

Object oriented programming

Lab 5 - 6 extra exercises

Dear Pet Lover!

People who love animals tend to be great people. I really hope you are one of these people and you not only love pets but also treat them well.

This extra exercise will be a much larger one but you can work on it during the 5 and 6 labs as well.

In this practice you need to manage owners and their pets, medical centers and doctors belonging to these and overall you need to model a pet hospitality system.

During the implementation feel free to use the **hints** and the **useful links** from the end of the document.

The project structure should be similar to this:

date

- MyDate (from your previous lab)

interfaces

- IDoctor
- IMedicalCenter
- IPatient

medCenters

- Veterinary

people

- Person
- Owner
- Veterinarian

pets

- Pet
- Dog
- Cat
- Parrot
- Turtle

utils

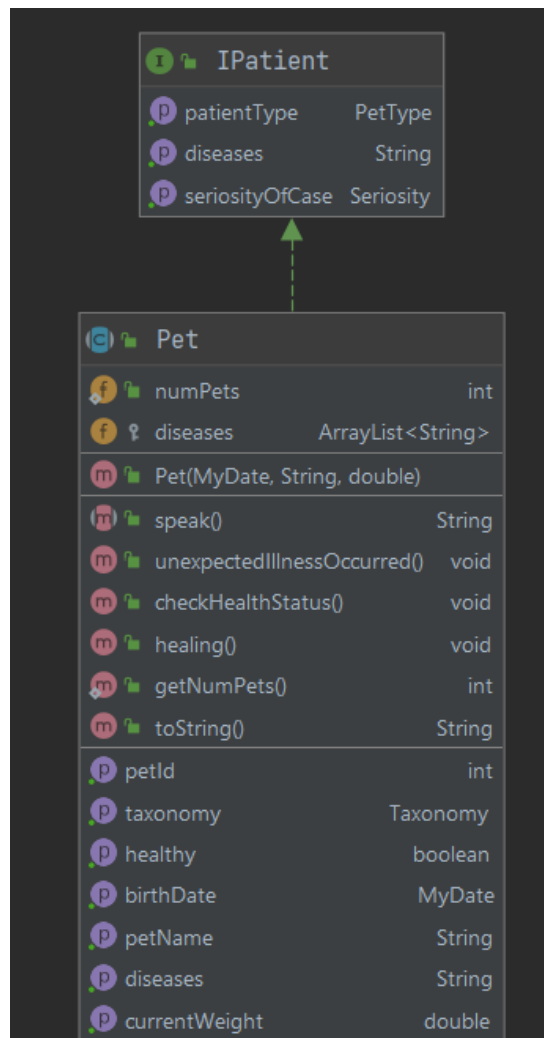
- DateUtil (from previous lab)
- Enums
 - *BodyType*
 - *CatBreed*
 - *DogBreed*
 - *Gender*
 - *PetType*
 - *Seriosity*
 - *Taxonomy*
 - *TurtleClassification*
- Util
 - *Disease*

Object oriented programming

Lab 5 - 6 extra exercises

Pets:

Each **pet** is characterized by an *ID*, *birthdate*, *name*, *taxonomy* (e.g. *mammal*), *current weight*, and *health status*. These characteristics apply to every pet (Cat, Dog, Parrot, Turtle). Be sure that we cannot create instances of a simple Pet. Furthermore, unfortunately every pet can get sometimes ill. You should store these in the *diseases* list. Because of possible sickness each pet can **become a patient (IPatient interface)**. Take in consideration all actions relevant for a patient (If this is too hard for you, just leave out the **IPatient** interface).



