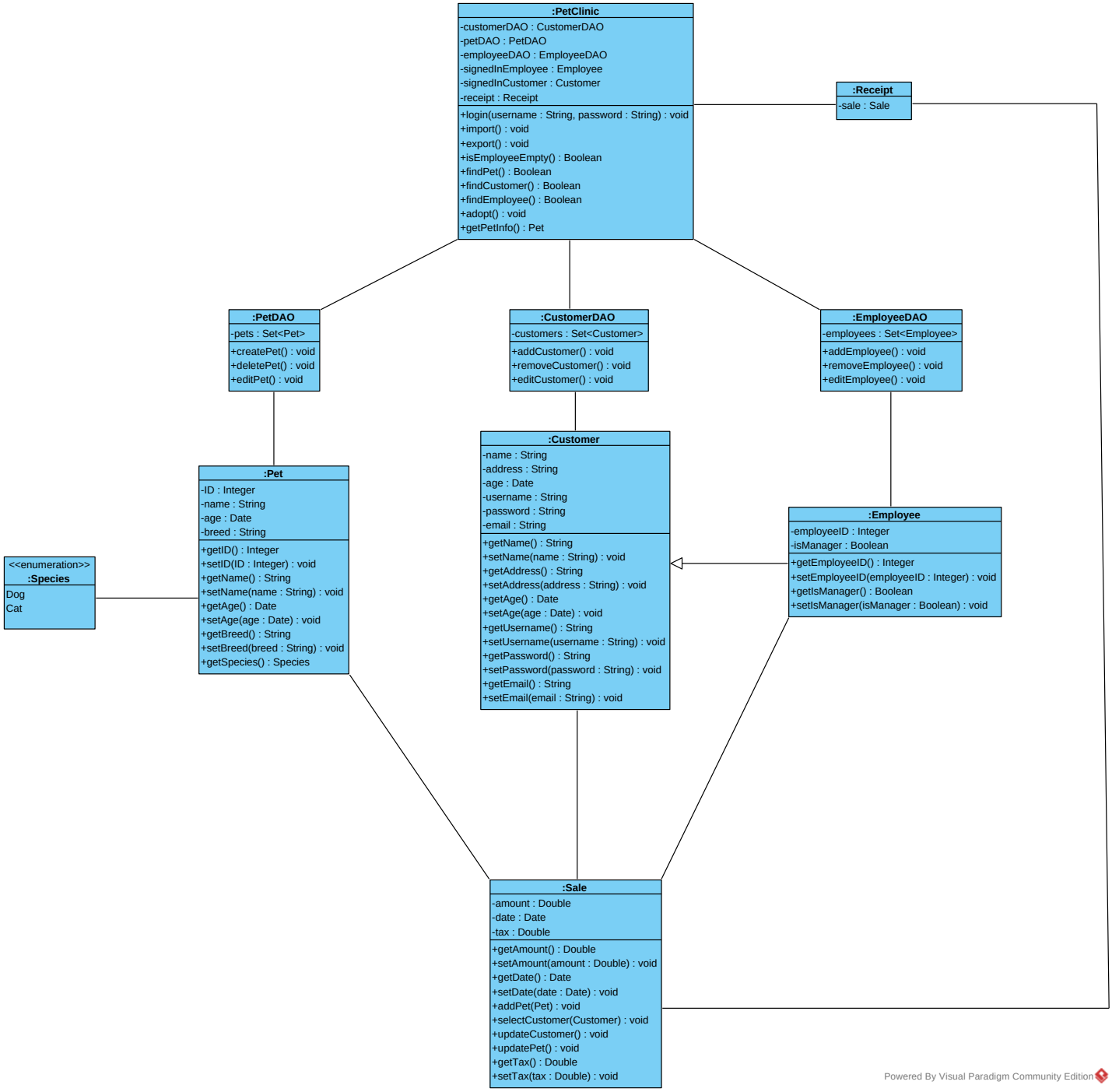


**GRASP JUSTIFICATION FOR A PARTICULAR