|  |  |
| --- | --- |
| **Practicum Case** |  |
| COMP6708  Object Oriented Programming |
| **Computer Science** | **O221-COMP6708-RV01-04** |
| ***Valid on*** *Odd Semester Year 2021/2022* | **Revision 00** |

## Learning Outcomes

* The main features of OOP
* The additional features of OOP
* Between conventional programming and OOP
* A program using additional features of OOP

## Topic

* Encapsulation

## Subtopics

* Concept (Why using encapsulation)
* Access Modifier
* Accessor and Mutator (Setter & Getter)

## Soal

*Case*

**BlueJack Vet**

**BlueJack Vet** is the new vet that has been the talk of people recently. In addition to their newest technology and their utmost care towards their services, they lacks the IT support to help them note down their medical data. You, as their most trustful freelancer programmer, are asked to help them build a very convenient app for that using **Java Programming Language** with Object-Oriented Programming concepts such as **Encapsulation**.

* In the beginning, the program will show the title



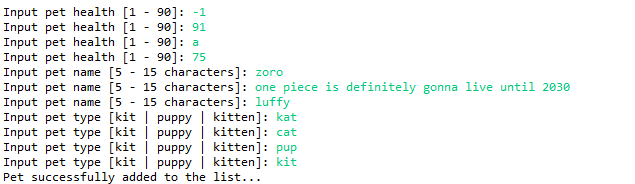
* The program will consist of **5 menus**:

1. Add Sick Pet
2. View Sick List
3. Update Pet Health
4. Treat Pet
5. Exit Program

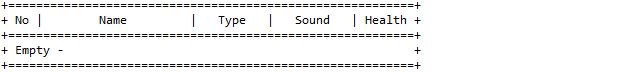
* The program will ask the user to input choose menu which must be **inputted between 1 and 5**



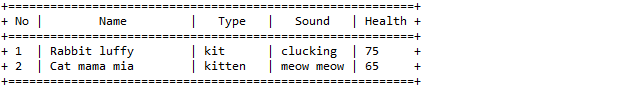
* If the user chooses menu 1, then:
* The program will ask the user to input pet detail, which consists of:
* **Health**, which must be **numeric** and between **1** and **90** (**inclusive**)
* **Name**, which **length** must be between **5** and **15** characters (**inclusive**)
* **Type**, which must be between **kit, puppy,** or **kitten** (**case sensitive**)
* After fulfilling all validation, the program will display a message



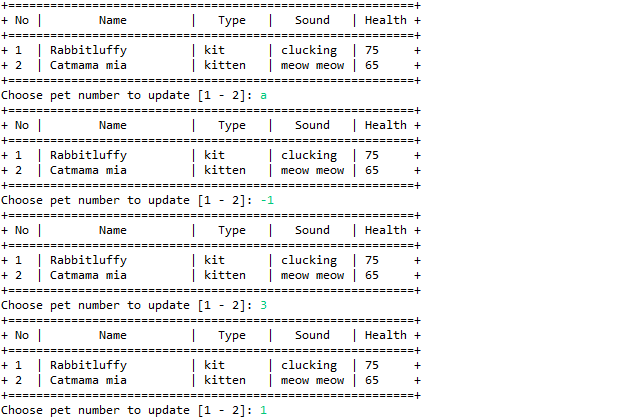
* After that, return to the main menu
* If the user chooses menu 2, then:
* If there are **no pet**, then display list with **empty** content



* Otherwise, display list of pet medical data
* **Name** is gotten from the result based on its type
* If the type is **kit**, append a string **“Rabbit”** at the start of its name
* If the type is **puppy**, append a string **“Dog”** at the start of its name
* If the type is **kitten**, append a string **“Cat”** at the start of its name
* **Sound** is gotten from the result based on its type
* If the type is **kit**, then display **“clucking”**
* If the type is **puppy**, then display **“bark bark”**
* If the type is **kitten**, then display **“meow meow”**



* After that, return to the main menu
* If the user chooses menu 3, then:
* If there are **no pet**, then return to the main menu
* Otherwise, display list of pet medical data
* Validate that choice option must be **numeric** and between **1** and the **current list of medical data**



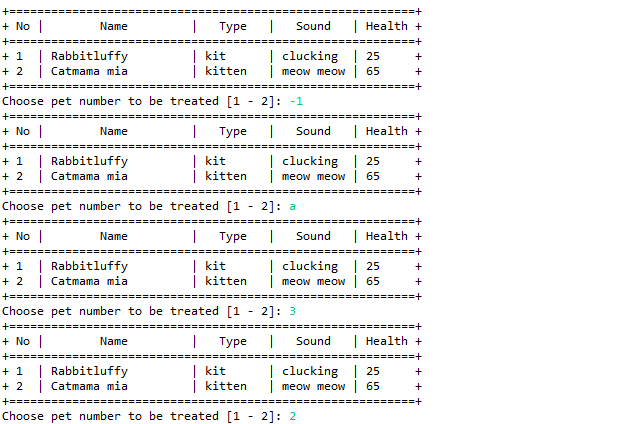
* Ask the user to input its updated health, which must be **numeric** and between **1** and **90**



* After that, update the corresponding pet health
* If the updated health is **below 20**, then set updated value to **25**
* If the updated health is **less than earlier** pet health, then the updated value is the **sum of earlier pet health and updated health**, **divided by 2**
* Finally, display a message indicating the corresponding pet data has been updated



* If the user chooses menu 4, then:
* If there are **no pet**, then return to the main menu
* Otherwise, display list of pet medical data
* Validate that choice option must be **numeric** and between **1** and the **current list of medical data**



* Finally, display a message indicating the corresponding pet has been treated and **remove** it from the list



* If the user chooses menu 5, then:
* Display a message of gratitude
* Exit the program



**Please ask your teaching assistant if there are any related questions.**