

**INDIVIDUAL ASSIGNMENT**

Course: Object Oriented Development with Java

CT038-3-2-OODJ

|  |  |
| --- | --- |
| **Student declaration:** | |
| *I declare that:* | 1. *We understand what is meant by plagiarism* 2. *The implication of plagiarism have been explained to us by our lecturer* 3. *This project is all our work and we have acknowledged any use of the published or unpublished works of other people.* |

|  |  |
| --- | --- |
| **Coursework Title** | **BINTANG VETERINARY CLINIC AND BOARDING SYSTEM (BVCB)** |
| **Lecturer’s Name** | **KAU GUAN KIAT** |
| **Name** | **YEW LI XIE** |
| **Intake Code** | **UC2F1505SE** |
| **ID** | **TP034896** |
| **Date Assigned** | **8th June 2015** |
| **Date Completed** | **17th August 2015** |

Contents

[ABSTRACT 3](#_Toc427587472)

[1.0 INTRODUCTION 4](#_Toc427587473)

[2.0 ASSUMPTIONS 5](#_Toc427587474)

[3.0 UML DIAGRAMS 6](#_Toc427587475)

[3.1 USE CASE DIAGRAM 6](#_Toc427587476)

[3.2 ACTIVITY DIAGRAM 11](#_Toc427587477)

[3.3 CLASS DIAGRAM 16](#_Toc427587478)

[4.0 OBJECT ORIENTED CONCEPT IMPLEMENTED IN BVCB SYSTEM 17](#_Toc427587479)

[4.1 ABSTRACTION 17](#_Toc427587480)

[4.2 ENCAPSULATION 18](#_Toc427587481)

[4.3 INHERITANCE 19](#_Toc427587482)

[4.4 POLYMORPHISM 19](#_Toc427587483)

[4.5 ABSTRACT CLASS 20](#_Toc427587484)

[4.6 CUSTOM EXCEPTION CLASS 22](#_Toc427587485)

[4.7 ASSOCIATION, AGGREGATION AND COMPOSITION 22](#_Toc427587486)

[5.0 USER MANUAL 23](#_Toc427587487)

[5.1 LOGIN PAGE 23](#_Toc427587488)

[5.2 RECEPTIONIST PAGE 23](#_Toc427587489)

[5.3 BOARDING STAFF PAGE 25](#_Toc427587490)

[5.4 VET PAGE 25](#_Toc427587491)

[5.5 OWNER PAGE 26](#_Toc427587492)

[6.0 ADDITIONAL FEATURE 27](#_Toc427587493)

[6.1 GRAPHICAL USER INTERFACE (GUI) 27](#_Toc427587494)

[6.2 ANONYMOUS CLASS 27](#_Toc427587495)

[6.3 ID GENERATOR 28](#_Toc427587496)

[6.4 ADD AND REMOVE EMPLOYEE 28](#_Toc427587497)

[6.5 JDATECHOOSER 28](#_Toc427587498)

[6.6 JTable 28](#_Toc427587499)

[7.0 CONCLUSION 29](#_Toc427587500)

[References 30](#_Toc427587501)

# ABSTRACT

A Veterinary Clinic management system is designed and developed for Bintang Veterinary. The system is written in Java with Object Oriented approach. Several assumptions were made to for the system. Besides, several UML diagrams were used to model the new system. Other than that, in the later section there is a user manual on the system that shows all the main functions.

# 1.0 INTRODUCTION

Bintang Veterinary Clinic and Boarding (BVCB) is a growing provider of veterinary care for both domestic and exotic pets. Having started out as a small business, the owner and main vet never saw the need to computerise the clinic, preferring to utilise a paper based system instead. The growing demands of the business now necessitate some form of computerisation in order to keep track of patients (pets) and their medical history, their owners, and the workload of each vet working at BVCB.

The clinic has 8 vets in total, although on any given day only 3 will be working. Each vet has one or more areas of expertise (ie fish, reptiles), therefore one of the challenges that the clinic is facing is trying to ensure the right balance of expertise is available. In addition to veterinary services, the clinic also provides a pet boarding service the management of which much also be handled by the new system.

In order to meets the need of the Veterinary Clinic, an information system called the BVCB System is designed and developed. The system is meant to be used by the receptionist, boarding staffs, vets and also the clinic owner. Each user of the system has his/her own username and password.

From the technical aspect, BVCB System is written in Java programming language and it is written with NetBeans IDE. It is designed with the Object Oriented Programming approach. Other than that, it stores its data and information in the form of binary file and also text file.

# 2.0 ASSUMPTIONS

There are a few logical assumptions made upon the BVCB Clinic and the BVCB System. First of all, the BVCB Clinic opens from 7am to 12pm, hence customer is only allowed to make an appointment within the range of 7am to 11am. Every appointment last an hour. Moreover, the BVCB clinic allows 5 types of pet which is Amphibian, Fish, Bird, Household Pet, and Reptile since those are the area of expertise of the vets hired by the clinic.

Furthermore, every customer is required to register themselves and their pets to the BVCB System before making any appointments with the vets. Each customer is allowed have a maximum numbers of 10 pets under their name.

Besides, receptionist is responsible to help the customer to choose an available vet based on the date requested by the customer and the vet week’s working rota. As for boarding staff, they are required to update the system on the pet’s last fed time right after they have fed it.

Other than that, the vet is responsible to input the consultation charges and also the boarding service charges for every pet seen by them into the BVCB System. Furthermore, the owner of the clinic is able to create every week’s working rota for the vet. The rota will then be showed up on the receptionist page. The owner is required to assign and balance up the available vets to each day of the week.

Last but not least, there is a reporting functionality for owner. There are two reports to be shown at the reporting page of the BVCB System. One of the report is the Vet Reports where it will display the number of pets seen by each vet and the amount of charges they have input into the BVCB System. For the other report is the Pet Reports where it will show the number of pet of every type and the number of pet that signed up for the boarding service.

# 3.0 UML DIAGRAMS

## 3.1 USE CASE DIAGRAM



### 3.1.1 USE CASE SCENARIO

Create Customer Profile Use Case Scenario

* The receptionist receives the registration form from a new customer where there are all the details of the customer and the pet.
* The receptionist enter all the customer details in the form into the system and save it. A new customer profile with a unique ID is created and added into the system. The system prompt the reception about the addition of the new customer profile.

Make Appointment Use Case Scenario

* The receptionist is informed by the customer to book an appointment on Sunday, 16th of August 2015, 6pm with his Poodle.
* The receptionist look for the week’s working rota to check for an available Vet that is expert on household pet and is working on Sunday. Doctor Strange is selected and his Vet ID and the appointment time is entered into the system for validation.

Appointment Hour Validation Use Case Scenario

* A Vet ID and appointment hour are entered into the system by receptionist in order to make an appointment for the customer.
* Vet ID of Doctor Strange is entered into the system. After that, the desired appointment hour of the customer which is 6pm is entered into the system. The system detects Doctor Strange is not available for booking on 6pm. The system prompt the receptionist to choose another appointment time. The customer is requested to choose another appointment hour.

