**Veterinary Surgery Application requirements**  
--------------------------------------------------------------------------------------------------------------**Basic Tasks**  
--------------------------------------------------------------------------------------------------------------

Creation of GUI that allows the user to do basic features such as input data to add records of pets to a store, show records already existing within the store, edit records of pets within a store, delete records from a store.

Object oriented programming. Use basic classes such as “Animal”, “Pet”, “Owner”, “Zoo”, “Clinic”.

Use basic object-oriented concepts such as encapsulation, inheritance, abstract classes, and composition.

Exception handling. Make sure you can’t add wrong type of data and that the code doesn’t break if a wrong type is inputted.

Retrieval of data – Allow the user to fetch data from a specific store and display them on screen. Do this by allowing the user to select a store file.

Use Javadoc’s to comment the code.

Implement a search feature.

**Classes**

**Clinic – Default Class**Clinic is an incredibly important class within this application, while the other classes are going to create objects which are going to added to lists, the Clinic class holds the HashMaps used to store the objects. The Clinic object is the object which is going to be serialised and deserialised which will allow the creation, keeping up to date, closing and tracking of clinic.

**Animal – Abstract Class**Animal is the standard basic class which the individual animals are derived from. HousePet and ZooAnimal inherit from the Animal class. Would hold the basic variables and methods used by all Animals.

**HousePet – Subclass of Animal**HousePet is a class in which inherits from Animal use of HousePet over just having all Animals being an “Animal” is that HousePet would have an Owner which is a person while a ZooAnimal would have institution (the zoo) linking to it instead. Making it, its own type of object makes sorting much easier as I can filter based on “object type”.

**ZooAnimal - Subclass of Animal**ZooAnimal is another class in which inherits from Animal, the use of ZooAnimal here is same as described within HousePet. Making it, its own type of object makes sorting much easier as I can filter based on “object type”.

**ExoticAnimal – Subclass of Animal**ExoticAnimal is a class which inherits from Animal, the ExoticAnimal class would have all basic information of an exotic animal. Making it, its own type of object makes sorting much easier as I can filter based on “object type”.

**FarmAnimal – Subclass of Animal**FarmANimal is a class which inherits from Animal, the FarmAnimal class would hold all the basic information of an exotic animal. Making it, its own type of object makes sorting much easier as I can filter based on “object type”.

**Person – Abstract Class**Person like the Animal class is the standard basic class in which the individual people are derived from. Owner and Vet inherit from the Person class. Would hold the basic variables and methods used by all people.

**PetOwner – Subclass of Person**PetOwner is a class in which inherits from Person, the petowner class would have all basic information linking giving the ability to link the animal to the petowner. Use of having owner class is that 1 petowner might have multiple pets and thus would make redundant data.

**Vet - Subclass of Person**Vet is a class in which inherits from Person, the vet class would have all basic information of the vet making certain processes much easier such as assigning a vet with a certain specialty to a certain type of animal.

**BusinessOwner – Subclass of Person**BusinessOwner is a class in which inherits from Person, while not being a must and may or may not be implemented. BusinessOwner would mostly refer to “Farmers” however can also be used in conjunction with “Zoos” which is why this class is thought of.

**Business – Abstract Class**Business is an abstract class which holds the standard information relating to a business in the case of this application the linking information between a zoo and a farm. Zoo and Farm inherit from this class.

**Zoo – Subclass of Business**Zoo is a class that inherits from the business class. Zoo will have the basic information of a Zoo much like an Owner of the owner class but has the name, phone, address, number of animals, animal IDs etc of the zoo instead.

**Farm – Subclass of Business**Farm is a class that like zoo inherits from business as it also a business. Farm will have the basic information of the Farm such as name, phone, address, number of animals, animal IDs etc. **Appointment – Default Class**Depending on time remaining I would like to add a functional appointments system. However, this is not a definite feature of the application. Appointment will have the basic information such as Animal ID, OwnerID/BusinessID, Date Appointment was Booked, Date Appointment is, Price of the Appointment/Treatment, Payment.

**To-Do List**

Setup GitHub Repository and Linked to NetBeans IDE which allows commits to be seen which helps seeing how progression was made. – Done

JavaApplicationDesignPDF for design of Application Pages + Pop up (Dialog Boxes) – In progress

Create a basic GUI using SceneBuilder (After Re-reading Handbook saw Visual Editors are not allowed…Need to redo the GUI in Hard Code using FXML as more practical to learn.) – In progress

Create the design of the application without functionality – In progress

Add basic functionality to the application, don’t take in any of the text fields.

Add the proper functionality of the application.

Make sure to validate the inputs only accepting the correct types etc.

Test application in entirety. Testing for Wrong Inputs, making sure that processes are visualised to the user through pop ups etc.

**Bibliography**

<https://docs.oracle.com/javafx/2/api/javafx/scene/doc-files/cssref.html>

<https://www.tutorialspoint.com/javafx/javafx_css.htm>

<https://www.w3schools.com/java/java_hashmap.asp>

<https://www.geeksforgeeks.org/hashing-in-java/>

[https://www.developer.com/java/hashingjava/](https://www.developer.com/java/hashingjava/https://www.w3schools.com/java/java_regex.asp)

<https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html>

<https://www.w3schools.com/java/java_regex.asp>