1. figure: Pet class

Object oriented programming













Lab 5 - 6 extra exercises

Methods:

- **Constructor:** You should specify *the birthdate, name and current weight* of your pet. The *health status* should be true (which means your pet is healthy). Assign a *unique pet id* for each pet (use the *number of pets* static field to ensure this).
- **speak:** Each pet can speak differently
- **unexpectedIllnessOccurred:** Unfortunately, sometimes our beloved pets can get sick. If you know the specific diseases for a given pet type assign randomly (one or more) disease(s) to a pet. Use the **Utils** and **Disease** classes to store all types of diseases by pet.
- *Hint (an example to assign randomly a disease to a Dog):*

```
int randomIllness =
    Util.random
        .nextInt(Util.getNumOfDiseasesByPet(PetType.Dog) - 1);

this.diseases
    .add(Util.getDiseasesByPetType(PetType.Dog)
        .get(randomIllness)
        .getDisease());
```

		Disease	
		treatments	ArrayList<String>
		Disease(String, PetType)	
		addTreatments(List<String>)	void
		toString()	String
		petType	PetType
		disease	String

2. figure: Disease class

Object oriented programming

Lab 5 - 6 extra exercises

- If a pet gets sick (new disease is added to the list) you should **modify the health status** of the pet)

```
this.isHealthy = false;
```

- Implement some extra methods which helps you to store and get a specific disease or treatment.

E.g.

```
public static void readDiseasesAndTreatmentsFromFile(String filename)
public static void printAllDiseasesByPetType(Enums.PetType type)
public static int getNumOfDiseasesByPet(Enums.PetType type)
public static String treatmentsByDiseaseType(String... diseaseType)
```

The content of the **diseases.csv** you can find on the link below:
https://moodle.ms.sapientia.ro/pluginfile.php/20084/mod_folder/content/0/diseases.csv?forcedownload=1

E.g. for a row in the CSV file:

Dog, Dental Disease, teeth cleaning, extractions, root canal

Where the first word means the **type of pet**, the second the **disease type** and the rest of the words the **recommended treatments**.

- **checkHealthStatus**: You can always check the current health status of a pet. If the pet is healthy write to the standard output a similar message to this:

```
Tim is healthy
```

Where Tim is the name of the pet.

If your pet is not healthy print all diseases from which he/she is suffering:

