Veterinary Simulation

Attribute Explanations

Class	Attributes	
Person	String name	Represents the name of the person
	int age	Represents the age of the person
	String address	Represents the address of the person
Animal	String name	Represents the name of the animal undergoing treatment.
	int age	Represents the age of the animal
	double weight	Represents the weight of the animal
	String gender	Represents the gender of the animal
	String[] preexistingConditions	Represents the initial conditions of the animal identified during the first screening.
	AnimalKind animalKind	Identify the kind of the animal
Treatment (Enum)	Represents the different types of treatments available at the veterinary clinic. vaccination, injury, diagnostics, emergency	
AnimalKind (Enum)	Represents the various kinds of animals that can be treated at the veterinary clinic. dog, cat, bird, reptile, rodent	
Case	Person owner	Details of person who is animal's owner
	Animal animal	Details of animal
	Treatment treatment	Identified treatment case
Node	Case currentCase	Represents the actual data in the case associated with the node.
	Node next	Represents the next node in the double-linked list
	Node previous	Represents the previous node in the double-linked list
Doctor	AnimalKind[] authorizedFor	Represents an array of animal kinds that the doctor is authorized to treat. This attribute defines the scope of the doctor's expertise.
	int operationTime	Represents the remaining operation time for the doctor. It is a countdown timer for ongoing treatments. When 'operationTime' equal 0, the doctor is ready to treat next case.
	int startingTime	Represents the timestamp when the doctor starts a treatment. Providing a timeline of events.
	int finishedTime	Represents the timestamp when the doctor finishes a treatment. Providing a timeline of events.
	Node handleCase	Represents the current case assigned to the doctor for treatment.

Prepared by : Panatda Pongkasemwiwat , Matriculation no.813796, Automotive Software Engineering's student 01.02.2024

Class	Attributes	
List	Node front	For accesss through the starting node in the list
	Node tail	For accesss through the tail node in the list
	int size	Represents numbers of node in the list
Assistant	(No additional attributes)	Inherited from its parent class from 'Person'
Veterinary	Doctor[] doctors	Represents an array of doctors available in the veterinary clinic. These doctors are ready to treat cases when the 'execute' method is executed.
	Assistant[] assistants	Represents an array of assistants available to participate in the veterinary clinic. Similar to doctors
	List queue	Represents a list containing all cases that have registered at the veterinary clinic and are awaiting treatment.

Method Explanations

❖ Animal.addPreexistingCondition():

Adds an array of Strings to the existing preexistingConditions array. The method checks for repeated data to avoid duplicates and determines the new size of the array.

1. Checks the original array to determine the length of preexistingConditions.
2. Compares each data between the added conditions and the existing preexistingConditions array, using the isRepeated() method to identify repeated data and calculate the new size of the array (newConditionsSize).
3. Creates a new array, newConditions, to store all target condition names. Copies the original data and the added conditons data sequentially into newConditions array.
4. Updates the preexistingConditions array with the newConditions.

❖ List.swap():

Swaps the positions of two nodes in the doubly linked list.

1. Checks for invalid index; returns false if either index is negative, exceeds the list size, or if both index are equal.
2. Get nodes corresponding to the provided indexes by getNodeAtIndex() method and ensures both nodes have valid values.
3. Handles special cases when one or both nodes are at the list's edges. (front and tail node)
4. Performs the swap:
   - Updates next pointers of the swapping nodes.
   - Updates previous pointers of nodes next to the swapping nodes.
   - Updates previous pointers of the swapping nodes.
   - Updates next pointers of nodes previous to the swapping nodes.

Prepared by : Panatda Pongkasemwiwat , Matriculation no.813796, Automotive Software Engineering's student 01.02.2024

❖ List.sort():

Sorts the list, prioritizing emergency cases at the front while maintaining the relative order of non-emergency and emergency cases.

1. Utilizes two nested loops for checking treatment types, with the inner loop reducing one iteration on each pass due to swaps completed at the end of node in each iteration.
2. Gets treatments from the adjacent nodes using List.getNodeAtIndex() and stores them in frontCompare and backCompare.
3. Swaps the positions of nodes only one condition if frontCompare is not an emergency case and backCompare is an emergency case.

❖ Veterinary.printMostUrgentCases():

Prints information about the most urgent cases up to a specified index.

