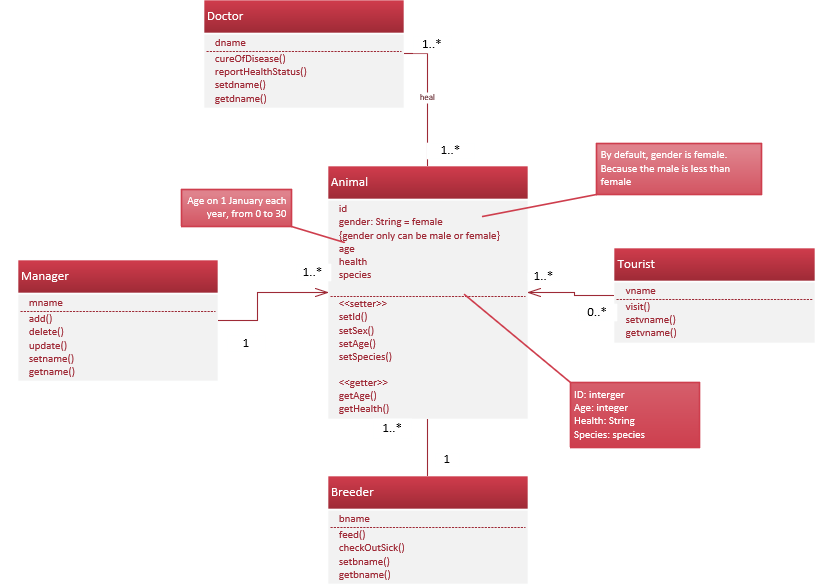
Zoo Management

**Description and Requirements**

Zoo management requires the management of animals, tourists, breeders, animal accommodation, animal doctors and zoo managers. Among them, the animals' eating is controlled by the breeders, and the animals' sicknesses such as colds are detected by the breeders and the animal doctors are called to see and treat them. The zoo manager has a statistical table of all the animals in the zoo, including their ID, age, gender, residence, weight, and health status. They can be added (introduced or born), deleted (aged out), and queried. Each animal has a breeder who manages the fixed daily meal, rest, activity, and visiting hours for the animals under his or her management. Tourists need tickets to enter and exit the zoo.

**Class diagram**

****

|  |  |
| --- | --- |
| class name | meaning |
| Animal | The most prominent resident of the zoo for tourists. There are attributes such as number, age, string, health status, species, and ID. |
| Tourist | Visit the zoo. |
| Manager | Manages all matters of the zoo and is the user of the system. |
| Breeder | Manage animal accommodation. |
| Doctor | To see the animals. |

-Properties

Number: numeric (integer)

gender: string

Age: numeric (integer)

Health: numerical (integer)

Species: string

id: numeric (integer)

-Operation

Set the number setnumber()

Get number getnumber()

Set health status sethealth()

Get health gerhealth()

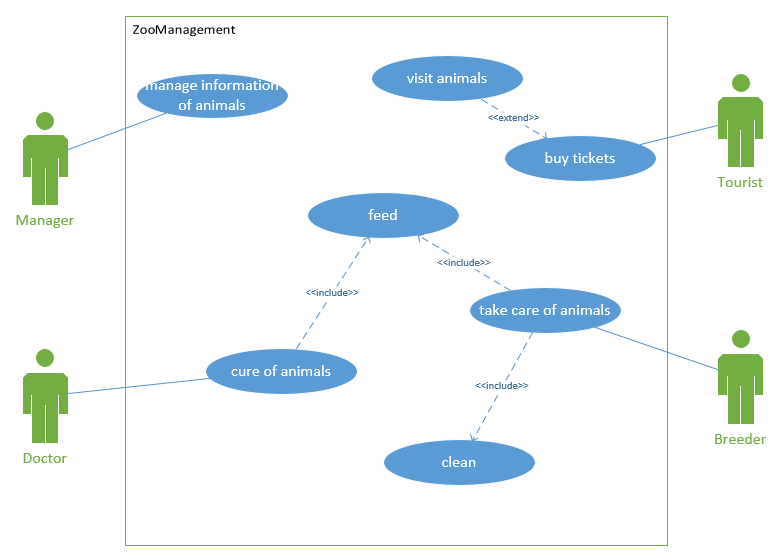
Set age setage()

Get age getage()

Set the ID setid

**Use case diagram**

With the requirements already in place, an initial understanding of the functionality to be accomplished by the system is given. The use case diagram is given below.



**Identification of participants**

Animals, tourists, breeders, animal doctors, zoo managers

**Capturing and describing requirements**

Use Case Name: Feeding Performed by: feeder Purpose: To complete the feeding of animals.