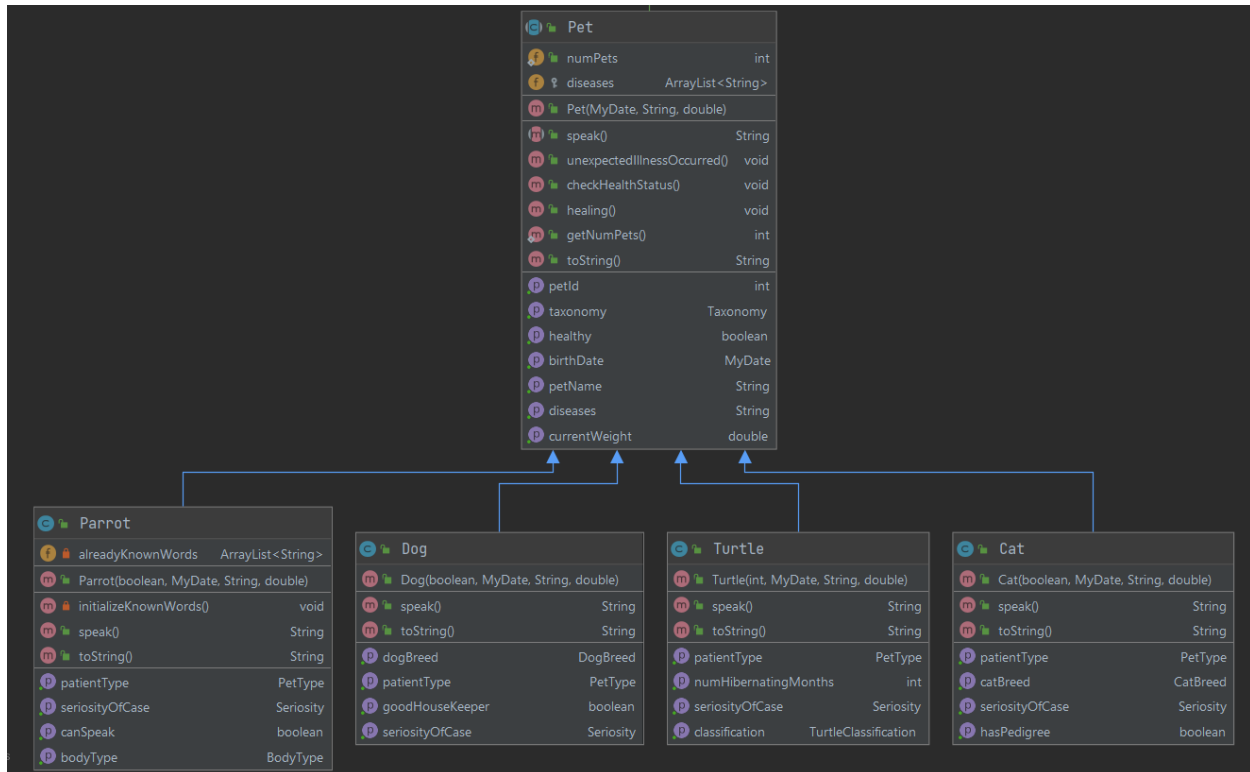
Object oriented programming

Lab 5 - 6 extra exercises

Lizzie is suffering from the following diseases:
- Stiffness and Pain

- **healing**: This method helps the pet to heal and become healthy again (this should be used after a veterinarian treatment).

Create the more specific pet classes: **Cat, Dog, Parrot, Turtle**



3. figure: Pet class and other specific pet types

Extra characteristics in case of a:

- Cat: **catBreed**, **hasPedigree**
 - Besides the common properties of a pet you need to initialize the **hasPedigree**, **taxonomy** (in this case: MAMMALS) properties and you also need to assign a random **breed type** (based on the CatBreed enum)
 - *speak method*: A cat should speak like: *Miau*
- Dog: **dogBreed**, **goodHouseKeeper**

Object oriented programming

Lab 5 - 6 extra exercises

- Besides the common properties of a pet you need to initialize the **goodHouseKeeper**, **taxonomy** (in this case: MAMMALS) properties and you also need to assign a random **breed type** (based on the DogBreed enum)
- **speak** method: A dog should speak like: Wuff-wuff
- Parrot: **canSpeak**, **alreadyKnownWords**, **bodyType**
 - Besides the common properties of a pet you need to initialize the **classification** (randomly based on the TurtleClassification enum), **numOfHibernatingMonths** and **taxonomy** (in this case: BIRDS) properties. If the parrot can speak you need to populate the **alreadyKnownWords** list with some items from the following file: https://moodle.ms.sapientia.ro/pluginfile.php/20084/mod_folder/content/0/commonWords.txt?forcedownload=1
 - **speak** method: When a parrot is speaking it should say all known words
 - **Hint:**

```
private void initializeKnownWords() {
    if(Util.commonWords.isEmpty()) {
        Util.readCommonWordsFromFile("commonWords.txt");
    }

    int randomNumberOfKnownWords =
        Util.random.nextInt(Util.commonWords.size());

    for(int i = 0; i < randomNumberOfKnownWords; ++i) {

        int generatedIndex =
            Util.random.nextInt(Util.commonWords.size());

        this.alreadyKnownWords
            .add(Util.commonWords.get(generatedIndex));
    }
}
```

- Turtle:
 - Besides the common properties of a pet you need to initialize the **canSpeak**, **bodyType** (randomly), **taxonomy** (in this case: REPTILES) properties.
 - **speak** method: Let's suppose that you cannot hear the voice of the turtle

