Middle East Technical University
Northern Cyprus Campus

CNG 443: Introduction to Object-Oriented Programming Languages and Systems
Assignment 4: FarmApp with Database and Security Checks

Date handed-out: 23 December, Friday
Date submission due: 06 January, Friday, 23:55 (Cyprus time)

Learning Outcomes

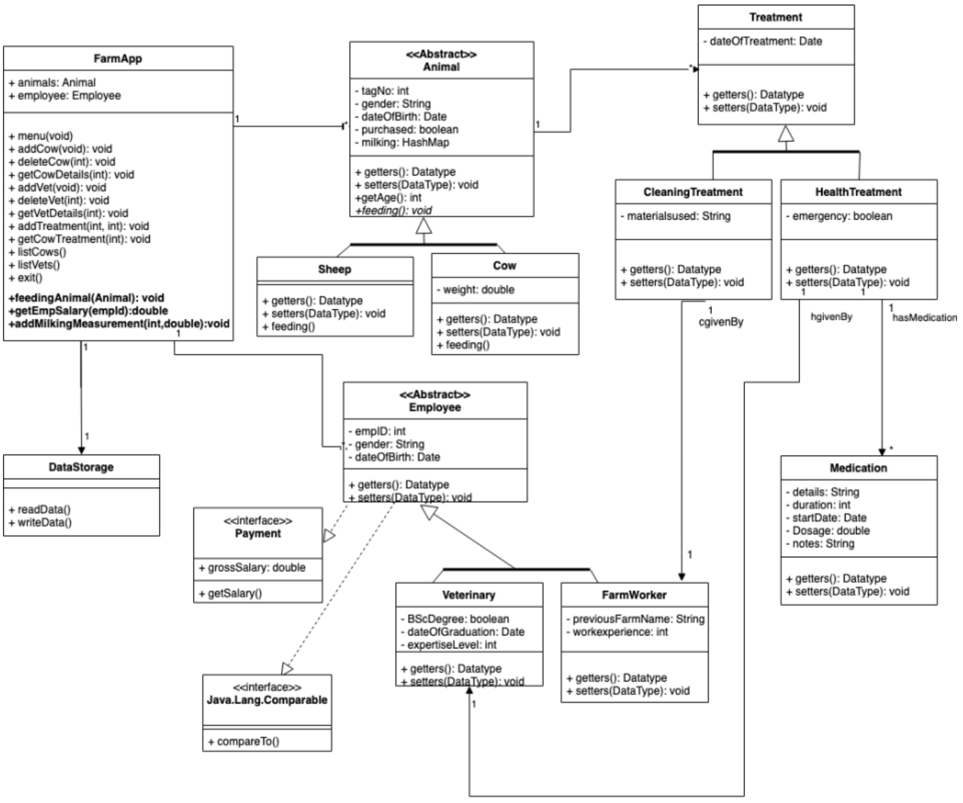
On successful completion of this assignment, a student will:

- Have practiced JDBC API.
- Have practiced Multithreading in Java.
- Have practiced Object serialization API in Java.
- Have practiced how to use Java Security API (in particular, MD5 implementation).

This assignment has two aims: (1) to learn how to connect a Java application to a backend database; (2) to learn how to use part of the object serialization and Java Security API, and multithreading. In order to practice these two objectives, this assignment is based on the previous assignment. We will improve the third assignment and connect it to a backend database and also use MD5 algorithm to do security checks in parallel with the main application.

PART1: Database Connection

In the previous assignment you created an application for Farm management. The figure below shows a summary class diagram for this application.



In the previous assignment, you stored your data in external files, but in this assignment, you need to use the **DataStorage** class to connect to an external database and store your customer details. Your application will work as follows:

- When you start your application, you need to access an SQL database and load the data from that database.
- When you close your application, you need to store the data back to the SQL database.
- You can find attached SQL script to create your database. In this assignment, you can use Oracle, MySql, MariaDB or postgreSQL. The attached script is for MySQL so if you want to use other DBMS, then you need to update your scripts.
- In this assignment, you need to use the JDBC framework.

Database Schema: The given database schema has only two tables: Animal and Employee. Both tables have a column called type which is a single character. For Animal table if the type is “c”, then the animal is a cow. If the type is “s”, then the animal is a sheep. Similarly, the Employee table has a type which is a single character. It can be either “v” or “f”. If it is “v” then it is a vet, if it is “f” then it is a farmworker. You can extend the database to store the full data, however, in this assignment we are just going to make Animal (Sheep/Cow) and Employee (Veterinary/FarmEmployee) details and not the Treatment and Medication details. They can be added when the application is running but they will NOT be stored persistently in the assignment., i.e., after the Animal list is populated from the Database, animals will have empty Treatment lists, which can be populated through the application.

PART 2: Object Serialization and Security

When the user wants to close the application, besides storing the data to a backend database, you need to also serialize your **Animal** objects into an external file. This will be used to check if somebody is trying to attack your Animal data while the application is not running. You will do this as follows: when user closes the application, you will make your Animal objects persistent to a file, generate an MD5 for that file and write it into another external file. When the application is loaded again, you will create a thread that reads the serialised animal objects, regenerates the MD5 for the serialized objects in your external file and check if it is the same with the MD5 that you stored when the application was closed. If they are the same, that means the serialized objects are not modified, if they are not then you will need to warn the user that the data has been updated.

NOTE: If you have not submitted your previous assignment, then for this assignment you can only work with the Animal (Sheep/Cow) and Employee (Veterinary/FarmEmployee) details to connect the application to the database and also do the serialization/threading and security part based on the Animal data.

Submission

You need to submit the following. Please organise your submission as a *single ZIP file* that includes the following:

- [**source folder**]: This should include your full source code and “FarmApp.jar” jar file of your application.

NOTE:

- No need to submit binary data files as your application needs to create them when the application is first used.
- Your source folder should also include a jar file called "FarmApp.jar".
- No need to generate JavaDoc and submit generated files.

Assessment Criteria

This assignment will be marked as follows:

Aspect	**Marks (Total 100)**
DataStorage implemented	10
Fully working database connection	15
Fully working database read and write	30
Object Serialization	20
MD5 check in a Thread	25

NOTE: No need to add javadoc to your GUI components but the rest of the core application and action listeners, javadoc needs to be provided.

In order to get full mark, all your classes should have the following: a constructor with full parameters, at least two constructors with partial parameters, overridden toString method, javadoc. The following grading scheme will also be used for the requested methods.

Fully working	0.2
Appropriate reuse of other code	0.2
Good coding style	0.2
Good Javadoc comments	0.2
Good and neat test results	0.2