Enter Prognosis & Diagnosis Use Case Scenario

* Doctor Strange is treating the ill Poodle where its details is available in the system.
* Doctor Strange opens the Poodle’s profile and enters the treatment results into the prognosis and diagnosis fields. The Vet then saves the entered results and the system updates the Poodle’s profile. The Poodle’s profile is then marked as ‘seen’ by the system.

Create Week’s Working Rota Use Case Scenario

* The owner has a list of employed Vets with all their expertise.
* The owner creates an empty new working rota for the following week. He then sets 3 different Vets for each day of the week namely Monday, Tuesday, Wednesday, Thursday, Friday, Saturday and Sunday while ensuring the balance of each available expertise. After setting vets for every day on the rota, the owner saves it and the system prompt the owner with a successful message on the creation of a new working rota.

### 3.1.2 USE CASE FORM

|  |  |
| --- | --- |
| Name | Create Customer Profile |
| Actor | Receptionist |
| Priority | Medium |
| Extension Points | None |
| Extends | Create Pet Profile, Make Appointment |
| Includes | None |
| Pre-conditions/ Assumptions | Customer does not have a profile in the system and is about to register his pet to the system. |
| Post-conditions | A new customer profile is created and updated to the system and the table. |
| Flows of Events | 1. Request customer details 2. Enter name 3. Enter NRIC 4. Enter telephone no. 5. E-mail address 6. Home address 7. Pet name 8. Pet type 9. If fields is not filled [A1] 10. Save customer details to system |
| Alternative Paths | Can be aborted any time   1. A1. Prompt User to complete filling in all the fields |

|  |  |
| --- | --- |
| Name | Make Appointment |
| Actor | Receptionist |
| Priority | High |
| Extension Points | Create Customer Profile, Make Boarding Booking |
| Extends | None |
| Includes | None |
| Pre-conditions/ Assumptions | The customer and his/her pet have an existing profile in the system. The customer provides the appointment time and date and his/her pet’s details. |
| Post-conditions | The appointment details is recorded and saved in the system. |
| Flows of Events | 1. Choose a pet among the pet profiles 2. Choose an appointment date 3. Choose a Vet based on the week rota 4. Choose an appointment hour 5. If one of the field is not filled [A1] 6. Proceed to appointment hour validation 7. Save appointment details to system |
| Alternative Paths | Can be aborted any time.   1. A1. Prompt user to complete filling in all the fields |

|  |  |
| --- | --- |
| Name | Appointment Hour Validation |
| Actor | Receptionist |
| Priority | High |
| Extension Points | None |
| Extends | None |
| Includes | None |
| Pre-conditions/ Assumptions | Appointment details (Pet ID, Vet ID, Appointment Date and Appointment Hour) are entered into the system. |
| Post-conditions | Display validation result. |
| Flows of Events | 1. Choose Appointment Hour 2. System check for clash in appointment hour for the particular vet on that day. 3. If appointment hour is not available. [A1] 4. Able to proceed to saving the appointment. |
| Alternative Paths | Can be aborted any time   1. A1. Prompt user to choose another appointment hour |

|  |  |
| --- | --- |
| Name | Enter Prognosis & Diagnosis |
| Actor | Vet |
| Priority | High |
| Extension Points | None |
| Extends | None |
| Includes | None |
| Pre-conditions/ Assumptions | There is a Vet Diary with the every appointed pet details. |
| Post-conditions | Prognosis and Diagnosis is saved to into the pet profile. |
| Flows of Events | 1. Choose a pet ID of the pet that is to be treated from the Vet Diary 2. Enter the Prognosis into the provided text field 3. Enter the Diagnosis into the provided text field after treating the pet 4. Save the Prognosis and Diagnosis into the system |
| Alternative Paths | None |

|  |  |
| --- | --- |
| Name | Create Week’s Working Rota |
| Actor | Owner |
| Priority | High |
| Extension Points | None |
| Extends | None |
| Includes | Display Vet Details |
| Pre-conditions/ Assumptions | There is available vet details in the system. |
| Post-conditions | A new Week Rota is created and saved to the system. |
| Flows of Events | 1. Create a new empty Week Rota 2. Assign 3 Vets into the Monday rota 3. Repeat step 2 for Tuesday, Wednesday, Thursday, Friday, Saturday and Sunday 4. If one of the of the day is has not set with Vets [A1] 5. Save Week Rota |
| Alternative Paths | Can be aborted any time   1. A1. Prompt user to complete the new week rota |

## 3.2 ACTIVITY DIAGRAM

### 3.2.1 CREATE CUSTOMER PROFILE ACTIVITY DIAGRAM



### 3.2.2 MAKE APPOINTMENT ACTIVITY DIAGRAM



### 3.2.3 APPOINTMENT HOUR VALIDATION ACTIVITY DIAGRAM



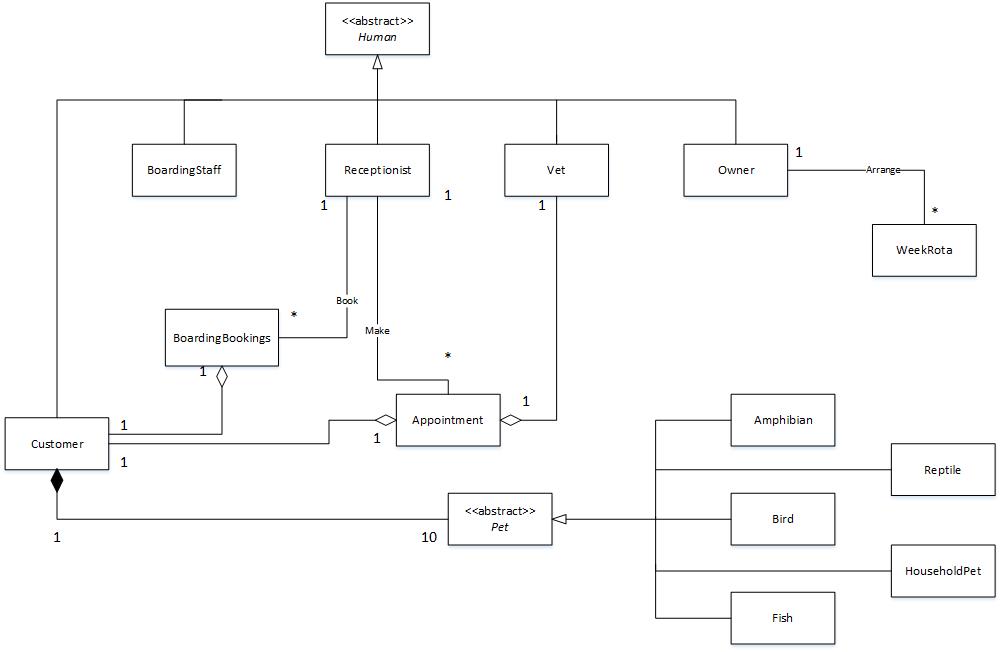
### 3.2.4 ENTER PROGNOSIS & DIAGNOSIS ACTIVITY DIAGRAM