Object oriented programming

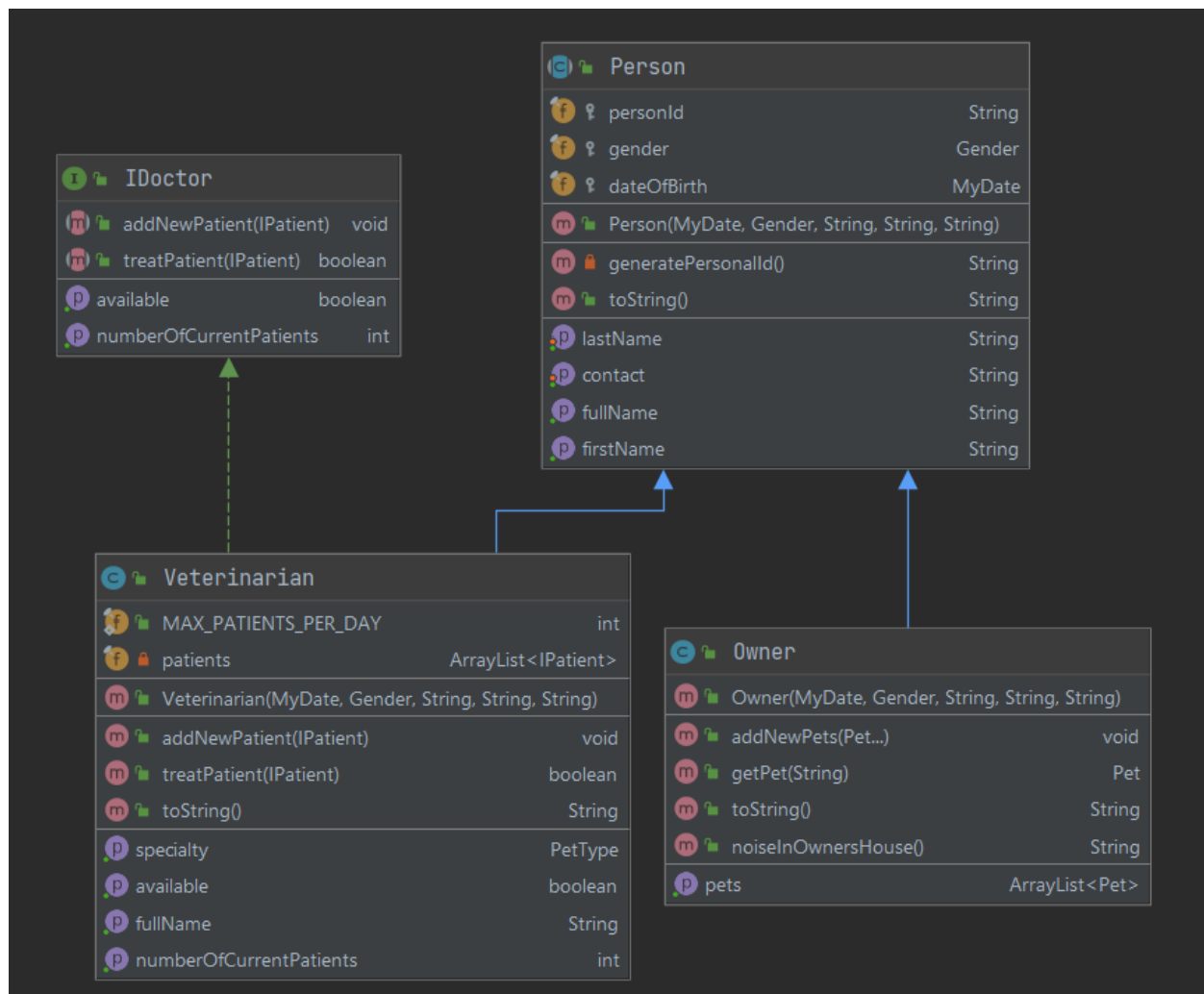
Lab 5 - 6 extra exercises

People

Each person is characterized by a **unique person ID, gender, birthdate, first name, last name and contact (e.g. phone number)**.

You should generate the unique person ID based on Romanian person ID scheme: if the person is a MALE start the ID with number 1, if she is FEMALE with number 2. Continue with 6 numbers based on the birthdate of the person (yymmdd) and then generate a random number with 6 digits.

In our case be sure that we cannot create instances of simple person. Each person is an **Owner** or a **Veterinarian**.



4. figure: Person class and specific types of person

Object oriented programming

Lab 5 - 6 extra exercises

Owner:

An owner can have pets. Therefore his/her house can be sometimes noisy (**noiseInOwnersHouse** method which illustrates the overall voice of pets). Make it possible for an owner to buy new pet(s) (**addNewPets** method) and to call one of his/her pets by name (**getPet** method).

