P3: Saving and Loading Farm Data (30 marks)

***Focus:** I/O and Exception Handling*

In this part, you will implement a load/save feature in your game. Start by downloading the starter_code that comes with this assignment then do as required below.

[25 marks] **(A) `Farm` class**:

- Add a method `void exit(String filename)`
  - [+4] The method saves the farm data to a file named `filename`. Farm data includes at least `availableFood` and `animals` (i.e. the array with all animal instances)
  - [+1] Once saved, display a message that data was saved successfully.
  - [+4] If errors (exceptions) happen during the saving process, the method should print an error message. You should have at least two `catch` statements (one of them is `FileNotFoundException`).
  - [+1] Make sure your code closes the output stream regardless of whether an error happens or not.
  - [+2] Implement any required changes in your code (e.g. in other classes) so that this method works.

- Add a method `void load(String filename)`
  - [+4] The method loads the farm data from a file named `filename`.
  - [+1] Once loaded, display a message that data was load successfully.
  - [+3] If errors (exceptions) happen during the loading process, the method should print an error message. You should have at least three `catch` statements (one of them is `FileNotFoundException`).
  - [+4] If `filename` is not found: in addition to the error message from the above bullet, initialize the game with default values, e.g. `availableFood` should be 1000, `animals` array should have 100 spots and 4 animals: a chicken, a cow, and 2 llamas.
  - [+1] Make sure your code closes the output stream regardless of whether an error happens or not.

- [+2] The constructor should take one argument for the filename of the farm data, `String filename`, and loads this file upon creating a new `Farm` instance. If the file doesn't exist, an error message should be displayed and the game should be initialized with default values (similar to the above bullet).

[1 mark] **(B) Update the `FarmTest` class** with the code given below.

```
Farm myFarm = new Farm("stat.dat");
myFarm.printSummary();
for(Animal a: myFarm.getAnimals())
   a.setEnergy(Math.random()*100);
System.out.println("\nAvailable food before feeding: " +
myFarm.getAvailableFood() + "\n");
System.out.println("\nInitial list of animals:\n--------------");
myFarm.printAnimals();
System.out.println("\nAdding a clone of the second animal\n--------------");
myFarm.addClone(myFarm.getAnimals()[1]);
myFarm.printAnimals();
myFarm.feedAnimals();
System.out.println("\nAvailable food after feeding: " +
myFarm.getAvailableFood() + "\n");
System.out.println("\nAfter SORTING:\n--------------");
myFarm.animSort();
myFarm.printAnimals();
System.out.println("\nFarm summary:\n--------------");
myFarm.printSummary();
myFarm.exit("stat.dat");
```

Above code should print an output similar to the one on the next page. Note that this output may be slightly different based on several factors such as your animals' energy and the available food in the farm.

[2 marks] **(C)** Run `FarmTest` multiple times and notice how the number of animals increases. **Explain why this happens?** (*write your answer as a comment in* `FarmTest` *class*).

Submission Instructions

For this part of the project, you need to do the following:

1- Create a Java project of with any name of your choice.
2- Create a package with the name P3 and write your code within this package.
3- Zip the package P3 and name the zipped file as: YourStudentID_P3. e.g., "1234567_P3".
4- Submit the zipped file to Canvas.

Note that you can resubmit an assignment one more time, but the new submission overwrites the old submission and receives a new timestamp.

```
Data loaded from stat.dat.
The farm has:
- 7 animals (1 Chicken, 4 Cows, and 2 Llamas)
- 779.78 units of available food
Llama2 says: I'm STARVING
Llama1 says: I'm hungry
Cow1 says: I'm STARVING

Available food before feeding: 779.7799999999999


Initial list of animals:
-------------------------
Cow1    : alive at (0.0,0.0) Energy=72.0
Llama2  : alive at (0.0,0.0) Energy=12.5
Llama1  : alive at (0.0,0.0) Energy=46.6
Chicken1: alive at (0.0,0.0) Energy=55.5
Cow1    : alive at (0.0,0.0) Energy=54.2
Cow1    : alive at (0.0,0.0) Energy=62.6
Cow1    : alive at (0.0,0.0) Energy=6.0

Adding a clone of the second animal
-----------------------------------
Cow1    : alive at (0.0,0.0) Energy=72.0
Llama2  : alive at (0.0,0.0) Energy=12.5
Llama1  : alive at (0.0,0.0) Energy=46.6
Chicken1: alive at (0.0,0.0) Energy=55.5
Cow1    : alive at (0.0,0.0) Energy=54.2
Cow1    : alive at (0.0,0.0) Energy=62.6
Cow1    : alive at (0.0,0.0) Energy=6.0
Llama2  : alive at (0.0,0.0) Energy=12.5

Cow1 ate 20.0 units
Llama2 ate 9.0 units
Llama2 says: I'm hungry
Llama1 ate 9.0 units
Chicken1 ate 5.0 units
Cow1 ate 20.0 units
Cow1 ate 20.0 units
Cow1 ate 20.0 units
Cow1 says: I'm hungry
Llama2 ate 9.0 units
Llama2 says: I'm hungry

Available food after feeding: 667.7799999999999

After SORTING:
--------------
Llama2  : alive at (0.0,0.0) Energy=21.5
Llama2  : alive at (0.0,0.0) Energy=21.5
Cow1    : alive at (0.0,0.0) Energy=26.0
Llama1  : alive at (0.0,0.0) Energy=55.6
Chicken1: alive at (0.0,0.0) Energy=60.5
Cow1    : alive at (0.0,0.0) Energy=74.2
Cow1    : alive at (0.0,0.0) Energy=82.6
Cow1    : alive at (0.0,0.0) Energy=92.0

Farm summary:
--------------
The farm has:
- 8 animals (1 Chicken, 4 Cows, and 3 Llamas)
- 667.78 units of available food
Data saved successfully to stat.dat.
```