### 3.2.5 CREATE WEEK’S WORKING ROTA ACTIVITY DIAGRAM



## 3.3 CLASS DIAGRAM

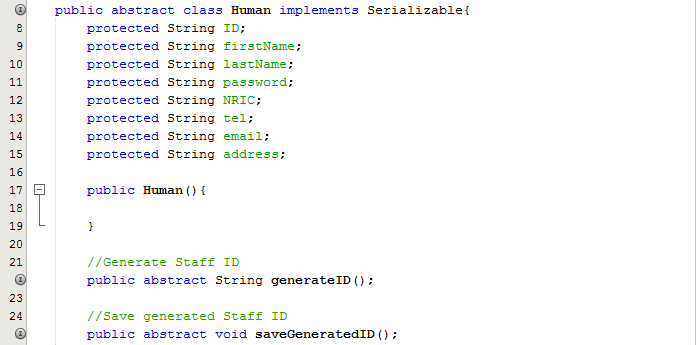


# 4.0 OBJECT ORIENTED CONCEPT IMPLEMENTED IN BVCB SYSTEM

All Java programs uses objects, the type of the objects is defined by its class or interface. “A class is a collection of fields that hold values and methods that operate on those values. Classes are the most fundamental structural element of all Java programs. You cannot write Java code without defining a class. All Java statements appear within methods, and all methods are implemented within classes.” (Flanagan, 2005) There are a few object oriented concepts used in the implementation of BVCB System, some of the main concepts are Abstraction, Encapsulation, Inheritance, and Polymorphism.

## 4.1 ABSTRACTION

An object is an abstraction (Lewis & Loftus, 2004). Abstraction is the hiding and ignoring of certain details of an object and instead showing only the important traits. Object is an instance of a class. In the BVCB System implementation, there is a total of 15 classes which has been shown in the UML Class Diagram earlier. These classes are some very good example of abstraction since every class contains fields and methods (which are the important traits of the object they represent) that describe its objects.

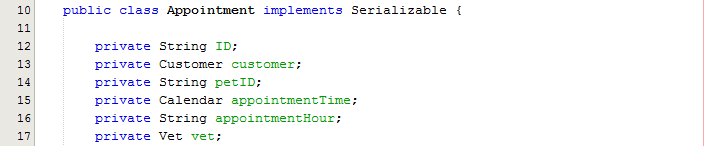


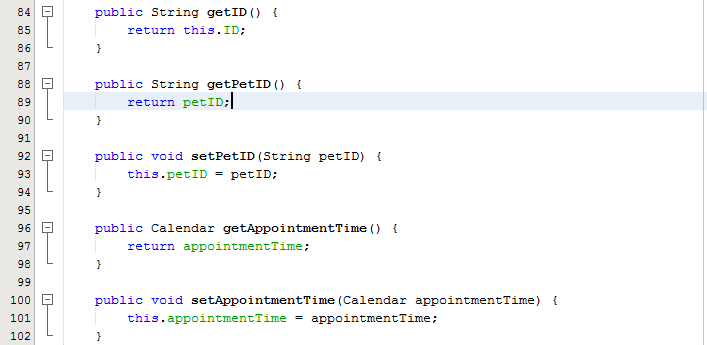
Figure

Figure 1 shows one of the class in the BVCB System that has all the fields that describe the important traits of a human user. A human user in this case, is represented by the vital information which are ID, first name, last name, password, NRIC, telephone number, E-mail address and home address.

## 4.2 ENCAPSULATION

Classes and their objects encapsulate their attributes and methods where their attributes and methods are intimately related. Encapsulation is concept where classes hide their information from other classes they communicate and interact with. Implementation details of an object are hidden within the objects themselves since they are normally not allowed to know how other objects implemented (Deitel & Deitel, 2015).





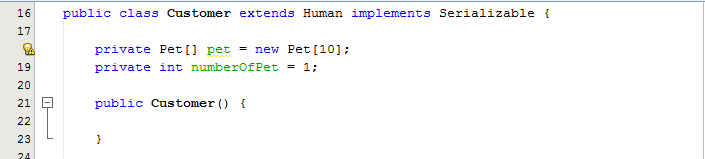
Figure

Figure 2 shows an example of encapsulation of the Appointment class inside the BVCB System. The attributes of the class is all set private and the only way to alter and read the value of the attributes is to call the get and set methods. From there it is shown that the attributes of the class is encapsulated by the methods. The methods act the barrier that encase the fields inside the class. Other than that, the ID attribute is designed to be read-only, note that there is no set method for it since an ID is unique and permanent for every appointment made. Hence, it highlights one of the strength of encapsulation where other classes or users has no way to change the ID of the appointment.

## 4.3 INHERITANCE

Inheritance is a “is a” relationship between two classes. It happens when a new class is created by acquiring an existing class’s members and possibly came up with new or modified functions. Inheritance saves time, allows a program to be implemented and maintained effectively. The existing class that the new class is inheriting from is called the superclass where the new class itself is called the subclass. A subclass can add fields and methods that the superclass does not have, hence it is more specific than its superclass. This is also called specialization (Deitel & Deitel, 2015).

In the BVCB System, there is a Human class and also Pet class that serves as the super class. Under Human class, there are 5 subclasses that extend it which are the Customer, Receptionist, BoardingStaff, Vet and Owner class. For Pet class, there are Amphibian, Fish, Bird, HouseholdPets and Reptile class. Those subclasses are more specific than their superclass where they inherits all the attributes and methods from their superclass.



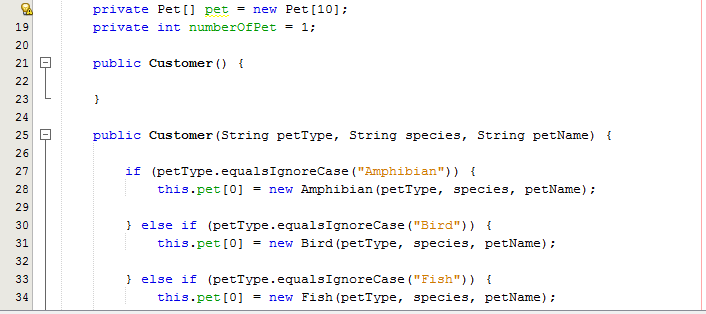
Figure

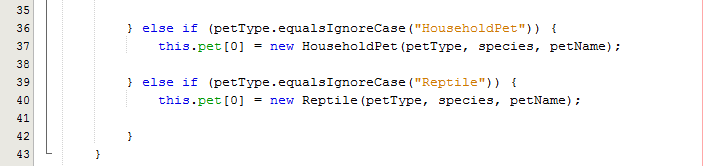
Figure 3 shows an example of a subclass Customer that is derived from Human superclass. Line 16 shows that Customer is having an “is a” relationship with Human class. As a result, a lot more Object Orientated concept can be achieved with Inheritance, such as Polymorphism.

## 4.4 POLYMORPHISM

Polymorphism is the ability of an object to take different forms. Polymorphic behaviour can only happen between a subclass and its superclass. “Polymorphism means that the actual type of the object involved in a method call determines which method is called, rather than the type of the variable being used to store the reference to the object.” (Horton, 2011)

In the BVCB System, polymorphism happens in the Pet class and its subclasses.





