Middle East Technical University
Northern Cyprus Campus

CNG 443: Intr. to Object-Oriented Programming Languages and Systems
Assignment 2: FarmApp -- Restructuring

Date handed-out: 11 November, 2022
Date submission due: 25 November 2022, 23:55 (Cyprus time)

Learning Outcomes

On successful completion of this assignment, a student will:

- Have used an UML class diagram to implement an application.
- Have practiced class hierarchy and the relevant design and implementation decisions.
- Have learnt how to maintain different types of objects.
- Have practiced and used abstract classes, and interfaces.
- Have learnt how to create a package for an application.
- Have also practiced Exception handling in Java.

Requirements

This assignment is about creating a small Java application for a large farm to manage the cows, sheep, their treatments and medication, as well as the vet and other staff working in the farm. In particular, it aims to maintain information about the animals, their illnesses and also the history of treatment of their illnesses. The figure below shows a summary class diagram for this application.

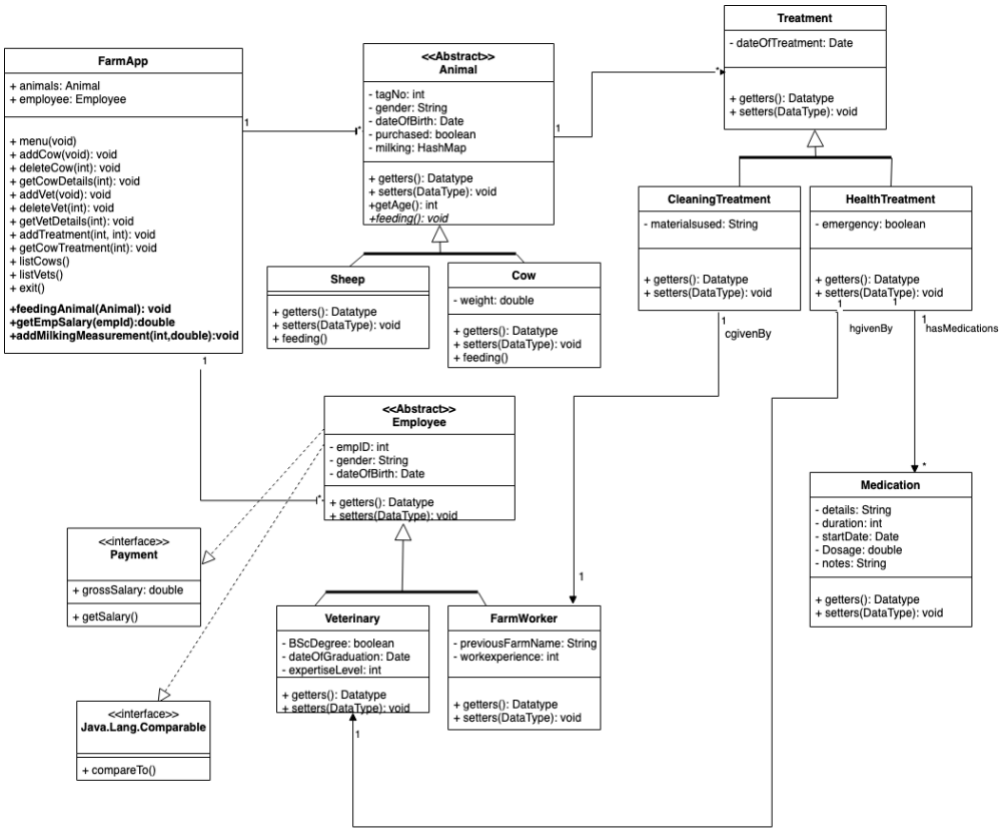

Figure 1 Updated FarmApp Application Class Diagram

The overall requirements are based on this class diagram, which is also summarised below:

