Middle East Technical University
Northern Cyprus Campus

CNG 443: Intr. to Object-Oriented Programming Languages and Systems
Assignment 1: FarmApp

Date handed-out: 20 October 2022, Thursday
Date submission due: 3 November 2022, Thursday, 23:55 (Cyprus time)

Learning Outcomes

Args.length check error vermesin diye command line'dan alrken

On successful completion of this assignment, a student will:

- Have written a class suitable for instantiation.
- Have written Javadoc comments for it.
- Have written code which creates and manipulates instances of this class.
- Have begun to appreciate the usefulness of reusing code, even within one class.
- Have developed a class based on use of an array as a means of storing a collection of objects.
- Have designed and written reusable methods for adding, searching for and deleting objects to/in/from the collection.
- Have defined and implemented appropriate test program to check the operation of the collection class.
- Have used a UML class diagram to implement an application.

Requirements

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

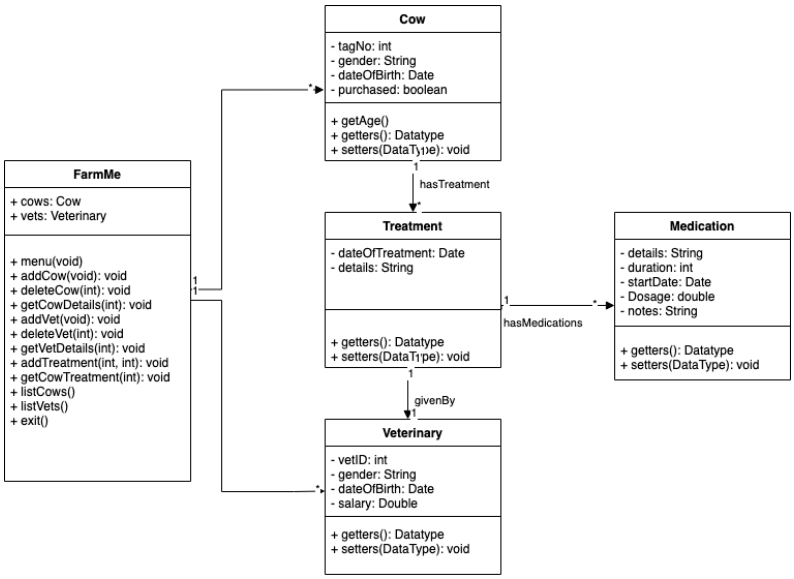

Figure 1 FarmApp -- Class Diagram

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 cows and also the vets 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;
    - ***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. If the tag number does not exist, the program should provide an appropriate error message.
    - ***void addVet()***: This method will add a new vet.
    - ***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 to a cow with the given tag number. If the vet with the given ID or the cow with the given ID does not e xist then the relevant error messages should be given. Please note that when a treatment is added, the relevant medication should also be recorded.
    - ***void getCowTreatment(int tagNo)***: Given a tag number, this method will return the treatment that has been done to the given cow. 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.
    - ***void listVet()***: This method will list all the vets.
    - ***void exit()***: This method should terminate the program.

- 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 3 Cow objects, 3 Vet objects, with each cow having one treatment and for each 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:

1. Store all your Java classes (.java files) and updated UML class diagram (.png format) in a single folder, ZIP the folder, and submit this ZIP file to ODTUCLASS.
2. Create a Jar file that can be used to execute your code. Please see the instructions below. Package all classes into a package called FarmApp.jar. With this package one should be able to run your application by just typing "java –jar FarmApp.jar".

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.
- You should use the Date class provided in java.utils. In order to read the date from the user, read a string with the "dd/mm/yyyy" format, in which dd, mm and yyyy represent the day, month and the year, respectively. Study the "Parsing Strings into Dates" section provided in:
    - https://www.tutorialspoint.com/java/java_date_time.htm

- 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/method, you need to have used at least @author, @version, @param and @return.

Assessment Criteria

This assignment will be marked as follows:

Aspect	**Marks (Total 100)**
All classes are implemented	25
Constructors are properly implemented for all classes. There should be at least **two** different constructors for each class.	10
All methods are implemented	25
Command line interaction and menu	10
Initial data population is done (3 Cow objects, 3 Vet objects, with each cow having one treatment and for each treatment one medication object)	10
Package	10
PopulateData Class	5
UML Class Diagram	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[1]	%40

[1] 15 principles will be considered during grading:
https://www.informit.com/articles/article.aspx?p=2223710