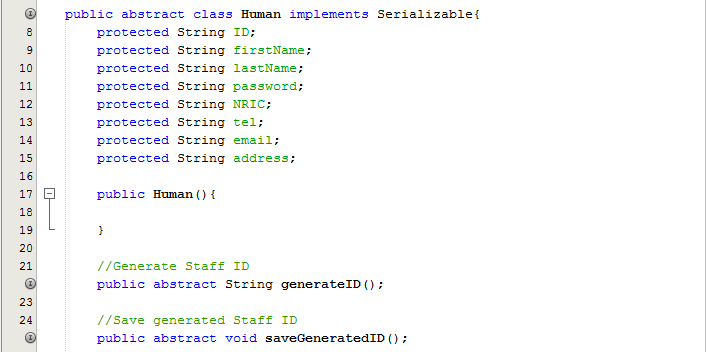
Figure

Figure 4, Line 18 shows a Pet type array variable called pet. In Line 28, 31, 34, 37 and 40, it shows that a Pet type variable is able to store any objects that is instantiated from Pet class’s subclasses. The advantage to this is that it saves a lot of time and codes by allowing the Customer class to only declare one Pet type variable instead of declaring 5 different type of variables. Besides, it also provide flexibility as it allows any method that need to take in a Pet argument to be written only once, instead of overloading the same method in order to accommodate all the 5 Pet subclasses.

## 4.5 ABSTRACT CLASS

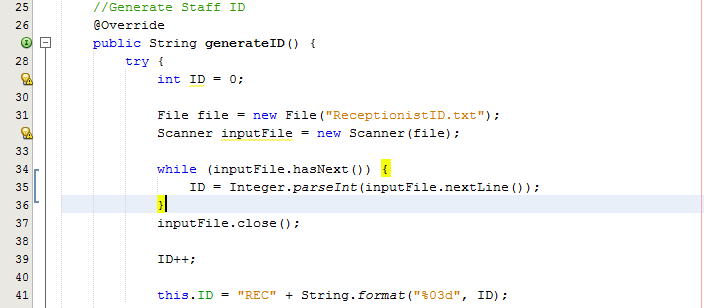
Java allows programmers to define method without implementing it by declaring it with an abstract modifier. An abstract method has no body, the method header simply just ends with a semicolon. Any class that contains abstract method has to be declared as an abstract class. An abstract class is unable to instantiate and the subclasses of the abstract class have to override all of its abstract method with a concrete method (Flanagan, 2005).

In the BVCB System, the Human and Pet class is too general to be instantiated, hence they are both abstract class. However, Human class does contains some abstract methods that is to be inherited by its subclasses.



Figure

Figure 5 shows that the abstract Human class contains 2 abstract methods. Since every subclass of the Human is required generate their own Staff ID and save it, they must override these 2 methods. As a result, abstract class and methods act as a guidelines and regulations for their subclasses.



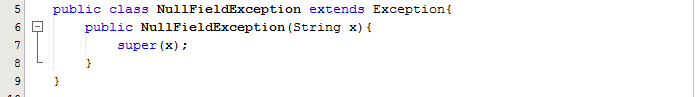
Figure

Figure 6 shows one of the subclass of Human that overrides the generateID() method.

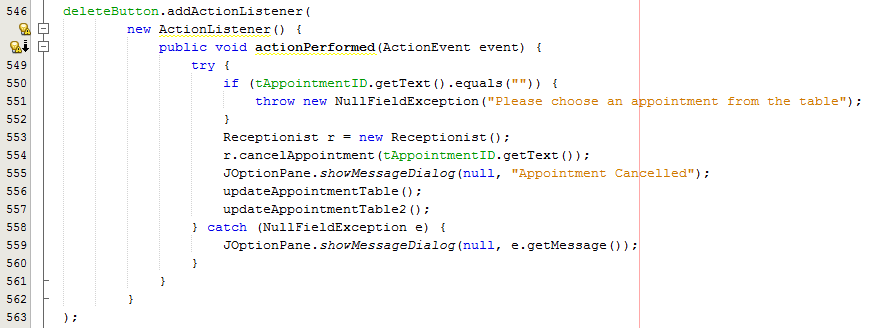
## 4.6 CUSTOM EXCEPTION CLASS

“An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions.” (Oracle, 2015) When an error occur within a method, an exception object is created and is handed to the runtime system. The action is called throwing an exception. In order to prevent the system from halting due to the exception, the program must be able to catch the exception object before it reaches the JVM.

In BVCB System, there is a custom exception class that is written specifically to be thrown whenever a ‘throw’ statement following the new exception object is written. The custom exception class extends the Exception class. In the system, whenever the user clicks a button and the system detect any empty field, the system will manually throw the custom exception object and it will be catch at the same time so that the system prompt the user to complete all the fields. The reason for it is to make sure there is no null value or empty string entered into the system. Please refer to Figure 7 and 8 below.



Figure



Figure

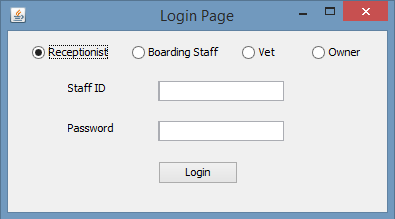
## 4.7 ASSOCIATION, AGGREGATION AND COMPOSITION

Association is a “has a” relationships between two or more objects where those objects have their own lifecycle and there is no owner. Aggregation is a specialized form of Association where objects have their own lifecycle but there is ownership. Composition is a specialized form of Aggregation where the objects has strong relationship. The child does not have their lifecycle without its parents (Chauhan, 2013).

In the BVCB System, Customer class has a composition relation with Pet class where any instances of the Pet is only created when a Customer object is created first. This is to represent the strong relation between a customer and his/her pets. Furthermore, Appointment class has an aggregation relation with the Customer and the Vet, this is because the Customer or the Vet of an appointment can be changed at any time, hence a weaker relation.

