Software Engineering and Programming Basics

Veterinary Clinic Simulation

Description of Classes

1.Person

Person class serves as the base class for “Doctor” and “Assistant” which represents attributes of a person like Name,Age and Address.

2.Doctor

Doctor class inherits from Person class.it represents Doctor in the Veterinary System who handles specific kind of Animals.Manages authorized animals for treatments.

3.Assistant

Assistant class also inherits basic information from Person.It represents Assistant in Veterinary who supports Doctor for smooth treatment process.

4.Animal

Represents the patient in the veterinary system which contains attributes such as name, age, weight, and preexisting conditions. Interacts with the “Case “ class for diagnosis and treatment.Also checks if certain Animal kind is valid or not and adds it to Animal Class only if it is valid.

5.Case

Represents a medical case, connecting Person (as the owner), Animal, and Treatment. It binds the entire scenario of a patient's visit to the clinic.

6.Node

A building block for the linked list in the List class.Also stores the information of a Case.

7.List

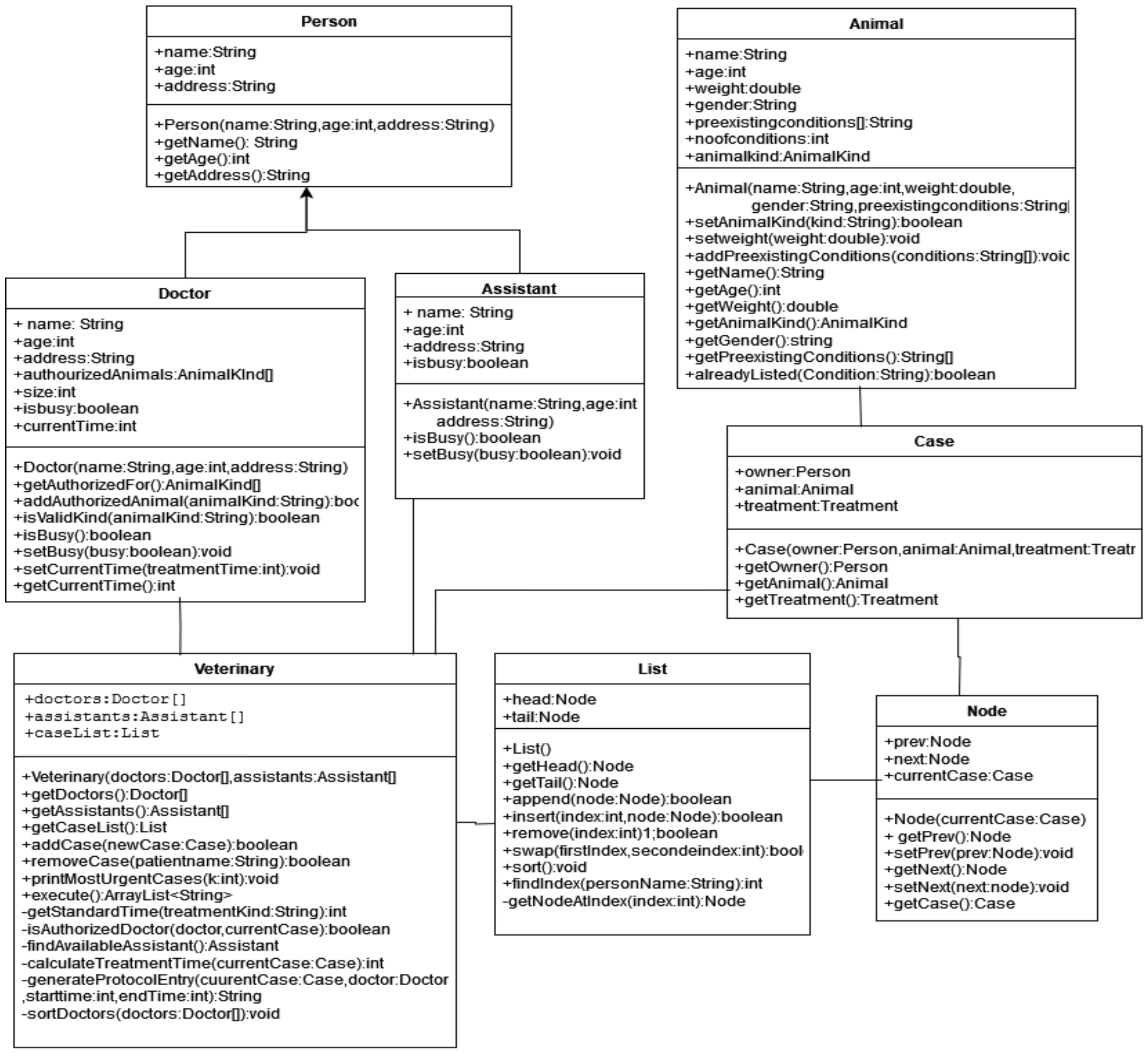
Manages cases using a doubly linked list structure. Supports operations like adding, removing, and sorting cases. Interacts with Node to maintain the order of cases.