- The main application called **FarmApp** will be used to maintain information about animals and also the employees working in the farm. FarmApp will also have the main method and will provide the overall interaction with the application. Therefore, this class should include the main method where an instance of this class is constructed and the menu of commands is displayed to the user. Since we have not yet covered Graphical User Interfaces (GUI) in this course, you need to implement it as a command-line application. The required methods are as follows:
    - ***void menu()***: This method will display the interaction menu to the user;
    - ***void addCow()***: This method will add new cow to the list of cows maintained. Each cow needs to have unique tag number, the gender will be specified as either male or female, the date of birth will be recorded and also whether the cow is purchased **or whether it is farm-raising and you need to also record its weight and create a hashmap for recording milks measurements. These measurements will be done regularly but when the cow is added it will be initially empty**.
    - ***void deleteCow(int tagNo)***: This method will read a tag number of a cow, and delete the corresponding cow object. If the tag number does not exist, the program should provide an appropriate error message.
    - ***void getCowDetails(int tagNo)***: Given a tag number, this method will display the Cow details (**no need to show medication and treatment details**). If the tag number does not exist, the program should provide an appropriate error message.
    - ***void addVet()***: This method will add a new vet with a unique ID.
    - ***void deleteVet(int vetID)***: Given a vet ID, this method will delete a vet. If the vetID does not exist, the program should provide an appropriate error message.
    - ***void getVetDetails(int vetID)***: Given a vetID, this method will display the vet details. If the vetID does not exist, the method should provide an appropriate error message.
    - ***void addTreatment(int vetId, int tagNo)***: **This method will record the treatment given by a particular vet to an animal with the given tag number. Animals can have two kinds of treatments: Cleaning and Health related. Therefore, you should ask user about the type of the treatment and add accordingly.** If the vet with the given ID or the cow with the given ID does not exist then the relevant error messages should be given. Please note that when a treatment is added, **the relevant medication should also be recorded, and similarly vet and farmworker should also be recorded accordingly.**
    - ***void getCowTreatment(int tagNo)***: Given a tag number, this method will return the treatment that has been done to the given cow. **If the treatments have medication, they should also be displayed**. If it has

multiple treatments all will be returned. Please note that if the tag number does not exist, relevant messages will be given.

- ***void getCowTreatment(int tagNo, Date dateOfTreatment)***: Given a tag number, this method will return the treatment that has been done to the given cow on the given particular date. If it has multiple treatments all will be returned. Please note that if the tag number does not exist, relevant messages will be given. If there is no treatment on a particular date then relevant error messages will be given.
- ***void listCow()***: This method will list all the cows. You do not need to display treatments and medications for a cow. **You need to display only the relevant cow details.**
- ***void listVet()***: This method will list all the vets.
- ***void feedingAnimal(int tagNo)*****: This method will call the feeding() method of the animal with the given tagNo to get information about how an animal needs to be fed. A cow's feeding decision is done based on their age and also the weight. It can be decided as follows:**
  - **If a cow is younger than 3 years old then they are only fed with grass.**
  - **If a cow is older than 5 years old and weight is less than 500 then it needs Total mixed ration (TMR) is a diet that includes hay, fermented grass (silage), maize silage and high energy grains like brewers grains, soy bean, cotton seed and citrus pulp.**
  - **If a cow is older than 10 years old, it needs grains and oilseed meals.**
  - **In all other cases, the cow needs be fed with grass and grains.**

  Similarly, a sheep's feeding decision is done based on their age as follows:
  - **If a sheep is male and younger than 5 years old then only grass.**
  - **If a sheep is female and younger than 8 years old then only grass.**
  - **If a sheep is male and older than 5 years old then Total mixed ration (TMR) diet is needed.**
  - **If a sheep is female and older than 8 years old then Total mixed ration (TMR) diet is needed.**

  Based on the logic given above, this method will display the relevant message based on the tagNo. TagNo will be used to decide about the animal, get the relevant details and then display the relevant messages given above.
- ***double getEmpSalary(int empId)*****: This method will return the salary of an employee. If the employee with that ID does not exist, it should give the necessary error messages. Employee salaries are calculated as follows:**
  - **Veterinary: Their salary will be calculated as follows: grossSalary given in the Payment interface + %10*grossSalary*(number of years since their graduation).**

For example, if the grossSalary is 10000 and it has been 2 years since they graduated then their salary will be 10000+10000*0.10*2=12000.