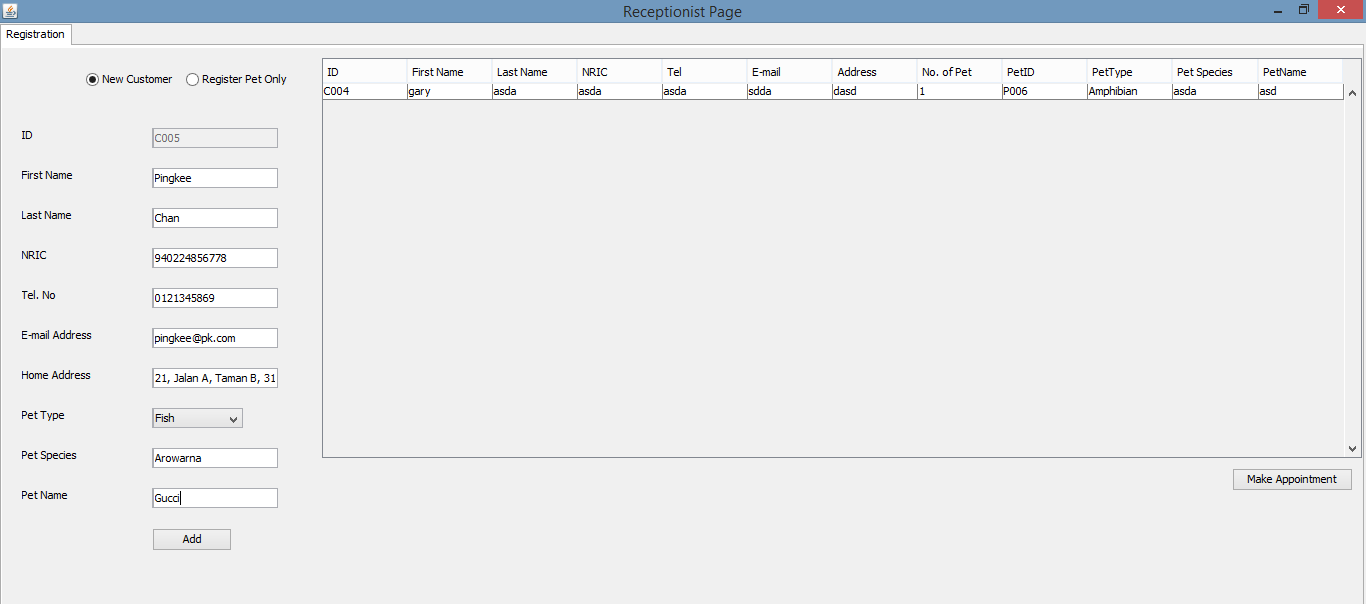
# 5.0 USER MANUAL

## 5.1 LOGIN PAGE

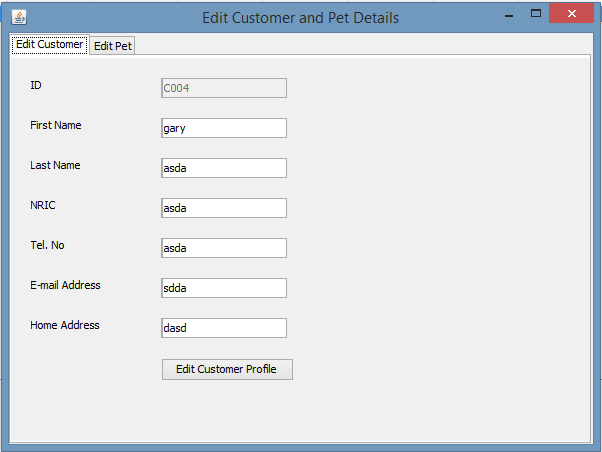


Users are required to choose their position and enter their StaffID as username and their password to proceed to their respective pages.

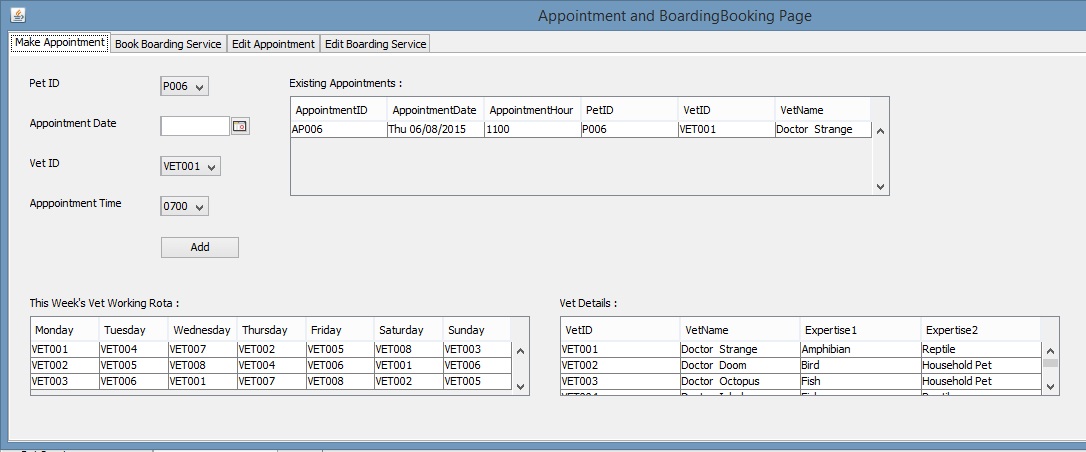
## 5.2 RECEPTIONIST PAGE



Receptionist is able to create customer and pet profile through in their first page. In order to edit a particular customer’s profile, the receptionist has to double click on that Customer’s row on the Customer table show in the Receptionist Page. A page will pop-up that allows editing of customer and pet profile.

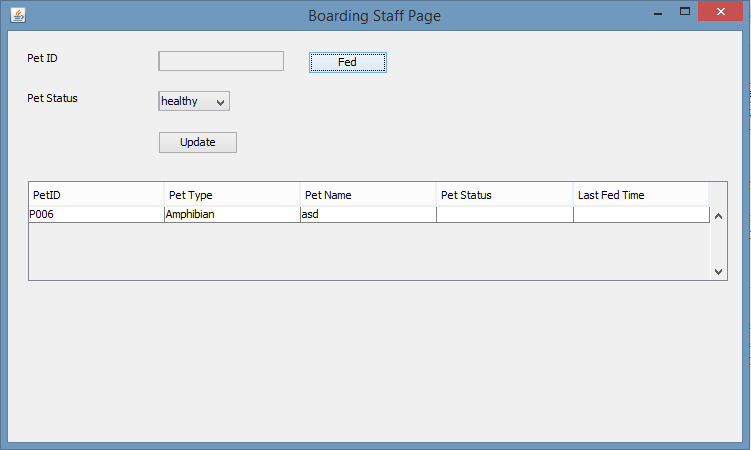


The receptionist is also able to make an appointment and boarding booking for a particular customer after selecting him/her and then press on the “Make Appointment Button” button.



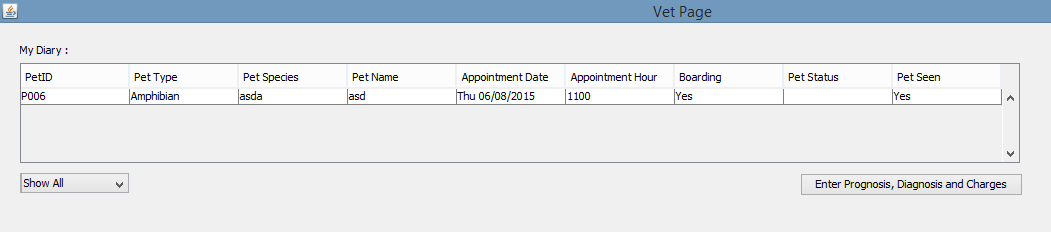
The receptionist is also able to edit any appointment and boarding service on the “Edit Appointment” and “Edit Boarding Service” tabs respectively.

