int)((Math.random()*3) + 1); j++) {
340                                     System.out.
341                                     println("female reproduced");
342                                     reproduceTiger
343                                     (i);
344                                     }
345                               }
346                               if(allTigers.get(i).
347                                 isHunting())){
348                                   allTigers.get(i).
349                                   chasingPrey(gameGrid, allDeer);
350                               } else if(matingSeason
351                                 && allTigers.get(i).getThirst() <= 15 &&
352                                 allTigers.get(i).getAge() < 3 && !allTigers.get(i
353                                 ).isHasBaby()) {
354                                     //running through
355                                     all tigers array and checking which one is close,
356                                     diff gender and doesn't already have a target mate
357                                     for (int j = 0; j
358                                     < allTigers.size() && allTigers.get(i).

```

```

344 targetType == null; j++) {
345                                     if(allTigers.
346                                         get(i).checkDistance(allTigers.get(j), 10) && (
347                                         allTigers.get(j).getGender() != allTigers.get(i).
348                                         getGender()) && allTigers.get(j).getTargetMate
349                                         () == null && allTigers.get(j).getAge() < 3 && !
350                                         allTigers.get(j).isHasBaby())){
351                                         allTigers.
352                                         get(i).setTargetMate(allTigers.get(j));
353                                         allTigers.
354                                         get(j).setTargetMate(allTigers.get(i));
355                                         }
356                                         }
357                                         }
358                                         }
359                                         //if the tiger
360                                         consumes food or water the stats are reset to 0.
361                                         if(allTigers.get(i).
362                                         getHunger() >= 15){
363                                         allTigers.get(i).
364                                         findPrey(allDeer, wind);
365                                         System.out.println
366                                         ("tiger found prey: " + allTigers.get(i).isHunting
367                                         ());
368                                         }
369                                         }
370                                         if(allTigers.get(i).
371                                         inWater(environmentGrid)){
372                                         if(salinityLvl >=
373                                         10){
374                                         allTigers.get(

```

```

367 i).setThirst(salinityLvl);
368                                         }else{
369                                         allTigers.get(
370                                         i).setThirst(0);
371                                         }
372                                         System.out.println
373                                         ("tiger found water");
374                                         }
375                                         allTigers.get(i).
376                                         increaseMonths();
377                                         allTigers.get(i).
378                                         setHunger(allTigers.get(i).getHunger()+1);
379                                         allTigers.get(i).
380                                         resetStartTime();
381                                         }

382                                         //there is a
383                                         //randomized probability of the thirst increasing in
384                                         //order to contrast the need of water for deers and
385                                         //tigers
386                                         if(Math.random()>.3
387                                         && allTigers.get(i).getAge() >= 3){
388                                         allTigers.get(i).
389                                         setThirst(allTigers.get(i).getThirst()+1);
390                                         }

391                                         }

392                                         //if the target mates
393                                         //are over 20 blocks away from each other than they
394                                         //will not be target mates no more
395                                         if(allTigers.get(i).
396                                         getTargetMate() != null){
397                                         if(!allTigers.get(
398                                         i).checkDistance(allTigers.get(i).getTargetMate
399                                         (), 20)){
400                                         allTigers.get(
401                                         i).getTargetMate().setTargetMate(null);
402                                         allTigers.get(
403                                         i).setTargetMate(null);
404                                         }
405                                         }
406                                         }

407                                         }

```

```

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

```

```

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

```

```

491         for (int c = allTigers.get(which).getY()
492             ()-1; c <= allTigers.get(which).getY()+1 && c <
493             gameGrid[0].length && c > 0; c++) {
494             if(checkEmptyAround(r, c) && !(c
495               environmentGrid[r][c] == 1)){
496               tempLocs.add(new Locations(r,
497               c));
498             }
499           }
500         }
501         if(tempLocs.size()>0 && allTigers.get(
502           which).getAge() >= 3 && Math.random() <= prob){
503           int num = (int)(Math.random() *
504             tempLocs.size());
505           allTigers.add(new Tiger(tempLocs.get(
506             num).getX(), tempLocs.get(num).getY(), 0));
507           gameGrid[tempLocs.get(num).getX()][
508             tempLocs.get(num).getY()] = 1;
509           allTigers.get(allTigers.size()-1).
510             setMother(allTigers.get(which));
511           System.out.println("new baby tiger");
512         } else {
513           System.out.println("no space for baby
514             tiger :(");
515         }
516       }
517     }
518     //this code is running through every
519     location around the ant.
520     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 );
557     System.out.println(rate);
558     count2++;
559     if(count2 == 13-rate && rate > 0){
560         salinityLvl += 1;
561         System.out.println("salinity: " +
562 salinityLvl);
563         riverWidth++;
564         setRiver(riverWidth);
565         count2 = 0;
566     }
567
568     public void windy(){
569         int chance = (int)(Math.random() * 4);
570         if(chance == 2){
571             wind[0] = (int)(Math.random() * 3) + 1
572 ;
573         } else if (chance == 3){
574             wind[0] = -1 * (int)(Math.random() * 3
575 ) + 1;
576         }
577         chance = (int)(Math.random() * 4);
578         if(chance == 2){
579             wind[1] = (int)(Math.random() * 3) + 1
580 ;
581         } else if (chance == 3){
582             wind[1] = -1 * (int)(Math.random() * 3
583 ) + 1;
584         }
585     }
586 }
```

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

```
1 package com.example.demo;
2
3 import javafx.application.Application;
4 import javafx.fxml.FXMLLoader;
5 import javafx.scene.Scene;
6 import javafx.stage.Stage;
7
8 import java.io.IOException;
9
10 public class HelloApplication extends Application {
11     @Override
12     public void start(Stage stage) throws
13         IOException {
14         FXMLLoader fxmlLoader = new FXMLLoader(
15             HelloApplication.class.getResource("hello-view.fxml"
16         ));
17         Scene scene = new Scene(fxmlLoader.load(),
18             600, 600);
19         stage.setTitle("Hello!");
20         stage.setScene(scene);
21         stage.show();
22     }
23 }
```