1. Checks for an invalid index and returns informative messages to interact with the user in case the index is out of bounds.
2. Sorts all queue cases in the list by List.sort() method (prioritize emergency case)
3. Uses for loop for printing detail in each node, get the detail from the node by the Node.displayUrgentCase() method, including treatment time by using the calculateTreatmentTime() method.

❖ Veterinary.execute():

Simulates the execution of treatments submitted in the queue, considering factors such as doctors' authorizes, the number of doctors and assistants, and treatment types.

1. Creates an ArrayList named veterinaryResults to store all concluded cases.
2. Sorts all cases in the queue using the List.sort() method, prioritizing emergency cases.
3. Initializes veterinaryTime to 0; increments with each completed task across all doctors.
4. Employs a while loop for operations until there are no cases in the queue and finishedStatus is true.
5. Checks the operationTime of each doctor:

   - Condition 1: If a doctor has finished treatment (operationTime == 0) and there are cases in the queue, finds the next case by checking the doctor's authorization using the findNextCase() method. Registers the matched case, records case details, startingTime, submits the duration of the operation, and removes the case from the queue by using the registerDoctorCase() method.

   - Condition 2: If a doctor is about to finish a case (operationTime == 1), stamps veterinaryTime as the finished time, concludes the overall information of treatment using the concludeCase() method, and decreases the doctor's operationTime.

   - Else condition: Decreases the doctor's operationTime.
6. If there are no more cases in the queue, verifies that all doctors in the veterinary have completed their cases and then updates finishedStatus to be true.
7. Increase veterinaryTime
8. Returns the ArrayList veterinaryResults containing all concluded cases.

Prepared by : Panatda Pongkasemwiwat , Matriculation no.813796, Automotive Software Engineering's student 01.02.2024

Additional Method Explanations

Class	Additional methods
Animal	**private boolean isRepeated(String[] array , String str)** Checks whether a given string represented by str, is repeated within the provided array of strings names array. Returns true if the str is found each element in the array. If no repetition is detected, it returns false. Utilizing in the addPreexistingCondition() method for avoiding duplication.
Node	**public String displayUrgentCase()** Gets details, including the animal's owner name, animal name and treatment type, from the current node into the correct string pattern which is returned string massage to be utilized by the Veterinary.printMostUrgentCases() method
List	**public void insertHead (Node node)** Adds the received node to the beginning of the doubly linked list. 1st condition adds while list is empty 2nd condition adds while list already contains nodes
	public boolean removeNode (Node node) Removes the received node from the list and returns true. If the received node is empty or list is empty, returns false.
	private Node getNodeAtIndex (int nodeIndex) Gets the node and returns from the received index, which is meaning beginning node is equal 0, with the index starting at 0.
Doctor	**Private boolean isValidAnimalKind (String animalKind)** Checks the received animalKind is valid base on the AnimalKind enum. Returns true if the animalKind is valid, returns false if the animalKind is not valid.
Veterinary	**private boolean haveVeterinaryAuthorize (Case newCase)** Check whether the received case can be matched with the authorizations of all doctors in the veterinary. Retruns true if at least one doctos has authorization for treating; otherwise, returns false.
	private int calculateTreatmentTime (String treatmentType) Return the standardTime for treatment based on different treatment types and number of doctors and assistants available. If the number of assistants is equal or less than the number of doctors, standardTime is doubled.
	private boolean isDoctorAuthorized (Doctor doctor, Node node) Check whether the received case can be matched with the authorizations of the received doctor. Retruns true if doctor has authorization for treating; otherwise, returns false.
	private String concludeCase (Doctor doctor) Creates and returns a formatted string containing details about a concluded case. Formating of caseConclusion is "$ownerName with $animalName was treated by $doctorName, $treatment, started at: $startingTime ended at: $finishingTime" Notes: The variables starting with a $ indicating the variables that will be replaced with specific values of each case.
	private void registerDoctorCase (int time, Node currentCase, Doctor doctor) Registers the currentCase, startingTime and treatmentTime into the doctor who is going to treat this case. And removes this case from veterinary waiting queue. The treatmentTime is definded by calculateTreatmentTime() method.
	private Node findNextCase (Doctor doctor) Finds the next case for the received doctor based on the doctor's authorization. This method iterates through the cases in the veterinary's queue and returns the first case that matches by using the isDoctorAuthorized() method. If no matching case is found, null is returned.

Prepared by : Panatda Pongkasemwiwat , Matriculation no.813796, Automotive Software Engineering's student 01.02.2024