BSc (Hons) in Computing in Software Development
BSc (Hons) in Computing in Games Development
Stage 2, Semester 1

Object Oriented Programming
CA2 Project – Farm Simulator

CA2 Weighting: 25%
Due: 11-12-22
Interviews: Class time during week beginning 12-12-20

Objectives:
To practice the following:

- Design and implementation of an application in Java to demonstrate Inheritance, Composition and use of the Collections framework
- To gain experience of testing the functionality of the application with Junit
- To use interfaces and understand their benefits
- To implement FileIO
- To implement Exception Handling for both checked and unchecked exceptions
- To use version control to track the development of the application with delivery of code required at the end of each week of the project

Functional Requirements:
Create a simulator that models a farm. Farms have an id, owner, a shed and herds of animals of various types. Some of these animals such as dairy cows and goats need to be milked. Animals that need to be milked are milked by a milking machine. The milk is stored in a milk tank. If a farm has multiple types of animals that need milking the milk from each animal type should be stored in separate milk tanks – dairy cow's milk and goat's milk cannot be stored in the same milk tank. Milk tanks come with a standard capacity of 2000 litres, and with customer specific capacities.
Create the class MilkTank with the following constructors and methods:

MilkTank Methods
public MilkTank()
public MilkTank(int customCapacity)
public double getCapacity()
public double freeSpace()
public void addToTank(double amount) adds only as much as the tank fits
public double getFromTank(double amount) tankers come to collect the milk from the tank
String toString() returns a string that describes a milk tank

Dairy cows produce milk. Dairy cows have ids, names and udders. Udder capacity is a random value between 20 and 40. This describes how much milk the dairy cow produces per day. Cows can choose to be milked as many times as they want a day. Most farmers milk their cows twice a day but with robotic milking systems cows sometimes choose to be milked four to five times a day. The class DairyCow should have the following methods:

DairyCow Methods
public DairyCow() creates a new cow with a random name
public Cow(String name) creates a new cow of the given name
String getName()
double getCapacity() returns the udder capacity
String toString() returns a string that describes a cow

You should also add the capacity for the farm to have BeefCow, Goat and Sheep. BeefCow and Sheep can't be milked. Goats can and they produce 2-3 litres of milk per day. BeefCow and Sheep are valued based on pedigree, weight and age.

Milking machines handle the milking. A milking machine needs to be connected to a milk tank in order to milk an udder. A milking machine has the following methods:

MilkingMachine methods
public MilkingMachine()
MilkTank getMilkTank() returns the connected milk tank or null if the tank isn't installed
void setMilkTank(MilkTank tank) installs the milk tank
void milk(IMilkable milkable) milks the cow and fills the connected milk tank; the method returns an IllegalStateException if no tank has been installed

Animals are milked in sheds. Sheds have an id and room for one milking machine. Note that when milking machines are installed, they are connected to a shed's milk tank. If a shed does not have a milking machine it cannot be used to milk animals. A shed has the following methods:

Shed methods
public Shed(MilkTank tank)
public Shed getMilkTank() returns the shed's milk tank
public void installMilkingMachine(MilkingMachine milkingMachine) installs a milking machine and connects it to the shed's milk tank
public void milkAnimal(Animal animal) milks the animal with the milking machine, the method throws an IllegalStateException if the milking machine is not installed
public void milkAnimal(Collection<Animal> animals) milks the herd of animals with the milking machine, the method throws an IllegalStateException if the milking machine is not installed
public String toString() returns the state of the milk tank contained in the shed

You are required to implement a system to simulate a farm with the classes detailed above. The system should provide the following functionality via a menu:

1. Add a farm. Ids should be unique and auto-generated.
2. Add an animal to the farm. Duplicates are not allowed. ID values are unique and auto-generated.
3. Edit, delete and print the detail of any animal.
4. Add a shed to the farm. Duplicates are not allowed. ID values are unique and auto-generated.
5. Edit, delete and print the detail of any shed.
6. Print the details of the farm including its sheds, their milk tanks and herds.

7. Milk all animals on the farm.
8. Allow for the death of an animal. When an animal dies it is removed from the herd.
9. Allow for a milk collection, where all milk tanks on the farm are emptied.
10. Print the animals in each herd in order of value. Milkable animals are valued by how much milk they produce. BeefCows and Sheep are valued by pedigree, weight and age. Choose 2 other orderings implement suitable Comparators to print objects on the farm in sorted order of your choice.
11. Load from and store to a text file the farm, sheds and herds when the application opens or closes. These can be separate files.

(50 marks)

Design Requirements:
You should demonstrate the use of the following Object Oriented Programming techniques and Java classes.

1. Constructors, getters and setters
2. Input validation
3. Method overriding on toString() and equals() methods.
4. Use of Comparable Interface
5. Use of Enums
6. Use of testing methods
7. Use of inheritance
8. Exception Handling
9. Use of composition

(25 marks)

Usability Requirements:
You will be awarded marks based on the usability of the application. Usability will be measured using the following metrics: learnability (i.e. an interface that is easy to learn), and error tolerance (the application recovers gracefully from an input error).

(10 marks)

Version Control Requirements:
You are required to use version control to track the development of this project. You should meet the following milestones with an appropriate commit message:

1. End of week 9, 20-11-22 Code repo set up and invitation sent to your lecturer
2. End of week 10, 27-11-22 Classes demonstrating inheritance for animals, Farm, Shed, MilkingMachine classes
3. End of week 11, 4-12-22, Menu, Manager classes and interfaces to provide functionality
4. End of week 12, 11-12-22 Testing and FileIO
5. 11-12-22, 21:00 assignment due, final commit
6. You will be interviewed on your work during class time in the final week of term beginning 12-12-22

(15 marks)

Submission Requirements:

1. Source code, link to code repo and other items must be submitted in the relevant sub-folders inside a single ZIP file through Moodle.

2. Each student will be required to attend an **interview** after the deadline date. Each student will be questioned on the functionality of the code.
3. The assignment must be entirely the work of each student. Students are not permitted to share any pseudocode or source code from their solution with any other individual in the class. Students may not distribute the source code of their solution to any student in any format (i.e. electronic, verbal, or hardcopy transmission).
4. Plagiarised assignments will receive a mark of zero. This also applies to the individual allowing their work to be plagiarised.
5. Any plagiarism will be reported to the Head of Department and a report will be added to your permanent academic record.
6. Late assignments will only be accepted if accompanied by the appropriate medical note. This documentation must be received within 10 working days of the project deadline. The penalty for late submission is as follows:
   - Marked out of 80% if up to 24 hours late.
   - Marked out of 60% if 24-48 hours late.
   - Marked out of 40% if 48-72 hours late.
   - Marked out of 20% if 72-96 hours late.
   - Marked out of 0%, if over 96 hours late.