Test your classes (Main):

For the following code:

```
//Create yourself as a happy pet Owner
//For specifying your birthdate use your DateUtil and MyDate classes
(from previous labs) to ensure a correct value of date.
Owner owner1 = new Owner(new MyDate(1997, 7, 8), Enums.Gender.FEMALE,
"Anne", "SMITH", "+40-741-234-567");

//Print the owner to the standard output using the toString() method
System.out.println(owner1);

//Is this owner's house quiet? Check it out!
System.out.println(owner1.noiseInOwnersHouse());

//Let's suppose you really love all kind of pets
//Become the happy owner of one pet from each category (Dog, Cat,
Parrot, Turtle)
//Generate a random birth date (the year should be between 2018 -
2020).
//Use the random field from the Util class to generate random numbers;

// create pets . . .

//Add all pets together to the owner
owner1.addNewPets(pet1, pet2, pet3, pet4);

//Print the owner once again to the standard output
System.out.println(owner1);

//I am curious. Is this owner's house noisy? Check it out!
System.out.println(owner1.noiseInOwnersHouse());
```

The output should be:

```
Anne SMITH is an owner
    Contact information:
        - Personal ID: 2970708236748
        - Date of birth: 1997-07-08
```


Object oriented programming

Lab 5 - 6 extra exercises

```
- She does not have any pets

Anne SMITH's house is quiet because she does not have any pets

Anne SMITH is an owner
  Contact information:
    - Personal ID: 2970708236748
    - Date of birth: 1997-07-08
    - She has 4 pets:
      ~ a Siamese cat, named Joe born in: 2019-04-30
      ~ a French Bulldogs dog, named Lizzie born in: 2018-08-11
      ~ a Small size parrot, named Pityu born in: 2018-11-28
      ~ a Testudinidae turtle, named Tim born in: 2018-10-22

Anne SMITH's house is noisy. Listen...
  - Miaou
  - Wuff wuff
  - This parrot already knows the following words:
    ago put base her order music too simple should receive course
    ground would where try them read happen place enough week summer night has
    war bottom there five pair body example take stood real seem game have heat
    into up upon then later table ten we month dream gold two street again little
    didn't front fire thousand clear two oh until four moon another talk travel
    see live hear cry like ball we passed place to probably two like record
    object travel here bed area half men many million house walk bottom low this
    dark up act after like song table summer front age turn until stay seem live
    other happen explain passed wind town notice boy often late great in between
    something word run far vowel father book words run change think take upon
    complete fish might deep mile never part good stand game once leave did some
    simple has upon size does before science five our school laugh land by house
    sit thousand notice thought or read most common would wrong kind start was
    among
  - You cannot hear the voice of this turtle
```

Veterinarian:

- A veterinarian should be able to have and treat patients (**IDoctor**). Each doctor sometimes is running out of time. Therefore, he/she has a limit of patients per day. This fact makes a veterinarian available or not for new appointments.
- In this exercise each veterinarian is specialized in one pet type. This means that if you have a dog, you can make appointments only to a veterinarian specialized in dogs.

Object oriented programming

Lab 5 - 6 extra exercises

Medical centers:

Each medical center can hire or fire doctors. Furthermore, a med center can assign a patient to a doctor if there is at least one opportunity. Each center also know the all number of doctors and patients and it also should be capable to check the available doctors and number of possible appointments.