## 5.3 BOARDING STAFF PAGE

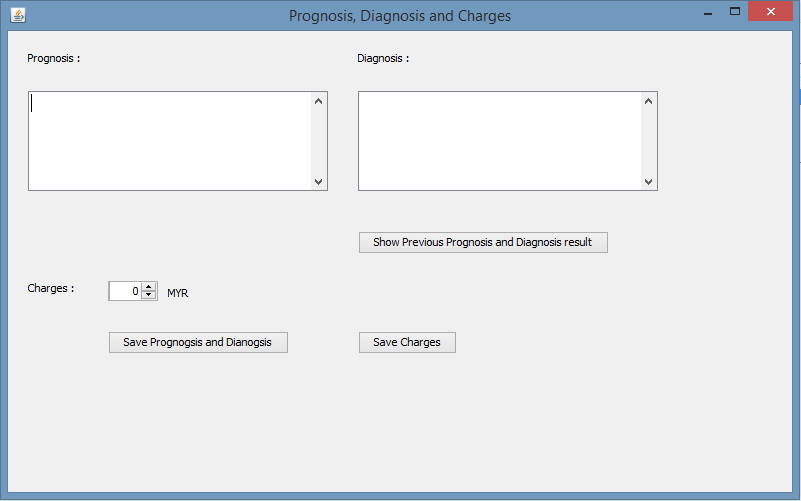


Boarding Staffs is able to choose a pet from the pet table to update its status and last fed time with only one click of the “Fed” button.

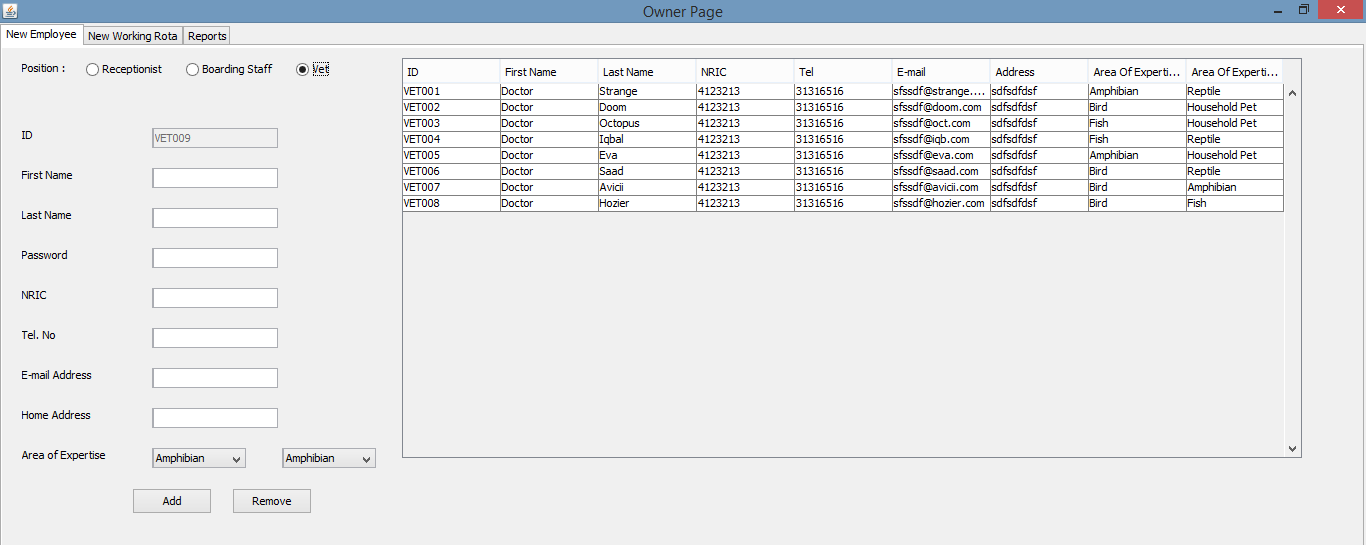
## 5.4 VET PAGE



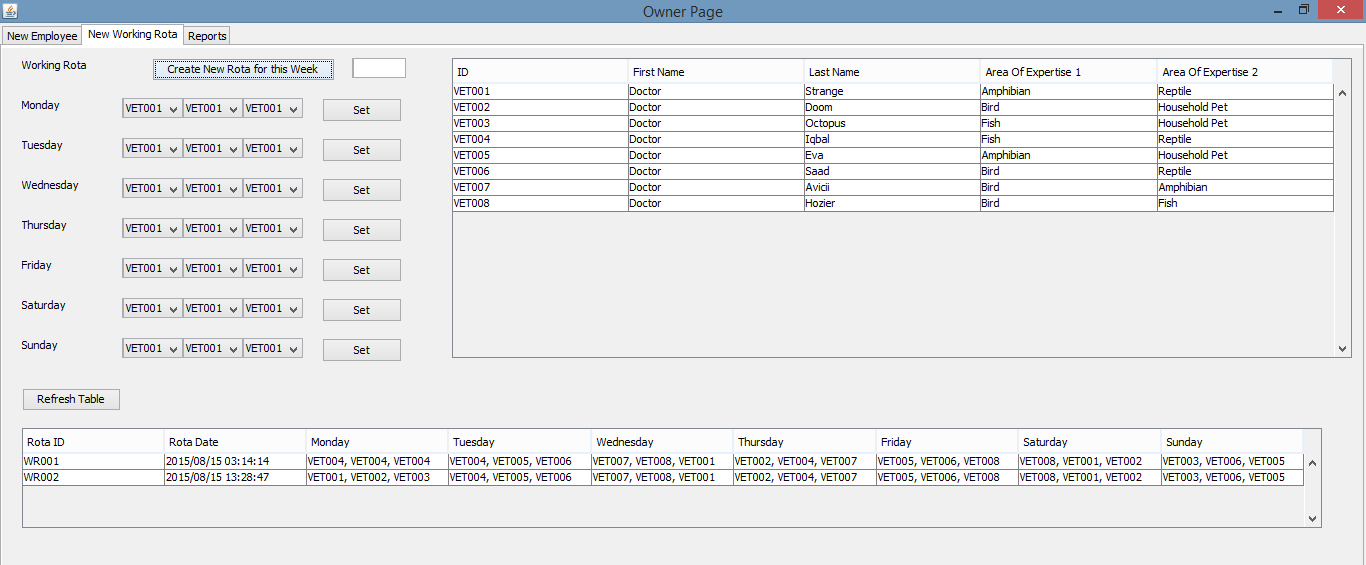
The Vet Page displays a different Vet Diary for every Vet. The Vet is able to toggle the Vet Diary to show all appointments or show only today’s appointment. Other than that, the Vet is able to enter the Prognosis, Diagnosis and Charges of any pet by clicking on the bottom right button, another page will pop-up that has text area that allow the vet to enter their findings.



