Project: Ants versus Bees

Due Monday, May 16 at midnight.

Note: you may work with a partner on this lab. If you are going to work with a partner, it should be the same partner you worked with for the lab because this project involves using code from the lab. If you work with a partner, first make sure both names are on the lab. Only one of you needs to turn the lab in via Sakai. If one of you bails, you are still responsible for turning in the project on time.

In this lab you will be implementing a game called **Ants versus Bees**. In this game, the ants are placed on the board by you and they either harvest food for the colony of ants, or they protect the queen ants by attacking bees. The bees leave their hive and enter the playing board, where they advance, stinging ants as they move toward the queens. The board is a matrix (start with a 5x5 matrix). The game ends when either all the bees have left the hive and there are no more bees on the board (in which case the ants won!) or when all the queens have been stung (the ants lose) or when the ant colony runs out of food and there are no ants on the board(again, the ants have lost).

Some specifics: Each turn the bees leave the hive in amounts of 1-5 bees (a random number between 1 and 5 including 5). The hive is placed in a raondom square in the right-most column of the colony matrix. Bees can leave either side of the hive (going up one square or down one square) when they leave the hive's square, unless, of course, the hive is in the 0^{th} square of the last column, in which case bees can only exit down one square, or if the hive is at the bottom square in the last column, in which case bees can only exit up one square. More than one bee can be in a square. Note that when bees leave the hive, they don't move forward, they just move up or down a square.

Each turn every bee that's on the board either advances to a square into the next left column (e.g. if the bee is in column 3, it will attempt to advance to column 2). It can either move diagonally up a square, straight ahead, or diagonally down ahead, assuming the bee isn't already in row 0 or in the last row of the colony matrix. If there is an ant in the square the bee was going to move to, the bee stings the ant, and the ant loses armor (1 armor point) (Aside: if the ant is a ThrowerAnt, it will equally attack the bee and, based on the Ant's damage field, the bee will lose armor points equal to the ant's damage. If the bee's armor is decreased to 0, the bee is removed from the colony matrix.) If there is more than one bee in a square, it moves forward independently of the other bees in the square.

Each turn the user places ants on the board. Unlike the bees, there can only be one ant per square. The ants don't move. They just either add food or attack bees. So far you have two kinds of ants: the Thrower Ant and the Harvester Ant. Harvester Ants cost one point to put on the board, but every turn they remain on the board, they add 1 food point to the colony's food supply. Harvester ants also have very weak armor – they only have 1 armor point, so if a bee attacks, they die and should be removed from the board. The thrower ants cost 2 food points to put on the board. They have slightly stronger armor – their armor is 2. And when they attack, they inflict a damage of 2 on the bee they are attacking (the bee that was trying to move into their square).

Finally, the queen ants are located to the leftmost side of the colony – I made a separate array for my queens, that is the same length as the colony matrix's number of rows. Queens are weak – when they are stung, they die immediately, and they never attack. They just sit around all day being queenly. Why the rest of the ants are so interested in defending the queens, I don't know. But they are. You don't

want all your queens to be stung. So, for instance, if you have a bee in row 3 column 0, and in its next turn it wants to move to row 2 column -1, the queen at index 2 would be killed (and that bee leaves the board).

That's pretty much it. You should use the **Insect/Ant/Bee/ThrowerAnt/HarvesterAnt** hierarchy of classes you created in lab. (Although I don't mind if you want to modify them like, for instance, if you think it would be easier to make the Insect class be an abstract class or an interface). You will most likely also need to add methods (and fields) to the various classes/subclasses that wouldn't have made sense outside of the context of the game.

