Spring Pet Clinic: Class UML Diagram

Description

The UML Class Diagram illustrates the entities and their relationships within the Spring Pet Clinic applications, also helps readers visualize the layout of the classes.

Purpose

This UML Class Diagram is included to provide a visual representation of the entities and their relationships within the Spring Petclinic application, enhancing understanding of its data structure

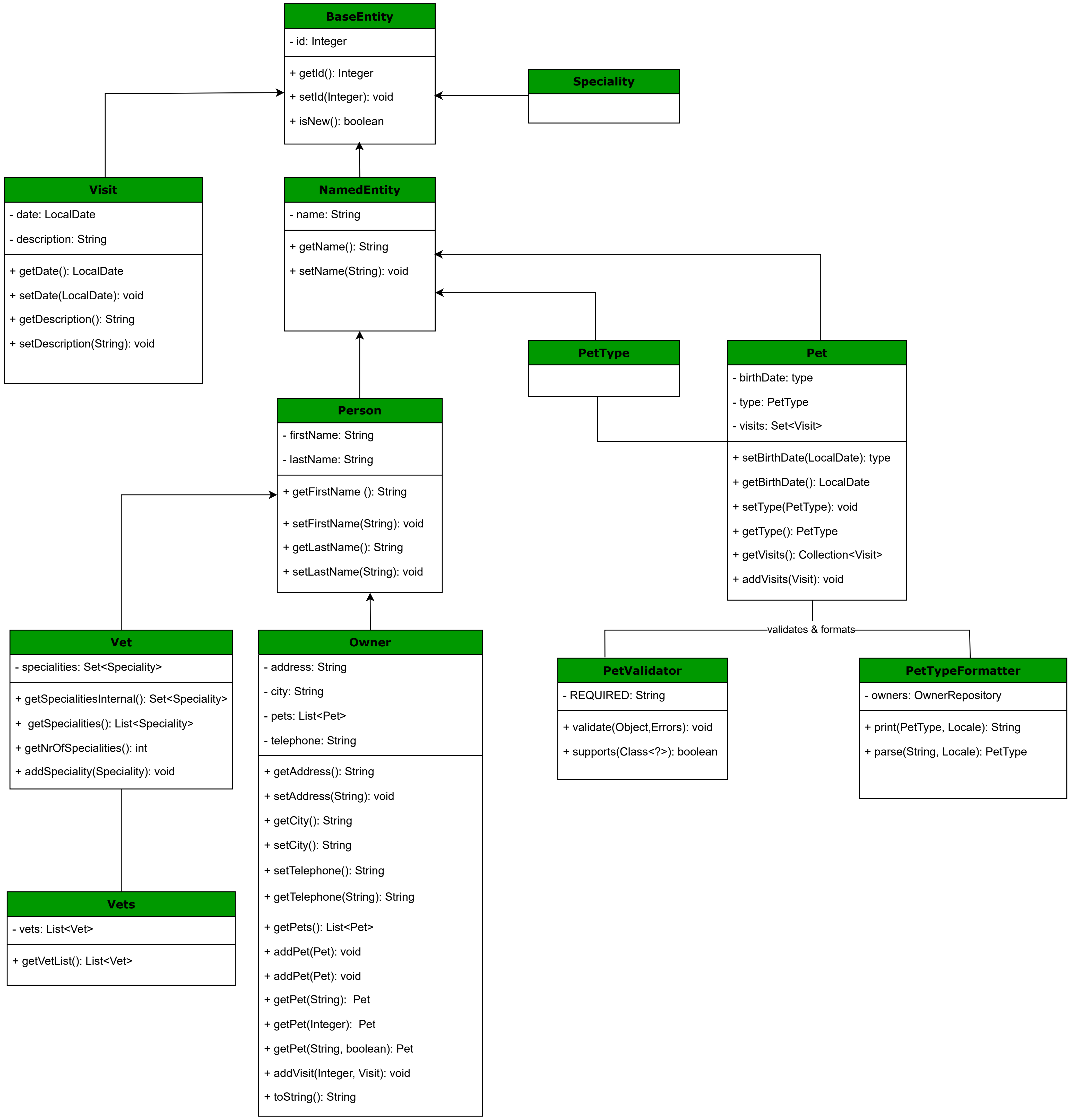
Outcomes

* The UML Class diagram provides a clear, visual representation of the application's underlying data structure.
* The diagram serves as reference point of data models for both technical and non-technical stakeholders.
* Developers who open the diagram for the first time will get to understand what to expect once they open or maintain the spring pet clinic.
* The diagram can serve as reference for database design & support

