

```
1  /**
2   * Now define Anteater Class
3   */
4  class Anteater extends Organism
5  {
6      public static final int ANTEATER_BREED = 8;
7      public static final int ANTEATER_STARVE = 3;
8
9      // Number of ticks since eating.
10     private int starveTicks = 0;
11
12     /**
13      * Default constructor.
14      */
15     public Anteater()
16     {
17         super();
18     }
19
20     /**
21      * 3-parameter constructor.
22      *
23      * @param world The World object this Anteater lives in.
24      * @param x The x coordinate of the spot in <b>world</b> this Anteater inhabits.
25      * @param y The y coordinate of the spot in <b>world</b> this Anteater inhabits.
26      */
27     public Anteater(World world, int x, int y)
28     {
29         super(world, x, y);
30     }
31
32     // Basic code reused from Ant.breed() method with changes to create an Anteater object instead
33     // of an Ant object.
34
35     /**
36      * Adjusts the breed counter for the Anteater object and creates a new one when the appropriate
37      * condition is met.
38      */
39     public void breed()
40     {
41         breedTicks++;
42         if (breedTicks == ANTEATER_BREED)
43         {
44             breedTicks = 0;
45             // Try to create a new Anteater object. Because world reference is passed in and
46             // Anteater object adds itself to that world, Anteater reference doesn't need to be
47             // explicitly saved.
48             if ((y > 0) && (world.getAt(x, y - 1) == null))
49             {
50                 new Anteater(world, x, y - 1);
51             }
52             else if ((y < World.WORLDSIZE - 1) && (world.getAt(x, y + 1) == null))
53             {
54                 new Anteater(world, x, y + 1);
55             }
56             else if ((x > 0) && (world.getAt(x - 1, y) == null))
57             {
58                 new Anteater(world, x - 1, y);
59             }
60             else if ((x < World.WORLDSIZE - 1) && (world.getAt(x + 1, y) == null))
```

```
61         {
62             new Anteater(world, x + 1, y);
63         }
64     }
65 }
66
67 // Basic code reused from Ant.move() method, with optimizations for move checking and additions
68 // to handle starvation.
69
70 /**
71  * Moves an Anteater object around and handles the starvation counter.
72  */
73 public void move()
74 {
75     starveTicks++;
76     int direction = (int) (Math.random() * 4);
77
78     // up
79     if (direction == 0)
80     {
81         if ((y > 0) && !(world.getAt(x, y - 1) instanceof Anteater))
82         {
83             // Reset starvation counter if Anteater "ate".
84             if (world.getAt(x, y - 1) instanceof Ant)
85                 starveTicks = 0;
86
87             // Move to new spot.
88             world.setAt(x, y - 1, world.getAt(x, y));
89             world.setAt(x, y, null);
90             y--;
91         }
92     }
93     // down
94     else if (direction == 1)
95     {
96         if ((y < World.WORLDSIZE - 1) && !(world.getAt(x, y + 1) instanceof Anteater))
97         {
98             // Reset starvation counter if Anteater "ate".
99             if (world.getAt(x, y + 1) instanceof Ant)
100                 starveTicks = 0;
101
102             // Move to new spot.
103             world.setAt(x, y + 1, world.getAt(x, y));
104             world.setAt(x, y, null);
105             y++;
106         }
107     }
108     // left
109     else if (direction == 2)
110     {
111         if ((x > 0) && !(world.getAt(x - 1, y) instanceof Anteater))
112         {
113             // Reset starvation counter if Anteater "ate".
114             if (world.getAt(x - 1, y) instanceof Ant)
115                 starveTicks = 0;
116
117             // Move to new spot.
118             world.setAt(x - 1, y, world.getAt(x, y));
119             world.setAt(x, y, null);
120             x--;
```

```
121     }
122   }
123   // right
124   else
125   {
126     if ((x < World.WORLDSIZE - 1) && !(world.getAt(x + 1, y) instanceof Anteater))
127     {
128       // Reset starvation counter if Anteater "ate".
129       if (world.getAt(x + 1, y) instanceof Ant)
130         starveTicks = 0;
131
132       // Move to new spot.
133       world.setAt(x + 1, y, world.getAt(x, y));
134       world.setAt(x, y, null);
135       x++;
136     }
137   }
138 }
139
140 /**
141  * Checks to see if the anteater is starving.
142  *
143  * @return boolean, indicating whether the anteater is starving (true) or not (false).
144  */
145 public boolean starve()
146 {
147     return (starveTicks == ANTEATER_STARVE ? true : false);
148 }
149
150 /**
151  * Returns "X" as the printable character for an Anteater object.
152  */
153 public String getPrintableChar()
154 {
155     return "X";
156 }
157 } // Anteater
158
```