PGCert IT: Programming for Industry

Assignment 02: Farm Manager 1964

Due: 5 April 2020 at 11:59pm

This assignment requires you to modify existing classes in the provided **FarmManager** application. You are also required to create new classes for this game. There are eight tasks in this assignment to complete. To run the game, simply run the **FarmManager** class.

Before starting the assignment, read through and gain an understanding of the existing code.

Notes:

- The exercises are provided in the Zip file (industry_assignment_02.zip).
- The assignment is out of 45 marks.
- You should have the following source files for the assignment:
    - `Animal.java`
    - `Cow.java`
    - `FarmManger.java`
    - `IProductionAnimal.java`
    - `Keyboard.java`
- After completing the assignment, you should have a UML class diagram, answers to questions and two additional source files:
    - `Chicken.java`
    - `Your own animal class`
- Zip your IntelliJ project for the assignment, including all the source files, the UML class diagram, and answers to questions before the due date.
- **IMPORTANT**: Read the instructions carefully before attempting each task.

Introduction

In this game, you can buy animals, raise them, and sell them back for a profit. Just remember don't run out of money!

Currently, the game has the following commands:

- *buy <animal name>* - buys new animals.
- *sell* - sells all your animals.
- *feed* - feeds as many animals as you have money for.
- *feed <animal type>* - feeds as many animals as you have money for that are of the given type.
- *stock* - lists which animals you have in stock.
- *money* - tells you how much money you have.
- *harvest* – harvests all animals that produce products.
- *quit* – provide a summary financial statement then exit the game.

Note that some commands will not work properly until you have implemented the required functionality.

Task One. UML Class Diagram (10 marks)

Create a UML class diagram to model the static structure of the **FarmManager** application. Your class diagram should capture the key classes and interfaces, and the relationships between them.

Task Two. Complete Farm Constructor (2 marks)

Complete the constructor of the **Farm** class which takes an integer parameter in the Farm class so that:

a) **money** is initialised to the given parameter.
b) **STARTING_MONEY** is initialised to the given parameter.
c) The **animals** array is initialised to hold 10 **Animal** objects.

Task Three. Complete Farm's buyAnimal method (4 marks)

The **buyAnimal** method in the **Farm** class allows users to buy animals for the farm. The purchase of an animal is only successful when:

a) The animal type exists,
b) The money is sufficient, and
c) The animals array is not full, i.e. the array does not contain any null elements.

The first two conditions have been implemented in the **buyAnimal** method. Now, complete the method by implementing the third condition. Here is some pseduocode to help you to complete this method:

- Go through each element in the **animals** array
    - If the element is null, assign the newly created animal to that element. Deduct the price of the animal from the money you have for the farm. Then, return true.

Task Four. Complete Farm's feed method (4 marks)

Complete the **feed** method in the **Farm** class to feed all the animals of the specified type on the farm. An animal is only fed when there is enough money to feed it. When you fed an animal, you also need to subtract the cost to feed from the money you have on the farm, and call the feed method on the animal. Don't forget to stop going through the **animals** array when there is no more animal to feed (i.e. when the animal becomes null)!

Hint: You may use the **getType** method on the animal to get the animal type. Then, compare to see if the type of animal matches the one specified from the parameter. The match should be case insensitive.

Task Five. Complete Farm's printStock method (4 marks)

Complete the **printStock** method in the **Farm** class to print information for all animals on the farm by calling the **toString** method on each animal. If there are no animals on the farm, simply print the message `"No animals on the farm!"`.

Hint: There are no animals on the farm if all elements in the **animals** array are null.

After completing the first five tasks, the **FarmManager** game should produce similar outputs as the following example:

```
Welcome to Farm Manager 1964!
>> stock
No animals on the farm!
>> money
$10000
>> buy cow
>> stock
Cow - $1000
>> feed cow
>> stock
Cow - $1100
>> buy cow
>> stock
Cow - $1100
Cow - $1000
>> feed
>> stock
Cow - $1200
Cow - $1100
>> money
$7520
>> sell
You sold all your stock for $2300
>> stock
No animals on the farm!
>> quit
Thanks for playing Farm Manager 1964!
Your farm finished with $9820
You made a loss of $180
```

Task Six. Create New Animals (10 marks)

a) Create a **Chicken** class, which extends the **Animal** class.

i) Create a private constant **int** field, which stores 300. Name the field **MAX_VALUE**.

ii) Create a constructor for the **Chicken** class that does not take any parameters. Within the constructor, set the **value** for the chicken to 200.

iii) Implement the **feed** method so that the **value** for the chicken increases when the chicken has been fed. The formula for increasing the **value** is:

value = value + (MAX_VALUE - value) / 2;

Note that the **value** of the chicken should not exceed the **MAX_VALUE**.

iv) Implement the **costToFeed** method to return 3 as the cost for feeding each chicken.

v) Implement the **getType** method to return "Chicken".

vi) Implement the **toString** method so that the user knows what it is. The String should be similar to the one in the **Cow** class.

b) Create a unique type of animal, which also extends the **Animal** class. Implement the appropriate constructor and methods for your animal. Don't forget to also implement an appropriate **toString** method on your animal.

c) Modify the **getAnimal** method in the **Farm** class appropriately to include **Chicken** and the animal type that you created in b).

Task Seven. Collect Products from Animals (6 marks)

Now, we would like to make some money by harvesting products from our animals. We want to be able to milk cows and collect eggs from chickens. In order to do so, you need to do the following steps:

a) Modify the **Cow** class to implement the **IProductionAnimal** interface. A cow can only be milked if its value has reached the maximum. The money you can make from milking a harvestable cow is $20, otherwise $0.

b) Modify the **Chicken** class to implement the **IProductionAnimal** interface. You can always collect eggs from a chicken regardless of its value. The money you can make from collecting eggs is $5.

c) Complete the **harvestAll** method in the **Farm** class so that you can make money by harvesting the products from all animals that implement the **IProductionAnimal** interface. You can only harvest an animal if it is harvestable. If the animal is harvestable, use the **harvest** method on the animal and add the money made from harvesting to the **money** you have on the farm.

Hint: use **instanceof** to determine if an animal is an instance of **IProductionAnimal**.

After completing all tasks, the **FarmManager** game should produce similar outputs as the following example:

```
Welcome to Farm Manager 1964!
>> money
$10000
>> stock
No animals on the farm!
>> buy cow
>> buy chicken
>> money
$8621
>> stock
Cow - $1000
Chicken - $200
>> harvest
>> stock
Cow - $1000
Chicken - $200
>> money
$8626
>> feed
>> stock
Cow - $1100
Chicken - $250
>> money
$8563
>> sell
You sold all your stock for $1350
>> quit
Thanks for playing Farm Manager 1964!
Your farm finished with $9913
You made a loss of $87
```

Task Eight. Questions (5 marks)

In your own words, answer the following questions. Please save your answers in a text file.

1. Give an example of inheritance from this assignment.
2. Briefly explain the differences between the **private** and **protected** instance variables used in this assignment. Why is not every variable **private** in the assignment?
3. Briefly explain why we cannot create an instance of the **Animal** class.
4. What is a key benefit of having the **IProductionAnimal** interface?
5. What could be the benefits and consequences of changing the **Animal** class to an interface?

Marking Guide

Your assignment will be marked based on the correctness of your program and your programming style. Here are some questions to consider for the programming style:

- Is the code well-structured?

- Is the code self-explanatory?
- Can you understand the code easily?
- Are all variables properly defined with meaningful and clear code?
- Is there any commented out code?

Marks will be deducted if your code cannot be compiled or has bad programming style.