Classes

1. BaseEntity
2. NamedEntity
3. Person
4. Visit
5. Speciality
6. Owner
7. Pet
8. PetType
9. PetTypeFormatter
10. PetValidator
11. Vets
12. Vet

The UML class diagram is shown below:



BaseEntity
- id: Integer
+ getId(): Integer
+ setId(Integer): void
+ isNew(): boolean

Speciality

Visit
- date: LocalDate
- description: String
+ getDate(): LocalDate
+ setDate(LocalDate): void
+ getDescription(): String
+ setDescription(String): void

NamedEntity
- name: String
+ getName(): String
+ setName(String): void

PetType

Pet
- birthDate: type
- type: PetType
- visits: Set<Visit>
+ setBirthDate(LocalDate): type
+ getBirthDate(): LocalDate
+ setType(PetType): void
+ getType(): PetType
+ getVisits(): Collection<Visit>
+ addVisits(Visit): void

Person
- firstName: String
- lastName: String
+ getFirstName (): String
+ setFirstName(String): void
+ getLastName(): String
+ setLastName(String): void

Vet
- specialities: Set<Speciality>
+ getSpecialitiesInternal(): Set<Speciality>
+ getSpecialities(): List<Speciality>
+ getNrOfSpecialities(): int
+ addSpeciality(Speciality): void

Owner
- address: String
- city: String
- pets: List<Pet>
- telephone: String
+ getAddress(): String
+ setAddress(String): void
+ getCity(): String
+ setCity(): String
+ setTelephone(): String
+ getTelephone(String): String
+ getPets(): List<Pet>
+ addPet(Pet): void
+ addPet(Pet): void
+ getPet(String): Pet
+ getPet(Integer): Pet
+ getPet(String, boolean): Pet
+ addVisit(Integer, Visit): void
+ toString(): String

PetValidator
- REQUIRED: String
+ validate(Object,Errors): void
+ supports(Class<?>): boolean

PetTypeFormatter
- owners: OwnerRepository
+ print(PetType, Locale): String
+ parse(String, Locale): PetType

Vets
- vets: List<Vet>
+ getVetList(): List<Vet>

Spring Pet Clinic: Layered Architecture Diagram

Description

This diagram breaks down Spring Petclinic's layered architecture, providing a visual overview of the application's layers and their respective components.