8.Veterinary

This class manages overall functioning of Veterinary Systems.Contains information of Doctors,Assistants and Cases.It registers the cases and adds them to a queue .Then executes Cases in the queue for a certain amount of time one by one with help of authorized Doctors and Assistants.

How Classes interacting each other

- Overall functioning such as Case Registration,Adding to the queue and Executing the case with help of Doctors and Assistants can be done in “Veterinary Class” by interacting with other classes.
- “Doctor” and “Assistant” classes instances are created and given as an array to “veterinary” class.
- Entire Patient information is contained in Case class instances where it interacts with “Person” class for owner information and “Animal” class for Patient Information and Case Registration is done in Veterinary Class.
- Each Case information is stored in “Node” and then Nodes are added to List
- Adding Cases to Queue can be done Using “List” Class and also Prioritizing emergency cases can also be done in List.
- Treatment Execution will be done in “Veterinary” class with the help of Doctors and Assistants.


Person
+name:String
+age:int
+address:String
+Person(name:String,age:int,address:String)
+getName(): String
+getAge():int
+getAddress():String
Animal
+name:String
+age:int
+weight:double
+gender:String
+preexistingconditions[]:String
+noofconditions:int
+animalkind:AnimalKind
+Animal(name:String,age:int,weight:double,
gender:String,preexistingconditions:String
+setAnimalKind(kind:String):boolean
+setweight(weight:double):void
+addPreexistingConditions(conditions:String[]):voic
+getName():String
+getAge():int
+getWeight():double
+getAnimalKind():AnimalKind
+getGender():string
+getPreexistingConditions():String[]
+alreadyListed(Condition:String):boolean
Doctor
+ name: String
+age:int
+address:String
+authourizedAnimals:AnimalKind[]
+size:int
+isbusy:boolean
+currentTime:int
+Doctor(name:String,age:int,address:String)
+getAuthorizedFor():AnimalKind[]
+addAuthorizedAnimal(animalKind:String):boc
+isValidKind(animalKind:String):boolean
+isBusy():boolean
+setBusy(busy:boolean):void
+setCurrentTime(treatmentTime:int):void
+getCurrentTime():int
Assistant
+ name: String
+age:int
+address:String
+isbusy:boolean
+Assistant(name:String,age:int
address:String)
+isBusy():boolean
+setBusy(busy:boolean):void
Case
+owner:Person
+animal:Animal
+treatment:Treatment
+Case(owner:Person,animal:Animal,treatment:Treatr
+getOwner():Person
+getAnimal():Animal
+getTreatment():Treatment
Veterinary
+doctors:Doctor[]
+assistants:Assistant[]
+caseList:List
+Veterinary(doctors:Doctor[],assistants:Assistant[]
+getDoctors():Doctor[]
+getAssistants():Assistant[]
+getCaseList():List
+addCase(newCase:Case):boolean
+removeCase(patientname:String):boolean
+printMostUrgentCases(k:int):void
+execute():ArrayList<String>
-getStandardTime(treatmentKind:String):int
-isAuthorizedDoctor(doctor,currentCase):boolean
-findAvailableAssistant():Assistant
-calculateTreatmentTime(currentCase:Case):int
-generateProtocolEntry(cuurentCase:Case,doctor:Doctor
,starttime:int,endTime:int):String
-sortDoctors(doctors:Doctor[]):void
List
+head:Node
+tail:Node
+List()
+getHead():Node
+getTail():Node
+append(node:Node):boolean
+insert(index:int,node:Node):boolean
+remove(index:int)1;boolean
+swap(firstIndex,secondeindex:int):bool
+sort():void
+findIndex(personName:String):int
-getNodeAtIndex(index:int):Node
Node
+prev:Node
+next:Node
+currentCase:Case
+Node(currentCase:Case)
+ getPrev():Node
+setPrev(prev:Node):void
+getNext():Node
+setNext(next:node):void
+getCase():Case

