```
/**
* Now define Anteater Class
*/
class Anteater extends Organism
{
    public static final int ANTEATER_BREED  = 8;
    public static final int ANTEATER_STARVE = 3;

    // Number of ticks since eating.
    private int starveTicks = 0;

    /**
     * Default constructor.
     */
    public Anteater()
    {
        super();
    }

    /**
     * 3-parameter constructor.
     *
     * @param world The World object this Anteater lives in.
     * @param x The x coordinate of the spot in <b>world</b> this Anteater inhabits.
     * @param y The y coordinate of the spot in <b>world</b> this Anteater inhabits.
     */
    public Anteater(World world, int x, int y)
    {
        super(world, x, y);
    }

    // Basic code reused from Ant.breed() method with changes to create an Anteater object instead
    // of an Ant object.

    /**
     * Adjusts the breed counter for the Anteater object and creates a new one when the appropriate
     * condition is met.
     */
    public void breed()
    {
        breedTicks++;
        if (breedTicks == ANTEATER_BREED)
        {
            breedTicks = 0;
            // Try to create a new Anteater object. Because world reference is passed in and
            // Anteater object adds itself to that world, Anteater reference doesn't need to be
            // explicitly saved.
            if ((y > 0) && (world.getAt(x, y - 1) == null))
            {
                new Anteater(world, x, y - 1);
            }
            else if ((y < World.WORLDSIZE - 1) && (world.getAt(x, y + 1) == null))
            {
                new Anteater(world, x, y + 1);
            }
            else if ((x > 0) && (world.getAt(x - 1, y) == null))
            {
                new Anteater(world, x - 1, y);
            }
            else if ((x < World.WORLDSIZE - 1) && (world.getAt(x + 1, y) == null))
```

```java
                {
                    new Anteater(world, x + 1, y);
                }
            }
        }

        // Basic code reused from Ant.move() method, with optimizations for move checking and additions
        // to handle starvation.

        /**
         * Moves an Anteater object around and handles the starvation counter.
         */
        public void move()
        {
            starveTicks++;
            int direction = (int) (Math.random() * 4);

            // up
            if (direction == 0)
            {
                if ((y > 0) && !(world.getAt(x, y - 1) instanceof Anteater))
                {
                    // Reset starvation counter if Anteater "ate".
                    if (world.getAt(x, y - 1) instanceof Ant)
                        starveTicks = 0;

                    // Move to new spot.
                    world.setAt(x, y - 1, world.getAt(x, y));
                    world.setAt(x, y, null);
                    y--;
                }
            }
            // down
            else if (direction == 1)
            {
                if ((y < World.WORLDSIZE - 1) && !(world.getAt(x, y + 1) instanceof Anteater))
                {
                    // Reset starvation counter if Anteater "ate".
                    if (world.getAt(x, y + 1) instanceof Ant)
                        starveTicks = 0;

                    // Move to new spot.
                    world.setAt(x, y + 1, world.getAt(x, y));
                    world.setAt(x, y, null);
                    y++;
                }
            }
            // left
            else if (direction == 2)
            {
                if ((x > 0) && !(world.getAt(x - 1, y) instanceof Anteater))
                {
                    // Reset starvation counter if Anteater "ate".
                    if (world.getAt(x - 1, y) instanceof Ant)
                        starveTicks = 0;

                    // Move to new spot.
                    world.setAt(x - 1, y, world.getAt(x, y));
                    world.setAt(x, y, null);
                    x--;
                }
            }
            // right
            else
            {
                if ((x < World.WORLDSIZE - 1) && !(world.getAt(x + 1, y) instanceof Anteater))
                {
                    // Reset starvation counter if Anteater "ate".
                    if (world.getAt(x + 1, y) instanceof Ant)
                        starveTicks = 0;

                    // Move to new spot.
                    world.setAt(x + 1, y, world.getAt(x, y));
                    world.setAt(x, y, null);
                    x++;
                }
            }
        }

        /**
         * Checks to see if the anteater is starving.
         *
         * @return boolean, indicating whether the anteater is starving (true) or not (false).
         */
        public boolean starve()
        {
            return (starveTicks == ANTEATER_STARVE ? true : false);
        }

        /**
         * Returns "X" as the printable character for an Anteater object.
         */
        public String getPrintableChar()
        {
            return "X";
        }
    } // Anteater

```