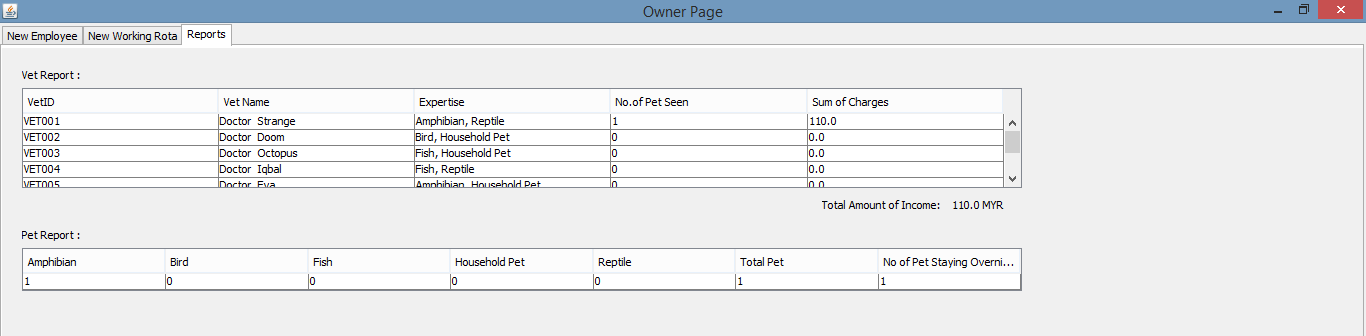
## 5.5 OWNER PAGE



The first tab of the Owner Page lets the owner add or remove any kind of employee in the system.



The second tab allows the owner to create a new working rota for the Vets.



Lastly, the owner is able to view the Vet Report and Pet Report.

# 6.0 ADDITIONAL FEATURE

## 6.1 GRAPHICAL USER INTERFACE (GUI)

The BVCB System is supported by GUI. Every GUI container and component is coded from scratch where there is no involvement of any drag-and-drop technique. GUI is able to yields high productivity and efficiency from the system and the user since everything is made easier and simpler compared to CLI. Besides, coding GUI provides higher level of control, hence it is more effective.

## 6.2 ANONYMOUS CLASS

Anonymous classes enable programmer to make their codes more concise where it allows the declaration and instantiation of a class at the same time (Oracle, 2015). When class is to be only used for once, it is better to make it an anonymous class. In BVCB, especially the event handler of the GUI is made anonymous class since it will only be used once by the same button every time.

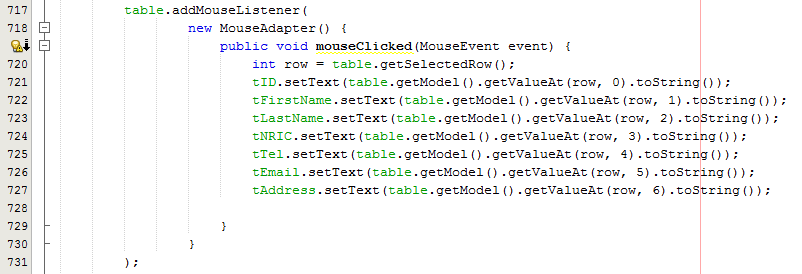


Figure : Anonymous Class Example in BVCB

## 6.3 ID GENERATOR

In order to make sure there is no two ID is the same for any objects in the BVCB, every class consists of a generateID( ) method where new and unique ID will be generated when the method is called.

## 6.4 ADD AND REMOVE EMPLOYEE

There is a functionality that allows the owner to add a new employee or remove any employee in the BVCB System.

## 6.5 JDATECHOOSER

JDateChooser is used in the Appointment and Boarding Booking Page where the receptionist is able to choose a particular date from the drop down calendar. Besides, JDateChooser has method that allows programmers to get the date from the JDateChooser as a Calendar object that is easy be handled and saved.

## 6.6 JTable

JTable is widely used in the BVCB System in order to display all the details of the necessary output. It is easy to populate the JTable with different fields of any object using an Enhanced For Loop. JTable can even be registered with a Mouse Listener so that user is able to get any value from the selected row.

# 7.0 CONCLUSION

In a conclusion, all of the requirements of the BVCB System is met where it covers all the functionality that has to be performed by the four major staff in the BVCB Veterinary Clinic which are the receptionist, boarding staff, vet and the owner. Besides, the BVCB System also meets the general requirement where it is written in Java with a pure Object Oriented approach. Moreover, several test run and input validation has been done on the system so that it has less bugs and is able to read and write objects into the binary files. Lastly, the system is considered as stable and easy to maintain as it is mainly written with Inheritance and Polymorphism concept.

# References

Chauhan, S., 2013. *Understanding Association, Aggregation, Composition and Dependency relationship.* [Online]   
Available at: http://www.dotnet-tricks.com/Tutorial/oops/T0Lb270813-Understanding-Association,-Aggregation,-Composition-and-Dependency-relationship.html  
[Accessed 17 August 2015].

Deitel, P. & Deitel, H., 2015. *Java How to Program.* 10th ed. London: Pearson.

Flanagan, D., 2005. *Java In A Nutshell.* 5th ed. s.l.:O'Reilly.

Horton, I., 2011. *Beginning Java.* 7th ed. Indianapolis: John Wiley & Sons, Inc.

koirala, S., 2012. *Understanding Association, Aggregation, and Composition - CodeProject.* [Online]   
Available at: http://www.codeproject.com/Articles/330447/Understanding-Association-Aggregation-and-Composit  
[Accessed 17 August 2015].

Lewis, J. & Loftus, W., 2004. *Java Software Solutions - Foundations of Program Design.* 4th ed. s.l.:Addison Wesley.

Malik, D., 2012. *Java Programming from Problem Analysis to Program Design.* 5th ed. Boston: Course Technology.

Oracle, 2015. *Anonymous Classes (The Java™ Tutorials > Learning the Java Language > Classes and Objects).* [Online]   
Available at: https://docs.oracle.com/javase/tutorial/java/javaOO/anonymousclasses.html  
[Accessed 17 August 2015].

Oracle, 2015. *What Is an Exception? (The Java™ Tutorials > Essential Classes > Exceptions).* [Online]   
Available at: https://docs.oracle.com/javase/tutorial/essential/exceptions/definition.html  
[Accessed 17 August 2015].

Rouse, M., 2008. *What is object-oriented programming (OOP)? - Definition from WhatIs.com.* [Online]   
Available at: http://searchsoa.techtarget.com/definition/object-oriented-programming  
[Accessed 17 August 2015].

Student Name:

