Advanced Object Oriented Programming-2 (PROG36859)
Assignment 3
Due Date: 02:00 PM, Dec. 06, 2019 (Friday)

INSTRUCTIONS

- This assignment **MUST** be completed in a **group of 2 (two) students** without any outside collaboration. (Students can form their own groups.) Copying or reproducing the work done by others (in part or in full) or letting others to copy or reproduce your own, and/or unauthorized collaboration will be treated as academic dishonesty under the College's Academic Dishonesty Policy.
- This is a group assignment. Single member work will not be marked, unless you obtain prior approval from your Instructor.
- This is an out of class assignment and you are required to complete this assignment on your own time.
- Your application must compile and run upon download to receive any mark. Your supplied outputs (e.g., screenshots) may not be marked, if the corresponding program fails to compile and run.
- Late submissions will be subject to a reduction in the grade earned as follows: A penalty of 10% of the value of the assignment will be immediately deducted for late submissions. An additional 10% of the value of the assignment will be deducted for each subsequent day (includes weekends and holidays) to a maximum of 3 days at which point the assignment will be assigned a mark of "zero".
- To submit the assignments, please follow the Submission Guideline provided at the end of this assignment.
- Each group (**both members**) **MUST** demonstrate their work and simulation during the class/lab session on **Dec 6th, 2019**. Both members must be able to demonstrate the complete work. Demonstration is important and missing it may cost significant part (at least 40%) of this assignment marks.
- You must include following information as comments at the beginning of your main class file.
    - Assignment No.: 2
    - Member 1: <Full Name>, <ID>
    - Member 2: <Full Name>, <ID>
    - Submission date:
    - Instructor's name: Syed Tanbeer
- **Total mark = 20** (weight = 10% of the final grade).

Feed Zoo Animals

Table 1 Animal Food Chart

Animal	Required amount of food (kg)
Elephant	15
Giraffe	9
Horse	5
Zebra	5
Deer	3

Simulate a zoo animal feeding system in Java. Assume the system maintains a stock of food and takes care of feeding for five different animals, which are elephant, giraffe, horse, zebra, and deer. Each of these animals has different amount of food requirements (as per Table 1). For example, at least 15 kg food is required in the food stock to feed an elephant, whenever it gets hungry.

Feeding Task: Animals become hungry at a random order, but one animal at a time (i.e., if an animal gets hungry, then no other animal can be hungry until the hungry animal is fed its required amount of food.) If the required amount of food (for the hungry animal) is available in the food stock, the hungry animal is fed right away and the food stock is updated with this food consumption. If that amount of food is not available in the stock, the Feeding Task system must wait for the Depositing Task (described below) system that adds food to the food stock.

Depositing Task: Food is deposited to the food stock at a random amount between 1 kg and 20 kg, at a time.

Concurrency: Both tasks (Feeding Task and Depositing Task) run concurrently, but in a synchronized manner. When the Feeding Task consumes food from the stock, the food stock is locked to the Depositing Task. Similarly, during the time the Depositing Task adds food to the stock, the stock is locked to the Feeding Task. However, after each deposit, the Depositing Task should send signal to (all) waiting Feeding Task for possible food consumption.

Processing Feeding Data: Your simulation will keep track of number of animals that have been fed. It will terminate both tasks of Feeding and Depositing, after feeding *n* animals (*n* is the total number of animals to be fed by the system, and may be supplied by the user, or pre-defined). It will also keep track of the total amount of food fed to each specific animal.

Storing data in database: Once the feeding process is done, your simulation should connect to your MySQL database and create a table called FeedingData (animalName, feedingCount, hungryCount), and store the feeding records for all the animals into the table (create the database called ZooDB first through connecting to your MySQL manually or programmatically). The animalName field should store the name of animal and feedingCount field should store the total number of times the animal was fed, and hungryCount field should store the total number of times the animal got hungry.

Processing data from database: Your simulation will also issue necessary SQL queries to the database to obtain and display following information:

(i) Complete information from the table along with column headings read from the database (Hint: get column heading through using ResultSetMetadata),

(ii) The animal that consumed highest amount of food. (Calculate it by using the value in feedingCount and required food unit for an animal). If there is a tie, display all tied animals,

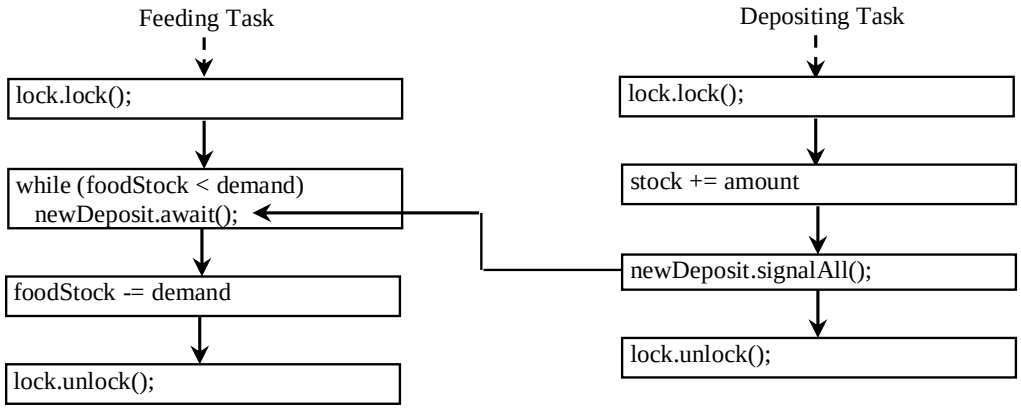
(iii) The animal that got hungry most of the time (i.e., hungryCount), If there is a tie, display all tied animals, and

(iv) Total amount of food consumed by all *n* animals.

