Task #1 (1.5 points): Provide a description of the system you will be modelling in this course.

In this project, our team is modelling a **Pet Adoption System**. The system is designed to support several types of users and to handle both the management of animals and the adoption process.

The main user roles in the system are:

- **Administrator** - manages the overall database, creates/deletes accounts, verifies information and supervises the whole process.
- **Shelter worker** - responsible for adding new animals, updating their records (health status, vaccinations, sterilization) and maintaining availability status.
- **Adopter (potential owner)** - registers in the system, browses available animals and submits adoption requests.
- **Veterinarian** – updates medical information and creates health records (check-ups, treatments, surgeries).

Classes and Attributes:

1. **User** (base class)
   - userID : int
   - name : String
   - email : String
   - phone : String
   - registrationDate : Date
   - role : UserRole
2. **Administrator** (inherits User)
   - adminID : String
   - department : String
   - accessLevel : String
   - canManageUsers : Boolean
   - canManageSystem : Boolean
   - canViewReports : Boolean
   - lastSystemAudit : Date
3. **Animal**
   - animalID : String
   - name : String
   - species : String
   - gender : String
   - breed : String
   - dateOfBirth : Date
   - arrivalDate : Date
   - temperament : String
   - status : AdoptionStatus
   - age : int
4. **MedicalRecord**
   - recordID : int
   - date : Date
   - description : String

    - vetID : int
    - vetNotes : String
5. **AdoptionRequest**
    - requestID : int
    - adopterID : int
    - animalID : int
    - requestDate : Date
    - adopterMotivation : String
    - status : String
6. **AdoptionContract**
    - contractID : int
    - adopterID : int
    - animalID : int
    - adoptionDate : Date
    - notes : String
7. **Shelter**
    - shelterID : String
    - name : String
    - address : String
    - phone : String
    - email : String
    - capacity : Integer
    - currentOccupancy : Integer
    - operatingHours : String
    - availableSpots : int
8. **ShelterWorker (inherits User)**
    - employeeID : String
    - position : String
    - hireDate : Date
9. **Adopter (inherits User)**
    - name : String
    - address : String
    - hasPreviousPets : Boolean
10. **Vet (inherits User)**
    - licenseNumber : String
    - specialization : String
    - clinicName : String

Subclasses:

- **Dog** - isTrained : Boolean
- **Cat** - isIndoorCat : Boolean

AdoptionStatus (Enum)

AVAILABLE | RESERVED | ADOPTED | MEDICAL_HOLD

Relationships between classes:

- **Shelter – Animal** → 1 to * : Each shelter can host many animals, but each animal belongs to one shelter.

- **Animal – MedicalRecord** → 1 to * : an animal may have several medical records.
- **Animal – AdoptionRequest** → 1 to * : multiple adopters can request the same animal.
- **Adopter – AdoptionRequest** → 1 to * : one adopter may send multiple requests.
- **Adopter – AdoptionContract – Animal** → many-to-many through AdoptionContract.
- **Vet – MedicalRecord** → 1 to * : each record is created by one vet.

Example scenario:

A shelter worker enters a new animal into the system with its basic details. An adopter browses through available pets and submits an adoption request. The administrator reviews the request and either approves or rejects it. If the request is approved, an adoption contract is created. Meanwhile, the veterinarian can add medical records about the animal both before and after adoption.

With this design, the system supports all the main processes: managing animals, coordinating different user roles, recording medical history, and processing adoptions