**TOTAL MARKS**

ID:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **REQUIREMENT ANALYSIS** | **Fail** | **Marginal Fail** | **Pass** | **Credit** | **Distinction** |
| **0-3** | **4** | **5-6** | **7** | **8-10** |
| **USE CASE DIAGRAM**  **WITH**  **DESCRIPTION**  **(10%)** | Incorrect overview use case, detail use cases with scenarios (where applicable) and use case descriptions OR incorrect user case notations OR illogical functional design. | Inappropriate overview use case, detail use cases with scenarios (where applicable) and use case descriptions. | The use cases are well presented (generalization) with no major mistake in logic and notation, and described all flows in use case descriptions. | Appropriate labelling and no mistake in logic and notation and clear description for normal flow, subflow and alternative flow in use case descriptions**.** | Comprehensive provision of the required  - overview use case;  - detail use case diagrams  with scenarios (where applicable: generalization, extends, includes);  - use case descriptions for each use case / scenario and no mistake in logic and notation and appropriateness. |
|  |  |  |  |  |
| **ACTIVITY DIAGRAM**  **WITH**  **DESCRIPTION**  **(10%)** | The business-level activity diagram(s) is without swimlanes and not logical in an overall cases. | The business-level activity diagram(s) is without swimlanes and logical partially. | The business-level activity diagram(s) provided are complete, with swimlanes, logical and with no major errors. | With labelling, with swimlanes, and no mistake in logic and notation. | With very good and meaningful labelling, with swimlanes, no mistake in logic and notation, and according to guidelines. |
|  |  |  |  |  |
| **CLASS DIAGRAM**  **(10%)** | No attributes and no associations are included. | Class diagram with attributes and associations. Both are incomplete and illogical. | Class diagram with attributes and associations. Both are complete with no major errors. | Class diagram with appropriate attributes and associations. With labelling and no mistakes in logic and notation. | Class diagram with appropriate, relevant attributes and associations. With very good and meaningful labelling according to guidelines. |
|  |  |  |  |  |  |
|  |  |  |  | **SUB TOTAL:** |  |
| **IMPLEMENTATION** | **Fail** | **Marginal Fail** | **Pass** | **Credit** | **Distinction** |
| **0-3** | **4** | **5-6** | **7** | **8-10** |
| **APPOINTMENT AND BOARDING MANAGEMENT**  **(10%)** | No program or work not done. Program incomplete with obvious errors. Not able to compile and run the program. | Not able to compile or run but evidence of the coding is available. Able to compile but not able to run the program. Able to compile and run the program but only able to add less than 50% of the details of appointment and boarding management listed in the assignment. Not able to demonstrate the use of object-oriented concepts. Data not stored in file. | Able to compile and run the program. Able to add at least 50% of the details of appointment and boarding management listed in the assignment. Able to demonstrate the use of at least one of the object-oriented concepts – such as creation of classes and objects. Attempted to write to file but with some errors. | Able to compile and run the program. Able to add at least 70% of the details of appointment and boarding management listed in the assignment. Able to demonstrate the use of more than one object-oriented concepts. Able to write to file and read from file with some errors. | Able to compile and run the program. Able to add at least 80% of the details of appointment and boarding management. Able to demonstrate the use of more than two object-oriented concepts. Able to write to file and read from file with no errors. |
|  |  |  |  |  |
| **VET DIARY/SCHEDULE MANAGEMENT**  **(10%)** | No program or work not done. Program incomplete with obvious errors. Not able to compile and run the program. | Not able to compile or run but evidence of the coding is available. Able to compile but not able to run the program. Able to compile and run the program but only able to add less than 50% of the details of the vet schedule managment in listed in the assignment. Not able to demonstrate the use of object-oriented concepts. Data not stored in file. | Able to compile and run the program. Able to add at least 50% of the details of the vet schedule managment in listed in the assignment. Able to demonstrate the use of at least one of the object-oriented concepts – such as creation of classes and objects. Attempted to write to file but with some errors. | Able to compile and run the program. Able to add at least 70% of the details of the vet schedule managment in listed in the assignment. Able to demonstrate the use of more than one object-oriented concepts. Able to write to file and read from file with some errors. | Able to compile and run the program. Able to add at least 80% of the details of the vet schedule managment in listed in the assignment. Able to demonstrate the use of more than two object-oriented concepts. Able to write to file and read from file with no errors. |
|  |  |  |  |  |
| **REPORTING**  **(10%)** | No program or work not done. Program incomplete with obvious errors. Not able to compile and run the program. | Not able to compile or run but evidence of the coding is available. Able to compile but not able to run the program. Able to compile and run the program but only able to add less than 50% of the reports listed in the assignment. Not able to demonstrate the use of object-oriented concepts. Data not stored in file. | Able to compile and run the program. Able to add at least 50% of the reports listed in the assignment. Able to demonstrate the use of at least one of the object-oriented concepts – such as creation of classes and objects. Attempted to write to file but with some errors. | Able to compile and run the program. Able to add at least 70% of the reports listed in the assignment. Able to demonstrate the use of more than one object-oriented concepts. Able to write to file and read from file with some errors. | Able to compile and run the program. Able to add at least 80% of the reports listed in the assignment. Able to demonstrate the use of more than two object-oriented concepts. Able to write to file and read from file with no errors. |
|  |  |  |  |  |  |
| **REPORT STRUCTURE** | **Fail** | **Marginal Fail** | **Pass** | **Credit** | **Distinction** |
| **0-3** | **4** | **5-6** | **7** | **8-10** |
| **REPORT FORMAT**  **AND**  **REFERENCES**  **(10%)** | The reference list is *all inapplicable* OR *irrelevant*. The format is NOT comply proper referencing system. The citation is NOT included at all. The simple document without formatting, header and footer, page number, etc. | The reference list is identified *mostly inapplicable* OR *irrelevant*, The format is partially comply proper referencing system. The citation is partially included.  The simple document with little formatting. | The reference list is complete but sort of complete citation.  The document is complete. | The above +  The citation is *clearly specified*.  The above +  with all the formatting criteria. | The above +  with rich source of explorations to form a complete reference.  The above +  precise, clear, complete, all the diagram, chart, picture, pie, symbol, glossary are completely organized. |
|  |  |  |  |  |
| **PROGRAM DOCUMENTATION**  **(20%)** | Documentation not done.  Content of documentation does not adhere to any of the requirements stipulated in the assignment requirements. | At least 1 object-oriented programming concept applied in the solution and briefly described.  No implementation code for the object-oriented programming concepts identified.  Screen capture of the output of the program does not have any explanation to describe the program. | Description of at least 2 object-oriented programming concepts that are applied in the solution with some evidence of the implementation code is included.  Screen capture of the output of the program with minimal explanation to describe the program. | Description of at least 3 object-oriented programming concepts that are applied in the solution are described and evidence of the implementation code being documented.  Screen capture of the output of the program with some explanation to describe the program.  Description and evidence of at least 1 additional feature which has been incorporated in the solution. | Description of at least 4 object-oriented programming concepts that are applied in the solution and evidence of the implementation code being documented.  Screen capture of the output of the program with appropriate explanation to sufficiently describe the program.  Description and evidence of at least 2 additional features which have been incorporated in the solution. |
|  |  |  |  |  |  |
|  |  |  |  | **SUB TOTAL:** |  |

| **PRESENTATION** | **Fail** | **Marginal Fail** | **Pass** | **Credit** | **Distinction** |
| --- | --- | --- | --- | --- | --- |
| **0-3** | **4** | **5-6** | **7** | **8-10** |
| **INDIVIDUAL PRESENTATION**  **(10%)** | Absent OR late OR not prepared for presentation session without valid reasons. Handled questions grossly bad and unable to demonstrate understanding of OO concept. | Reading from presentation material. Presentation material is NOT well prepared. Presentation sequence NOT well organized and not smooth. Handled questions badly and unable to demonstrate understanding of OO concept. | Reading occasionally from presentation material.  Presentation material is well prepared. Presentation sequence acceptably organized and smooth. Handled questions well and demonstrated fundamental level of understanding of OO concept. | Good oral presentation. Presentation material is well prepared. Presentation sequence well organized and smooth. Handled questions well and demonstrated good understanding of OO concept. | Give an impactful presentation where the presenter delivers smooth oral presentation aided beautifully by well-prepared presentation material. Presentation sequence excellently planned organized and smooth. Handled questions well and demonstrated good understanding of OO concept. |
|  |  |  |  |  |