Attributes Used in Each Class

Person:

String name,int age,String address → these attributes are to store basic information of a person.

Doctor:

String name,int age,String address → these attributes are to store basic details of a doctor

AnimalKind[] authourizedAnimals → to store authorized animal kinds to an array

int noofauthorizedAnimals → to keep track of how many animals added to the list so that the next animal can be added at the next index.

boolean isbusy → to set busy status of a doctor

int currentTime → to set the currentTime of a doctor(to keep track of cases doctor dealing with)

Assistant:

String name,int age,String address → to Store basic details of an Assistant

boolean isbusy →to store the busy status of an Assistant.

Case:

Person owner → to store owner information

Animal animal → to store patient information

Treatment treatment → to store treatment kind

Animal:

String name,int age,double weight,String gender → these attributes hold the basic information of an Animal.

String[] preexistingConditions → this array stores the pre-existing conditions of an animal.

int noofConditions → it stores the number of conditions added to the array so that next condition will be added at the next index.

AnimalKind animalkind → to store new animal kind.

Node:

Case currentCase → to store information of a case.

Node prev → stores previous Node information.

Node next → stores next Node information.

List:

Node head → to store head node information.

Node tail → to store tai node information.

Veterinary:

Doctor[] doctors → to store all the doctors in the veterinary system.

Assistant[] assistants → to store all assistants in the veterinary systems.

List caseList → to add registered cases to the queue.

Additional Methods

boolean isValidKind(String animalKind)

- This Method can be found in **"Doctor"** Class.It receives animalKind as String and checks if it is valid or not. If animalKind is valid it returns true else returns false.

boolean isBusy()

- This Method can be found in **"Doctor**" Class.It returns true if doctor is busy(currently treating case) else it returns false.Basically to check busy status of a particular Doctor.
- This Method can also be found in **"Assistant**" Class.it returns the busy status of a particular Assistant.

Void setBusy(boolean busy)

- This Method can be found in **"Doctor"** and **"Assistant"** Classes. It manually sets the busy status of a particular Doctor and Assistant.

Void setCurrentTime()

- This method can be found in **"Doctor**" class.It adds treatment time of a case doctor currently dealing with so that next treatment starts at the next minute.It basically sets the starting time of the next case that particular doctor deals with.

int getCurrentTime()

- This method can be found in **"Doctor"** class.It returns the currentTime of a doctor that is basically the starting time for the next case of a particular Doctor.

boolean isAlreadyListed(String Condition)

- This method can be found in **"Animal"** class.It will check if a particular preexisting condition is already listed in Preexisting conditions of an animal.

Node getNodeAtIndex(int index)

- This method can be found in **"List"** class.This method will returns Node at the given index

boolean isAuthourizedDoctor(Doctor doctor,Case currentCase)

- This method can be found in **"Veterinary**" class.It will check if a given doctor is authorized to treat a given case.

int getStandardTime(String treatmentKind)

- This method can be found in **"Veterinary"** class.It returns standard treatment time of a given treatment kind.