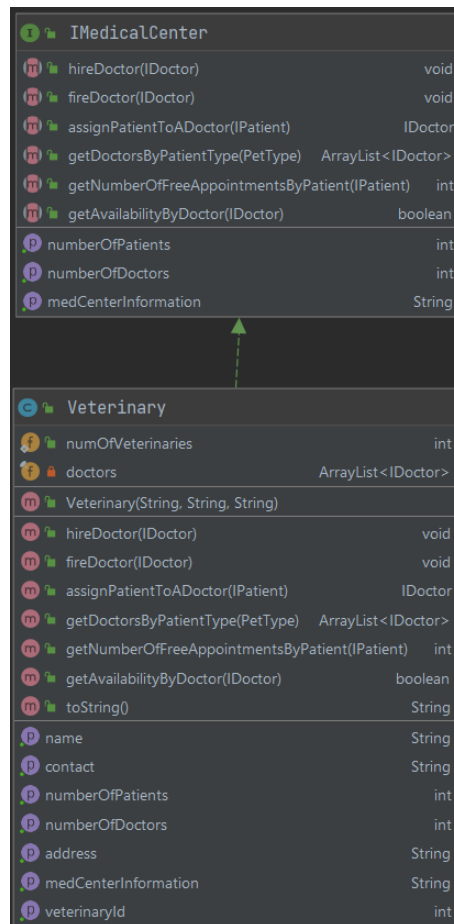
```
String getMedCenterInformation();  
void hireDoctor(IDoctor doctor);  
void fireDoctor(IDoctor doctor);  
IDoctor assignPatientToADoctor (IPatient patient);  
int getNumberOfDoctors();  
int getNumberOfPatients();  
ArrayList<IDoctor> getDoctorsByPatientType(Enums.PetType type);  
int getNumberOfFreeAppointmentsByPatient(IPatient patient);  
boolean getAvailabilityByDoctor(IDoctor doctor);
```

Object oriented programming

Lab 5 - 6 extra exercises

Veterinary:

- A veterinary should “behave” like a medical center.
- It has the following attributes: **ID (based on the number of all veterinaries static field), address, contact and name and a list of doctors**



5. figure: Medical center behaviors and Veterinary class

Test your classes (Main):

Check the following examples:

```
//Create a medical center by the information on the following website:
//https://assisivet.business.site/
Veterinary medicalCenter = new Veterinary(
    "str. Suceava Nr. 4 Targu Mures 540366 Romania",
    "0740 491 345",
    "AssisiVet");

//Print the medical center to the standard output
```

Object oriented programming

Lab 5 - 6 extra exercises

```
System.out.println(medicalCenter);

//Read all diseases and treatments from the diseases_and_symptoms file
//Store these information in the diseases static ArrayList from the
Util class
Util.readDiseasesAndTreatmentsFromFile("diseases.csv");

//Check what kind of disease can occur in case of a
Cat/Dog/Turtle/Parrot
Util.printAllDiseasesByPetType(Enums.PetType.Dog);
```

Output:

```
AssisiVet veterinary
  - Address: str. Suceava Nr. 4 Targu Mures 540366 Romania
  - Contact: 0740 491 345
  - Number of doctors: 0
  - Number of current patients: 0

All diseases in case of Dog
  ~ Treatments for Dental Disease:
    - teeth cleaning
    - extractions
    - root canal

  ~ Treatments for Ear Infections disease:
    - clean and dry the ear
    - topical antibiotic
    - anti-fungal

  ~ Treatments for Itchy skin or skin infections disease:
    - shampoo to treat allergies
    - antibiotic ointment for more severe skin infections

  ~ Treatments for Vomiting and Diarrhea disease:
    - proper diet

  ~ Treatments for Stiffness and Pain disease:
    - keep your dog at a healthy weight
    - use a glucosamine or a chondroitin supplement

  ~ Treatments for Urinary Tract Problems disease:
    - proper antibiotic treatment

  ~ Treatments for Obesity disease:
    - high-quality diet
    - regular exercise
```

Object oriented programming

Lab 5 - 6 extra exercises

Code:

```
//Unfortunately Lizzie gets ill unexpectedly
owner1.getPet("Lizzie").unexpectedIllnessOccurred();

//Check the health status of Lizzie
owner1.getPet("Lizzie").checkHealthStatus();

//You should get an appointment to a veterinary to take care of Lizzie
//Try to get an appointment
medicalCenter.assignPatientToADoctor(owner1.getPet("Lizzie"));
```

Output:

```
Lizzie is suffering from the following disease:
    - Dental Disease

We're sorry but unfortunately there are no opportunities to treat your
pet
```