[**IMPORTANT**: You **must** retrieve above information from the database table through SQL queries.]

Development: Write the simulation to demonstrate thread cooperation through a thread pool for the Feeding Task and Depositing Task. Suppose that you create and launch two threads, one adds food to the food stock (i.e., Depositing Task), and the other consumes food from the same food stock (i.e., Feeding Task). The second thread has to wait if the amount of food to be consumed is more than the current stock. Whenever new food is added to the stock, the first thread notifies the second thread to resume. If the amount is still not enough for a consumption, the second thread has to continue to wait for more food in the stock. Assume the initial food stock is 0, and the amount of food to be deposited and the selection of animal to be feed (i.e. the amount of food to be consumed) are randomly generated.

To synchronize the operations, use a lock with a condition: *newDeposit* (i.e., new food added to the stock). If the stock is less than the amount to be consumed, the Feeding Task will wait for the *newDeposit* condition. When the deposit task adds food to the stock, the task signals the waiting Feeding Task to try again. The interactions between the two tasks are shown in following figure.



Handling the database operations and query processing can be done through a separate thread (not within the thread pool), which should be started after finishing the Feeding Task (i.e., after shutting down the thread pool).

Output: A sample output for a typical run for *n* = 10 is shown below. The left most column displays the (random) amount of food added to each deposit task, the middle column displays on animal feeding information or waiting for food information from the feeding task, and the last column shows the status of food stock after each operation of depositing task and feeding task. Because this simulation relies on threads, thread synchronization, depositing a random amount of food to the food stock, selecting a random animal from the animal list, and the output from different

run may vary significantly. However, in every case, the thread pool should terminate, after feeding *n* animals. For example, in the following output, the thread pool terminates after feeding 2 giraffes, 2 elephants, 1 zebra, 3 deer and 2 horses (total 10 animals) selected in random order.

```
Deposit Food    Feed Animals                                Stock (kg)
Add 3kg                                                     3
                Zebra got hungry, Wait for food..
Add 7kg                                                     10
                Giraffe got hungry, Feed Giraffe 9kg        1
                Deer got hungry, Wait for food..
Add 4kg                                                     5
                Giraffe got hungry, Wait for food..
Add 1kg                                                     6
                Elephant got hungry, Wait for food..
Add 15kg                                                    21
                Elephant got hungry, Feed Elephant 15kg     6
                Giraffe got hungry, Wait for food..
Add 12kg                                                    18
                Elephant got hungry, Feed Elephant 15kg     3
                Deer got hungry, Feed Deer 3kg              0
                Horse got hungry, Wait for food..
Add 3kg                                                     3
                Deer got hungry, Feed Deer 3kg              0
                Horse got hungry, Wait for food..
Add 3kg                                                     3
                Zebra got hungry, Wait for food..
Add 7kg                                                     10
                Giraffe got hungry, Feed Giraffe 9kg        1
                Giraffe got hungry, Wait for food..
Add 8kg                                                     9
                Horse got hungry, Feed Horse 5kg            4
                Zebra got hungry, Wait for food..
Add 5kg                                                     9
                Elephant got hungry, Wait for food..
Add 5kg                                                     14
                Zebra got hungry, Feed Zebra 5kg            9
```

```
                Elephant got hungry, Wait for food..
Add 3kg                                                          12
                Horse got hungry, Feed Horse 5kg                 7
                Giraffe got hungry, Wait for food..
Add 3kg                                                          10
                Deer got hungry, Feed Deer 3kg                   7
Finished FeedingTask thread....
Finished DepositingTask thread...


Query result from DBOperation thread...
     1. All rows from FeedingData table:
     animalName feedingCount hungryCount
     Deer       3            4
     Elephant   2            5
     Giraffe    2            6
     Horse      2            4
     Zebra      1            4

     2. Highest amount of food consumed by: Elephant

     3. The most hungry animal: Giraffe

     4. Total food consumed by all 10 animals: 72 kg
```

Submission Guideline:

Submit following files to the "Submit Assignment 3" Dropbox available at Slate.

i. Feed Zoo Animal project as a single archive file (e.g., .zip version)

ii. A document file (Word or pdf) containing **screenshots** showing outputs from **2** sample runs of your program (as shown above) with two different value for *n* (e.g., 10 and 15) (30% marks deduction for missing/incorrect screenshots.)

---: End of Assignment 3:---