Assistant findAvailableAssistant()

- This method can be found in **"Veterinary"** class.It returns an available Assistant by checking the busy status of an Assistant.

int calculateTreatmentTime(Case currentCase)

- This method can be found in **"Veterinary"** class.It returns treatment time of a particular case.It actually returns standard treatment time if more number of assistants present than doctors in anteroom else it returns double of standard time.

String generateProtocolEntry(Case currentcase,Doctor doctor,int startTime,int endTime)

- This method can be found in "Veterinary" class.It generates a String for executed cases as below "$ownerName with $animalName was treated by $doctorName, $treatment, started at: $startingTime ended at: $finishingTime" .

Void sortDoctors(Doctor[] doctors)

- This method can be found in "Veterinary" class.It will sort the list of doctors based on their currentTime(whoever has less previous case treatment time will be moved to first).

Implementation of Methods

Animal.addPreexistingCondition

- This method receives array of conditions as input then iterating through conditions will check if each condition is already listed by comparing condition with all the conditions that already existing in PreexistingConditions array which is attribute of this class.If it is not already listed then I have added that particular condition to PreexistingConditions at the index of preexistingCondition array length.then I have incremented the length so that next condition will be added at that position.

List.swap

- This method receives two indices to be swapped as input.so first will get the Nodes at their indices.If one of the node is head or tail then updated the head or tail to other nodes.then first I have swapped the next pointers of the both nodes.then updated the previous pointers of next nodes(only if next node is not null and if it is null no changes).After that I have swapped previous pointers and then updated next pointers of previous nodes(only if previous node is not null and no change if it is null).

List.sort

- In this method I have created two separate linked list for Emergency and Non emergency cases first.then iterating through the original caseList I have disconnected each node from original caseList and then compared case's treatment kind with emergency.If it returns true then i appended it to the Emergency list.if it is not emergency then I appended it to Non Emergency list
- When original caseList is empty then I have reconnected Emergency and Non emergency list and updated head and tail pointers.If there are Emergency cases, then the head is set to Emergency list's head and tail is set to the tail Non Emergency list.If there are no emergency cases then head is set to head of Non Emergency list.

Veterinary.printMostUrgentCase

- This method receives the parameter k which indicates the number of most urgent case to print.First I have created the copy of original caseList as Urgentcases then I have sorted the Urgentcases list using sort method in list.So now sort method ensures that all urgent cases are moved to first.
- Now iterating through the Urgencases list until index k,I have printed "owner's name,animal's name,treatmentkind,standard treatment time" for each case.In case, given input is negative then I printed the message "invalid index".

Veterinary.execute

- First I have created an Arraylist of Strings to store the protocols of executed cases.After that since emergency cases has to take priority over other cases, I have sorted the caseList using sort method in list.Then iterating through the caseList and then iterating through doctors array,if currentcase is not equal to null then I have calculated treatmentTime for that particular case by calling method caluclateTreatmentTime (which returns standard treatment time of a case if more assistants than doctors present in anteroom otherwise it returns double of standard treatment time).
- Now it will check if the doctor is authorized to treat that particular case and if he is busy,only if both conditions are true then that case is assigned to this doctor otherwise will get the next doctor and checks until those conditions are true then that particular case will be assigned to that particular doctor and sets doctor"s busy status to true.In case,first doctor is both available and authorized for first case it is assigned to him and then will get next case and repeat the process.
- Since a doctor also needs an assistant will find an available assistant by calling method findAvailableAssistant. And if an assistant is also available then the doctor treats the case. For that I have generated a protocol entry "$ownerName with $animalName was treated by $doctorName, $treatment, started at: $startingTime ended at: $finishingTime" like this.then I added it to the ArrayList "Protocol". Since the doctor is done with the current case I have changed his busy status to false and then updated his currenttime (current time+treatment time+1) so that his next case will start from that current time. And then removed that case from the original caseList since it is executed.
- After each iteration through the doctors array, I have sorted the doctors based on their current time(previous cases treatment time).So that whoever finished the previous case first will be available first for next cases.
- This process will repeat until all the cases are executed in caseList. When the case list is empty it will return the Arraylist protocol.

Name:Saipriya Shagam

/ Matrikel Number:812012