Use Case Name: Visit Performer: Tourist Purpose: Visit animals

Use case name: Seeing a doctor Performer: Animal doctor Purpose: To treat animals and keep them healthy

Use Case Name: Add Implementer: Manager Purpose: To complete the management of animals throughout the zoo

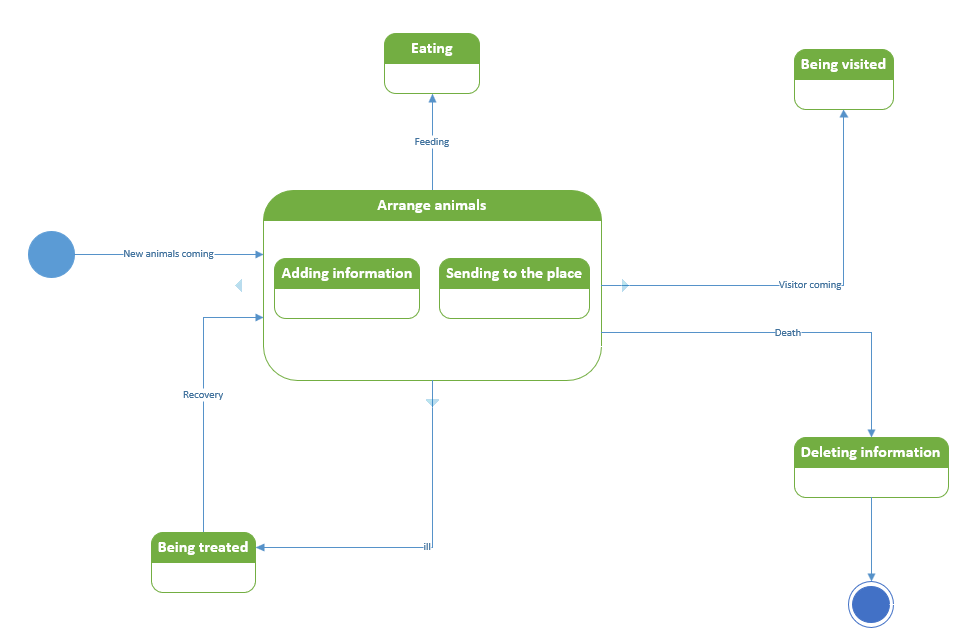
Use Case Name: Delete Executed by: Manager Purpose: To complete the management of animals throughout the zoo

Use Case Name: Update Implementer: Manager Purpose: To complete the management of animals throughout the zoo

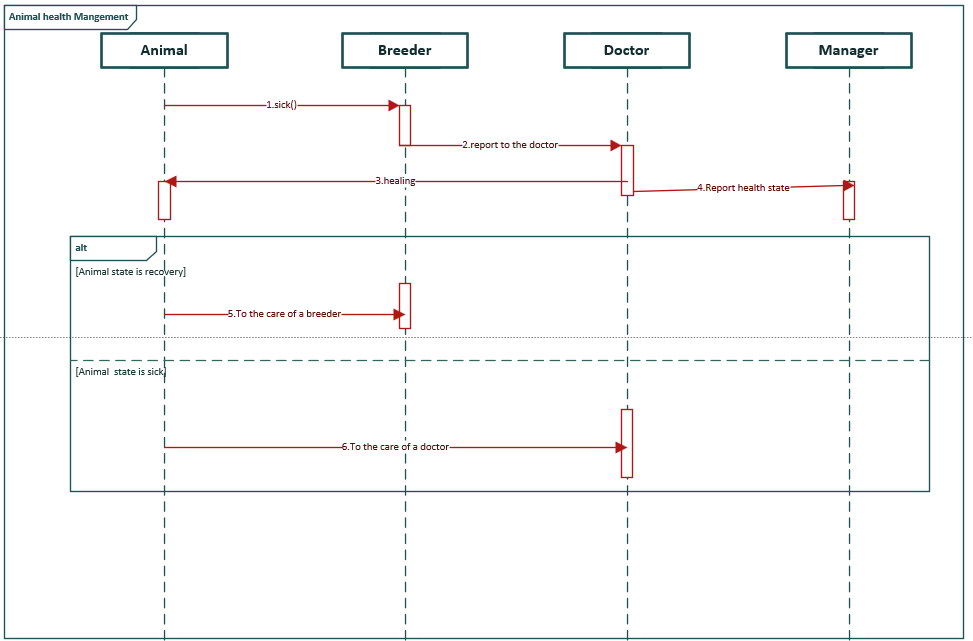
|  |  |
| --- | --- |
| use case name | See a doctor |
| Overview of use cases | This use case describes an animal management system in which an animal doctor treats an animal based on its health status. |
| Participants. | Animal Doctor |
| Pre-Conditions | Animal sickness |
| Post-Conditions | The doctor reports the health status of the animal to the manager. |
| event stream |  |
| Basic event flow | 1. The breeder hands over the sick animal to the animal doctor. 2. Animal doctors see animals. 3. Physician to determine animal health 4. Animals are temporarily housed in a doctor's care facility while they are sick. 5. The doctor takes care of the animals 6. The doctor feeds the animal medicine. 7. The doctor reports the health status of the animal to the manager. 8. Managers record animal information. 9. Animal restoration of 10. The animals are returned to their homes from new and are watched by breeders. |

**State Diagram**

A state machine diagram represents the life history of an object (class). A state machine diagram should be drawn for some objects that implement important behavioral actions. Drawing a state machine diagram requires determining the full range of possible states that can occur during the lifetime of an object, which events will cause a transfer of state, and which actions will occur.



### Sequence diagram



**Activity diagram**

The main role of activity diagrams is to represent the business workflows and concurrent processes of the system. Activity diagrams are drawn for the main business workflows of the self-selecting system.

Mapping activities requires identifying the objects involved in the activity, the action state, the action flow, and the object flow.