- **FarmWorker: Their salary will be calculated as follows: grossSalary given in the Payment interface + %2*workexperience. For example, if the grossSalary is 10000 and work experience is 2 years then they will receive 10000+10000*0.02*2=10400.**

- ***void addMilkingMeasurement(int tagNo, double amount)*: The farm owner would like to record milking measurements that are done for an animal. They measure once in a while to see how much milk an animal provides, and the farm owner would like to record these. This method will get the animal tagNo, the amount of milk provided. To store such details, the Animal class has a Hash Map that will store date (the day of the entry will be used) and the milk amount recorded. There can be only one measurement per day, therefore, the date can be used as a key. Values will be stored as follows:**
  - **2022-10-11=30**
  - **2022-10-12=40**

  That means for an animal there were two measurements with the values of 30 and 40. Hint: You can use the LocalDate class.
- ***void exit()***: This method should terminate the program.

- In the methods list above, the following naming convention used. Add means you add data, delete means you delete data and list means you display data.
- In this assignment, you need to do exception handling. Please make sure that all the **checked exception types are handled** in your methods.
- The given class diagram has all the fields and methods needed, so please follow the diagram. If you need extra fields, you can but please make sure that you update your class diagram.
- Since you did not learn how to make your class persistent or use a database, you will lose data every time you run your application. Therefore, you need to create some objects before you start your application. Your application needs to start with **2 Cow objects, 2 Sheep, 2 Vet objects, 2 Farmworkers** with each animal having **one treatment of each type** and for each health treatment one medication object. To create this data, you need to create a class which is called *PopulateData* that can be used to populate your application with these initial data.
- Once you complete your implementation, fully update the UML class diagram and submit it as well. Original UML diagram was created with Draw.io. You can use that or any other tool to create your updated UML diagram (e.g. Draw.io (www.draw.io), LucidChart (www.lucidchart.com/), Visio, etc.). This assignment also has an attachment that is the Visio version of this diagram so that you can import it to a tool and edit it.

Environment: As a development environment, you can use any IDE you like but you are strongly recommended to use **Intellij** (https://www.jetbrains.com/idea/).

Submission: You need to submit the following:

Please organise your submission as a *single ZIP file* that includes the following:

- [**Jar file**]: A JAR file that can be executed on a command line. Mark sure that FarmApp is the main class.
- [**doc folder**]: This should include the full Javadocs generated.
- [**source folder**]: This should include your full source code.
- [**diagram folder**]: This should include the updated UML.

Extra Requirements:

Some additional requirements are listed below:

- We have not yet covered how to use a Database or make objects persistent in this course. Therefore, this assignment maintains objects such as cows and vets in arrayLists.
- We have not yet covered Graphical User Interfaces (GUI) in this course. So please provide a command-line interaction (CLI).
- For each class, please decide what kind of constructors are required, the access types of methods and fields. If you use private fields, make sure that you provide accessor and mutator methods. For each class, you need to do constructor overloading and provide at least two constructors.
- **Regarding Date representation, you can use LocalDate and Calendar classes**.
- Pay attention to the overall design, layout and presentation of your code.
- You need to submit your Java code with proper JavaDoc comments. For each class, you need to have used at least @author and @version. For each method, you need to use at least @param and @return.

Assessment Criteria

This assignment will be marked as follows:

Aspect	**Marks (Total 100)**
All classes are implemented	10
All class hierarchies are implemented	10
All interfaces are implemented and used	10
For all classes constructors are properly implemented	10
For all classes all required data fields are implemented	10
For all classes all required methods are implemented	10
All methods in the FarmApp are implemented	30
Package Structure and Jar for Invoking the application	5
Exception Handling is done	5

For each of the items above, we will also use the following grading criteria:

Half working	%20
Fully working	%20
Appropriate reuse of other code	%10
Good Javadoc comments	%10
Good quality code	%40