an agent moves → Agent method
an agent eats sugar → "
sugar grows back → cell method
an agent dies → Agent method
A good factorization
of your program

1. easier to maintain & test
2. makes it more flexible & reusable
private

Landscape

Cell

private int sugar
private int capacity
private int growthRate
private Vector<Agent> agents

public int getSugar
public void setSugar(int sugar)

setAgent
getAgent

setFood
getFood
Agent moves to cell eats all sugar

```java
Agent

Cell c = getMycell();

(c.eat(c.getSugar())

(c.sugar = 0;

public void eat()

public void eat(int count)
```
Does a cell know its own coordinates?

No, it can ask the landscape.

How do the cell know what cells are next to it?

Ask the landscape.
What about movement?
The algorithm for movement belongs in the Agent class!
Movement:

public void move() {
    // scan row to find cell w/max score
    // scan col
    // choose max of those two
    // move there
}
Rewrite method

```java
public void move() {
    // find 'best' cell
    // move to it
}
```
public void move() { 
    Cell nc = find NextCell(); 
    getCell().remove(this); 
    nc.add(this); 
}
private Cell findNewCell() {
    Cell c = getCell();
    for (i = 0; i < vision; i++) {
        // get sum for cell (x+i, y)
        if (sum > maxUpdate) {
            for (i = 0; i < vision; i++) {
                // get sum for cell (x+i, y)
                if (sum > 0) {
    //...
How to parameterize out the difference in directions?

```java
private Cell findNewCell() {
    Cell c = getCell();
    for (int i = 0; i < vision.jtt; i++) {
        if (cell @ x+i,y has more user) 
            if (cell @ x-i,y )
    }
```
private Cell scan(int dx, int dy)
{
    for (int i = 0; i < vision; i++)
    {
        // check cell @ x + idx, y + idy...
    }
}

move()
{
    scan(1, 0);
    scan(-1, 0);
    scan(0, 1);
    scan(0, -1);
}