Hire three doctors to the veterinary and try once again to assign Lizzie to a doctor. If the assignment is successful the doctor can start to heal your pet.

```
//Add veterinarians to the medical center

//Check the availability by doctor

//Print the number of free appointments for Lizzie
System.out.println("Number of possible appointments for Lizzie: " +
medicalCenter.getNumberOfFreeAppointmentsByPatient(owner1.getPet("Lizz
ie")));

//Check once again if you can find an appropriate treatment for Lizzie
IDoctor doc =
medicalCenter.assignPatientToADoctor(owner1.getPet("Lizzie"));
if(doc != null) {
    doc.treatPatient(owner1.getPet("Lizzie"));
}
```

If there was at least one veterinarian for Dogs the output should look like:

```
Peter KALE is a Dog specialist veterinarian
    Contact information:
```

Object oriented programming

Lab 5 - 6 extra exercises

```
- Personal ID: 1930911580645
- Date of birth: 1993-09-11
- He does not have any patients

Number of possible appointments for Lizzie: 1
Lizzie successfully assigned to Peter KALE
Dr. Peter KALE started to heal your pet...
    Generating required treatment...
    Treatments for Dental disease:
        - teeth cleaning
        - extractions
        - root canal

Your pet should recover soon. Take care.
Pet removed from patient list.
```

Check these methods for all of your pets!

Hospitality system:

Write your code so it works as a hospitality system (interactivity with the user).

It should work for the following users:

- Owner
- Veterinarian
- Veterinary

If you choose to be an **Owner** you can do the following actions:

- First you need to specify all required information about yourself (create Owner)
- Buy new pets
- Try to get appointments for your pets
- Choose veterinary
- Choose doctor
- Ask your options (medical centers, veterinarians), opportunities for your pet etc..

If you choose to be a **Veterinarian** you can do the following actions:

- First you need to specify all required information about yourself (create Veterinarian)
- Accept patients

Object oriented programming

Lab 5 - 6 extra exercises

- Treat patients
- Prioritize patients (you need to include seriousness to a disease as well)
- List medical centers
- Ask a medical center to apply to a job etc..

If you choose to be a **Veterinary** you can do the following actions:

- First you need to specify all required information about yourself (create Veterinary)
- Accept patients if there is at least one doctor who is available
- Hire doctors
- Fire doctors
- List all of your patients and doctors etc

Fields for enums:

CatBreed:

```
Polydactyl,  
Snowshoe,  
Calico,  
BritishShorthair("British Shorthair"),  
Siamese,  
JapaneseBobtail("Japanese Bobtail"),  
Persian,  
ScottishFold("Japanese Bobtail"),  
GrayTabby("Gray abby");
```

DogBreed:

```
LabradorRetrievers("Labrador Retrievers"),  
GermanShepherds("German Shepherds"),  
GoldenRetrievers("Golden Retrievers"),  
FrenchBulldogs("French Bulldogs"),  
Bulldogs,  
Beagles,  
Poodles;
```

Object oriented programming

Lab 5 - 6 extra exercises

BodyType:

```
Small,  
Medium,  
Large
```

TurtleClassification:

```
Testudinoidea,  
Emydidae,  
Testudinidae,  
Geoemydidae,  
Trionychidae
```

Seriosity:

```
GREEN,  
YELLOW,  
RED
```

Gender:

```
FEMALE,  
MALE
```

PetType:

```
Dog,  
Cat,  
Turtle,  
Parrot
```

Taxonomy:

```
MAMMALS,  
BIRDS,  
REPTILES,  
FISH,  
AMPHIBIANS,  
INVERTEBRATES
```


Object oriented programming

Lab 5 - 6 extra exercises

Useful links

- https://www.w3schools.com/java/java_abstract.asp
- <https://docs.oracle.com/javase/tutorial/java/IandI/abstract.html>
- https://www.w3schools.com/java/java_interface.asp
- https://www.w3schools.com/java/java_methods_param.asp
- https://www.w3schools.com/java/java_methods_overloading.asp
- https://www.w3schools.com/java/java_polymorphism.asp
- Find the whole class diagram on the link below:
<https://moodle.ms.sapientia.ro/mod/resource/view.php?id=8689>