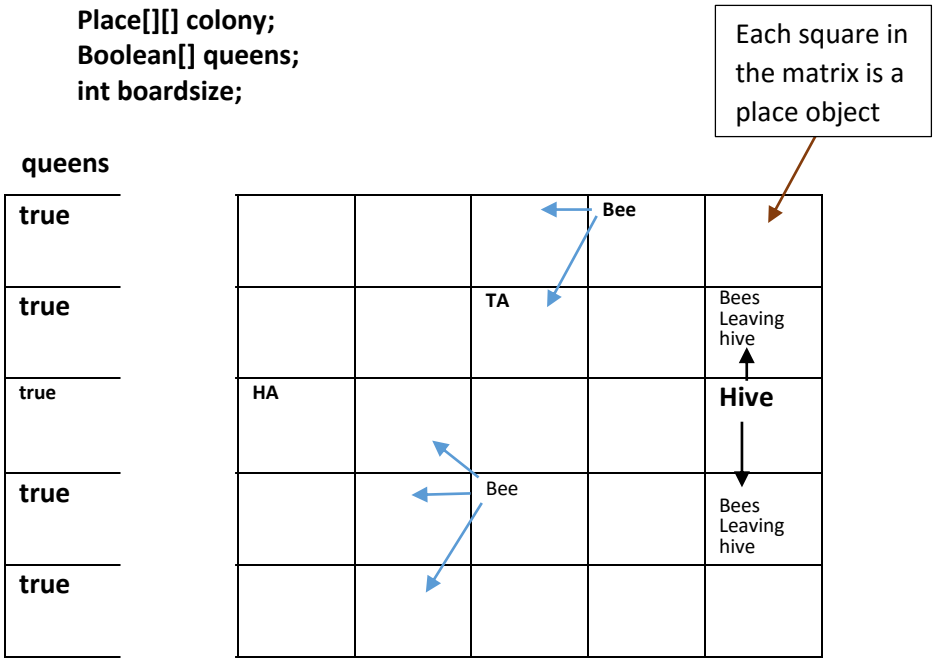
Other than following these basic rules and using the hierarchical structure created in lab, you have freedom to implement this any way you like.

Hints:

Here's a general idea of what the colony should look like:

Colony:

int Numbees;
Place hive;
int food = 2;
Place[][] colony;
Boolean[] queens;
int boardsize;



I made a place class. My place class consisted of a field for my x and y coordinates, an array of bees, an ant field, and then different methods for adding and removing the Ant and the bees from the array of bees.

I then made a colony class. The colony class consisted of a matrix of place objects, with the ants and the bees all originally set to null. The colony class also had fields for the hive – a Place object with x and y coordinates. The colony class had a field for the original number of bees in the hive (I started with 50), a field for the original amount of food for the ant colony (2), a boolean array for the queens (since they

don't really need to be ant objects, they can just be alive or dead), and then an integer value for the size of the matrix (the matrix is square, e.g., 5x5, 6x6, 4x4, etc, so the size would be the size of an edge, or 5, 6, 4 respectively).

In the colony class were the methods for running the game. There was a method for running each turn. There was a method for checking to see if anyone won, and there was a method for playing until someone won. I further refined my methods by adding a checkqueens method, an addants method, a launchfromhive method (for bees leaving the hive), and a moveallbees method.

The methods for removing armor were written in the ant/bee classes, and the adding bees to the array of a place and removing bees from an array were in the place class definition. The place class also had a method for removing an ant from a place.

Extra Credit Opportunities:

There are many ways in which you could make this game more interesting. One is to add a timer (Java has timer utilities that you can include. And then have the bees launch and move using the timer, instead of after every turn. So if you move too slowly, the bees will attack and win because they're moving on their own!

You can also add all sorts of new ants: maybe a nastyAnt that doesn't block the bee from moving, but does 1 damage to its armor, maybe a SchwarzeneggerAnt that has really strong armor, but costs a lot to place on the board, maybe a SkunkAnt that, when attacked, throws out poison damaging all bees within a 2-square radius – I'm making up ants here, but you get the idea.

You could also add a few random killer bees that have different damage and armor than regular bees.

Again, feel free to be creative. The amount of effort put in will determine the amount of extra credit.

If you really want to get into extra credit, you can add graphics. The way this game was written should lend itself nicely to tie in with a graphical user interface. Feel free to make this game really cool and add graphics!