Assignment 2: Amoeba Colony

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 Amoeba Colony!**

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… an amoeba colony! 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 amoebas and see how they thrive (or not!).

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

* **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 amoebas 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 colony:

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

* **Feed** – Like all pets, amoebas get hungry. In our case, though, only when they are 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 to breed (they need 1 day of food for each time they breed).
* **Breed** – Other than eating, amoebas don’t do much else but breed. You’ll want to be sure they have some entertainment, so you’ll need to ask your user if they want to breed their amoebas and, if so, how many times. For each time they *successfully* breed, the colony doubles in size. (*hint: re-read about feeding*)
* **Vitamins** – Vitamins can help our amoebas stay healthy. Ask your user if they want to give their colony vitamins.
* **Sickness** – Unfortunately, all pets sometimes get sick. When that happens, 10% of the amoebas die off. For every session:
  + If the colony is NOT given vitamins, it has a 25% chance that it may get sick.
  + If the colony IS given vitamins, it has a 20% chance that it may get sick.

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

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:

* Colony Name
* Caretaker Name
* Starting Size
* How many times they were fed
* Requested number of times to breed
* How many times they *successfully* bred
* Whether they got sick and how many died
* Final number of amoebas in the colony

*For Part 1*, create a class structure and algorithms for your AmoebaColony 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 Amoeba Colony 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:

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

// PantherID: [Your Name]

// 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.

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

* + Remember I told you that you would be given a method that generates a random number? 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:

Upper limit of the random number generated (exclusive)

Random r = new Random();

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

Note that the number in the parens (e.g., 10 above) is the upper limit of the random number, exclusively. So, the random number that you get here will be an integer between 1 and 10, once you add the 1. 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 **separate 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.*