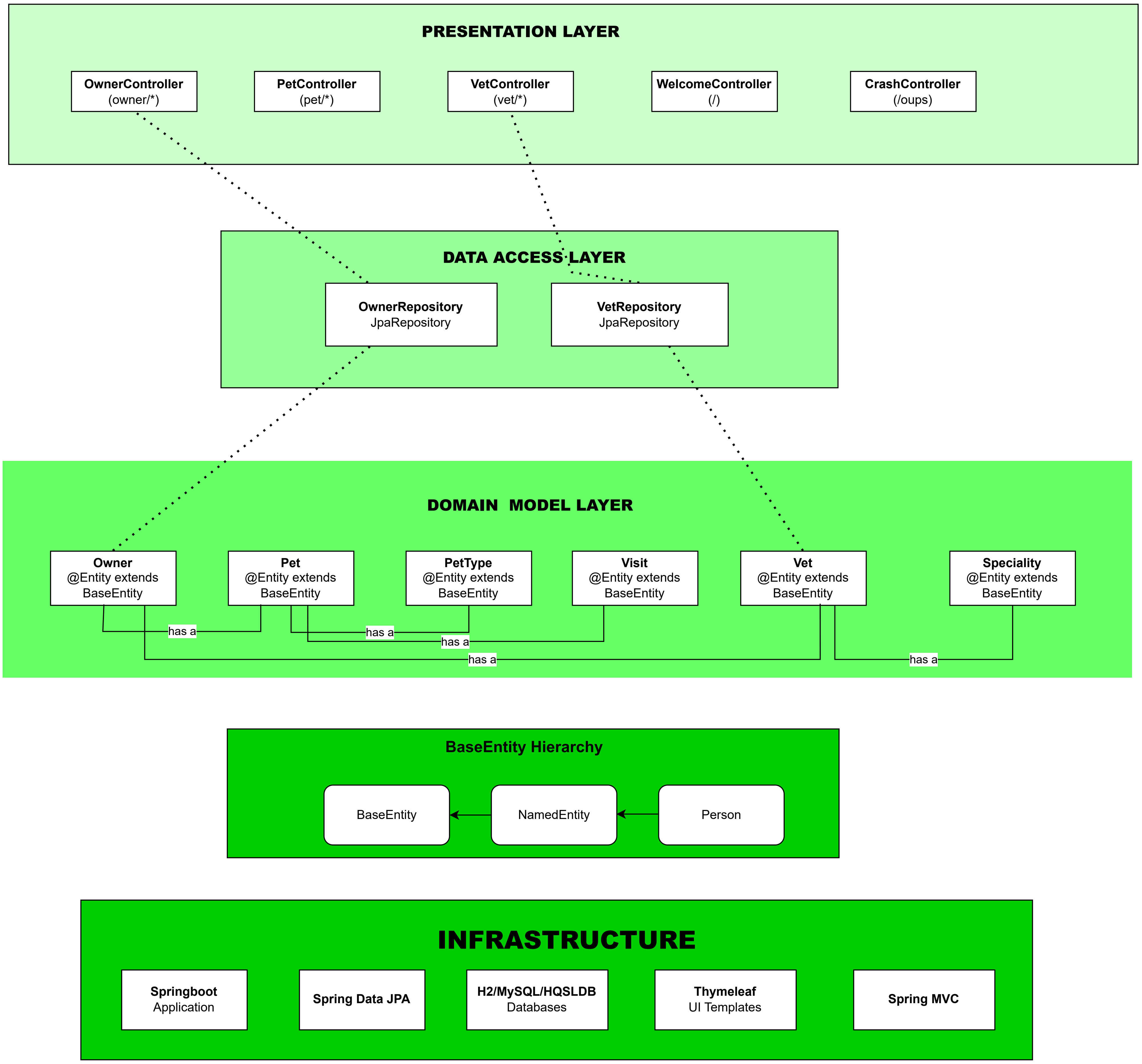
Purpose

This layered architecture diagram is included to help the reader learn how the Spring Petclinic application is organized in layers, and to better understand the relationships between its

Layers documented:

1. Presentation Layer : Handles user HTTP requests through controllers such as OwnerController,PetController,VetController, etc.
2. Data Access Layer : Handles communication with the database, it uses Spring Data JPA repositories.
3. Domain Model Layer : Contains the core business logic and data. it includes entities like Owner, Pet, Vet . it serves as a model for real world business objects.
4. BaseEntity Hierarchy: Provides a common base(from BaseEntity) for entities.
5. Infrastructure : Shows supporting technical components, includes but not limited to Spring Boot, H2/MySQL/HSqldb databases, Thymeleaf, and Spring MVC.

The Layered Architecture Diagram is shown below:


PRESENTATION LAYER
OwnerController (owner/*)
PetController (pet/*)
VetController (vet/*)
WelcomeController (/)
CrashController (/oups)
DATA ACCESS LAYER
OwnerRepository JpaRepository
VetRepository JpaRepository
DOMAIN MODEL LAYER
Owner @Entity extends BaseEntity
Pet @Entity extends BaseEntity
PetType @Entity extends BaseEntity
Visit @Entity extends BaseEntity
Vet @Entity extends BaseEntity
Speciality @Entity extends BaseEntity
has a
has a
has a
has a
has a
BaseEntity Hierarchy
BaseEntity
NamedEntity
Person
INFRASTRUCTURE
Springboot Application
Spring Data JPA
H2/MySQL/HQSLDB Databases
Thymeleaf UI Templates
Spring MVC

Spring Pet Clinic github Repository: spring-projects/spring-petclinic: A sample Spring-based application