EECS 2030: Lab Test 2

Part 1: Getting Started

Download a zip file containing the labtest Eclipse project.

Import the project into Eclipse by doing the following:

1. Under the **File** menu choose **Import...**
2. Under **General** choose **Existing Projects into Workspace** and press **Next**
3. Click the **Select archive file** radio button, and click the **Browse...** button.
4. In the file browser that appears, navigate to your download directory (exactly where this is depends on what computer you working on; on the lab computers the file will probably appear in your home directory)
5. Select the downloaded **zip** file and click **OK**
6. Click **Finish**.

Explore the existing methods and the test cases, try to understand the purpose of each line. For example, what parameters the constructors are expected to take, what the output should be, and why some operation(s) should be disallowed.

Part 2: Design and Implementation

The following classes are included (listed alphabetically):

Animal

AnimalFarm

Goat

Horse

Pig

and

AnimalsTester

You are to design the class hierarchy and the API in such a way that the main method in the last class works correctly and produces a desired output, and the classes satisfy the following:

- (10 pts) any animal must have a name, which may be empty, stored in a **String** field called **name**; once it's set, it should not be possible change it
- (10 pts) the animals may be created with a name or without specifying a name (in the latter case, the name of the animal should be an empty string)
- (10 pts) the name of any object which is an **Animal** should be accessible only via a *getName()* method; only one of your classes should implement this method
- (20 pts) any animal should be able to produce a characteristic noise, by having a *getNoise()* method return a **String** object with values *Baa*, *Neigh*, or *Oink-oink*, for goat, horse, or pig objects respectively

- (10 pts) it should be possible to print the animal objects via calling *toString()*, with the output containing the type of the animal and its name, e.g., *Horse Benjamin.*
- (10 pts) an **AnimalFarm** may be created either as an empty farm, or as a farm with the animals provided in any **Collection** containing objects that are **Animal**s. While doing this, care should be taken so that the client is not able to modify any objects inside the **AnimalFarm** object
- (10 pts) it should be possible to get a **List** of any **Animal**s on the farm. While doing this, care should be taken so that the client is not able to modify any objects inside the **AnimalFarm** object
- (20 pts) it should be possible to get a count of each type of animal on the farm as a map of strings to integers.

IMPORTANT: The classes may include some methods already; you may not modify their declaration (e.g., access modifiers, return types, list of parameters). You may, however, add any non-public methods to aid you, as long as they do not lead you to violate any of the requirements listed above. Note that you can get the class name of an object by invoking *.getClass()* on it. One of the methods in the returned **Class** object returns a **String** with the class name (e.g., "Animal", "Goat").

In your implementation, use calls to superclasses' constructors and methods as much as possible to avoid code duplication. Most constructors and methods will require a couple of lines of code (1–3); *animalCounts()* will be a bit longer. It is possible that you will need to make some class(-es) or some of your method(-s) abstract. However, any public constructors and the methods are available for the tester class should still be able to function.

The tester class is provided mainly to illustrate the intended behaviour; thus, having success running it does not ensure that your code meets all the requirements. Feel free to modify the tester in any way you desire during the development; only the first 5 classes will be considered during the grading process. The output of the original tester:

```
[Testing anon Goat]
Goat
[Testing Goat Muriel toString]
Goat Muriel
[Testing Goat getName]
Muriel
[Testing Goat noise]
Baa
[Testing Horse Benjamin]
Horse Benjamin
[Testing Horse noise]
Neigh
[Testing Pig Squealer]
Pig Squealer
[Testing Pig noise]
Oink-oink
[Testing Animal list + Animals' toString]
```

```
[Goat , Goat Muriel, Horse Benjamin, Pig Squealer]
[Testing Animal farm toString]
[Goat , Goat Muriel, Horse Benjamin, Pig Squealer]
false
true
{Horse=1, Goat=2, Pig=1}
[Goat ]
{Goat=1}
[]
{}
```

Submission

Find all the `java` files in your project and submit them electronically via Moodle into a specified folder (no zipping is required). There should be 5 files (or 6 if tester class is submitted) in total.

Academic Honesty

You must work on your test alone.

No other students taking this test may be present in the same household with you during the test.

Any direct or indirect communication (including but not limited to chatting platforms, Discord, CourseHero) is prohibited during the test.

You may not share the test questions, nor the answers, with anyone, neither during, **nor at any time after the test.**

Using outside sources, other than Java language and API documentation, is not allowed.