Assignment 2: Bee Farm

For this assignment, you will be building a class, creating objects, doing comparisons and doing some calculations. You need to remember what you have learned in class, lab, books and your assignments. Be sure to refer to them when you need to.

There are 2 parts to this assignment. In the first part, you are going to be given a problem and you will then need to create a structure and write algorithms to solve it. In the second part, you’ll be turning this algorithm into a java program.

So let’s get started!

**Part 1: Loving your Pet Bees!**

Your little sister has decided that she wants a pet. You love animals and want to further engender her love of animals, but, well, her last pet, goldie the goldfish, didn’t fare too well. So you went on the hunt for a different kind of pet that will fit the bill. Much to your joy, you have found the perfect pet for her… a Bee farm! They are easy to take care of and don’t die off too quickly. Now all that you have to do is build a caretaker program so that she can easily take care of her Bees and see how they thrive (or not!).

Here’s what you need for the Bee farm (*hint: these can be be passed/given to the class constructor if you want to challenge yourself*):

* **Queen’s name** – Every Bee colony has a queen, and every queen has a name. So you’ll need to ask the user what they want to name their queen Bee.
* **Colony name** – Every pet needs a name, right? So you’ll need to ask the user what the name of their colony is.
* **Caretaker name** – You’ll need to ask who the colony’s mom/dad is.
* **Starting size** – How many Bees is the colony starting out with? This can vary, so you better ask the user about that also.

These are the things that can be done with the Bee farm:

(*hint: Do not assume that these must be methods. Think about how they will be used and then decide)*

* **Feed** – Like all pets, Bees get hungry. In our case, though, only when the queen is going to breed. Fortunately, with the program you are building, you can ask the user how many days their colony should be fed and then check to make sure they have enough food for the queen to breed and have babies (they need 1 day of food for each time she breeds).
* **Breed** – Other than eating, queen Bees don’t do much else but breed and have babies. You’ll want to be sure she has some entertainment, so you’ll need to ask your user if they want to breed their queen Bee and, if so, how many times. For each time she *successfully* breeds and has babies, the colony triples in size. (*hint: re-read about feeding*). *Note:* *As you are not required to use loops, you may limit the number of times the queen breeds to 10 if you can’t figure out another way of doing this (also see lifespan below)*.
* **Harvest** – You love the bees, but especially love that they make honey. You’ll want to be sure that you harvest your honey. First, the bees only produce honey if they have been fed appropriately. For every ounce of honey, the bees need to have been fed for two days.
* **Lifespan** – Although we love our Bees, they don’t always live very long. In fact, the colony depends on the life of the queen. Queens can only live about 10days. So if the caretaker gives the colony food for more than 10 days, the queen will die. If that happens, then 50% of the Bees will die.
* **Expand Colony** – Additional hives can be added to the Bee farms, allowing the Bees to expand the colony. Ask your user if they want to add an additional hive.
  + If the colony IS expanded, there is a 70% chance that a new queen will be born.
  + If the colony IS NOT expanded, there is a 20% chance that a new queen will be born.

(*hint: you can use the random number generator from Assignment 1 to help you with this*)

* **New Queen** - If a new queen is born, the new queen’s name will be the name of the original queen with “2.0” added to the end of the name.

For this program, you’ll want to ask your user about any pertinent information up front. *Do not worry about having any loops to ask them things like “Do you want to feed your colony again?”* You will only ask them ONCE for the needed information and then tell them how their colony is doing.

For your output (nicely formatted in a JOptionPane), you will want to include:

* Caretaker Name
* Bee Colony Name
* Queen’s Name
* Starting Size
* How many days they were fed
* Requested number of times to breed
* How many times they *successfully* bred
* How much honey you could harvest
* Whether the queen got sick and how many died
* Whether the colony was expanded
* Whether a new queen was born and what her name is
* Final number of Bees in the colony

*For Part 1*, create a class structure and algorithms for your BeeFarm class, and then do several iterations of tests (i.e., analyze it and step through to make sure that it is logically correct). Also write the pseudocode for your tester class (where your main will go). Put these in a Word or Open Office document. You’ll turn that document in with the program that you create in Part 2.

**Important! As you are working on this, be sure to break this down into smaller pieces. Take it step-by-step, and don’t try to finish this in one sitting. It will make it MUST easier.**

**Part 2: Creating your Bee Farm program**

Once you are done writing and testing your class structure and algorithm, you are ready to start coding!

1. Once again, you first you need to create a project. Here’s a nice tutorial on how to do that in Netbeans. If you are using Dr. Java or Eclipse, just do a quick search on youtube.com and you’ll find lots of candidates.

<http://www.youtube.com/watch?v=ezUHG1cuxkM>

Be sure to give your project a *nice, meaningful name* (and make sure it adheres to Java’s naming conventions).

1. Once you have your shell ready, there are a few things to know before you start translating your algorithm into code
   * At the top of your class file, be sure to include the following:

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// FIU PANTHERID: [Your FIU PantherID]

// CLASS: COP 2210 – [Semester Year]

// ASSIGNMENT # [#]

// DATE: [Date]

//

// I hereby swear and affirm that this work is solely my own, and not the work

// or the derivative of the work of someone else.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* + You will need a method that generates a random number again. Here’s what you need to do to use it:
    1. Include the following code at the top of your class file (so that you can use this class:

import java.util.Random;

To find out more about this, go to <http://java.sun.com/javase/7/docs/api/index.html> (like you did in Lab Assignment 2)

* + 1. You’ll need to use some variables. Here’s how you get a random number:

Random r = new Random();

int x = 1 + r.nextInt(10);

Upper limit of the random number generated

Note that the number in the parens (e.g., 10 above) is the upper limit of the random number. So, the random number that you get here will be an integer between 0 and 10. Need a larger range? Just change the 10 to the top of your range.

Here’s another example, in this case if you are printing a random number to the console:

System.out.print( 1 + r.nextInt(5) + " " );

1. You’ll need to use an if-else statement. We haven’t covered that yet, so here’s the structure:

if( *variableName1* < *variableName2*){

// put in what is done if the value of

// *variableName* is less than the value of

// *variableName2*

}else{

// put in what is done if the value of

// *variableName* is greater than or equal to the

// value of *variableName2*

}

(*hint: think about what you need to do with Feed and Breed*)

1. Now start translating your algorithm into java code.
   * Remember to code and then compile frequently. It will make it easier to find any bugs.
   * Also remember that you will need to create a **tester class** (where your main method will reside).
2. Once you get your program running correctly, there is one more thing to do. *Any input requested from the user and/or output received from the user should be in a window* (see E.1.14 and E.1.15 from lab 1). At this point, you probably have your output going to the console. For your final submission, it needs to go to a window (JOptionPane). Don’t forget any additional libraries that you need to import to do this.

That’s it! *Now you can nurture your own amoeba colony, and, chances are, they’ll do just fine as your sister’s pet*! Of course, you’ll also need to turn it in to Moodle.

**Submission Requirements**

You must upload a zip file to Moodle that includes your complete source project in Netbeans, ready to load, and also contains the output in separate data files, and your Word/Open Office document with your algorithm.

**VERY IMPORTANT:** If you do not provide output in separate, easy to find data files, I will assume that your program does not work on those test cases, and grade accordingly. *Do not embed the output in your source code.*