```
package AntPheromones;

/**
Food.java

A simple class to make food for ants to carry back to the
nest.
At first we thought about having a large number of food
objects in three piles,
but then decided that having three food objects with a
magnitude attribute would
be more effective.

*/

import java.awt.Color;
// import java.awt.Point;
// import java.util.Vector;
// import java.util.ArrayList;
 import java.awt.BasicStroke;

import uchicago.src.sim.gui.Drawable;
import uchicago.src.sim.gui.SimGraphics;
// import uchicago.src.sim.space.Diffuse2D;
// import uchicago.src.sim.gui.ColorMap;

public class Food implements ObjectInGrid, Drawable {
// class variables, should be the same for all objects
    public  static int              nextId = 0; // to give
each an id
    public  static TorusWorld       world;      // where the
agents live
```

```
    public  static Model            model;      // the model
"in charge"
    public  static GUIModel         guiModel = null;   //
the gui model "in charge"

    public  static BasicStroke      foodEdgeStroke = new
BasicStroke( 1.0f );
// instance variable
    public int          id;         // unique id number for
each food instance
    public int          x, y;       // cache the food pile
x,y location
    public int          size;       // "size" of food - how
much is there
    public Color        myColor;    // color of this agent


// an Food constructor
// note it assigns ID values in sequence as foods are
created.
// blatantly stolen from Rick's ants
    public Food ( ) {
        id = nextId++;
        x = 0;      y = 0;
        size = 50;  // no idea what to put here
        setInitialColor();   // no idea what to do here
either
    }

    public void setInitialColor () {  // set agents initial
color
        myColor = Color.white;
```

```
    }
    // from the ant class - we'll see if it works
    // note these are class (static) methods, to set class
(static) variables
    public static void setWorld( TorusWorld w ) {   world =
w; }
    public static void setModel( Model m ) { model = m; }
    public static void resetNextId() { nextId = 0; }  //
call when we reset the model



    ////////////////////////////////////////////////////////////
    // setters and getters
    //
    public int getId() {  return id; }
    public int getX() { return x; }
    public void setX( int i ) { x = i; }
    public int getY() { return y; }
    public void setY( int i ) { y = i; }


/**
// draw
// we implement Drawable interface, so we need this method
// so that the food can draw itself when requested  (by the
GUI display).
*/
public void draw( SimGraphics g ) {
    g.drawFastRoundRect( myColor );
}
```

```
}
```