CLASS**

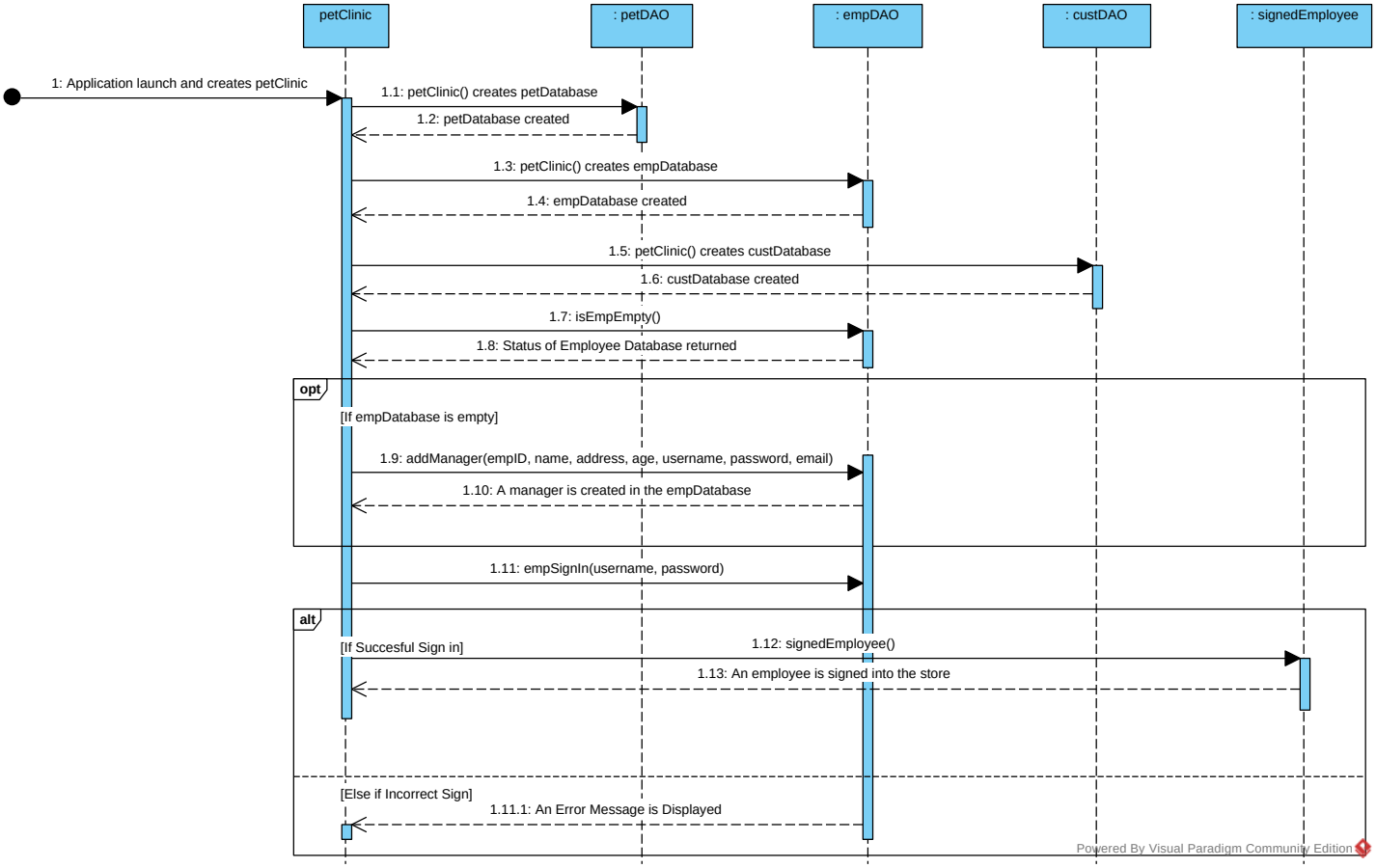
**CLASS:** PetClinic

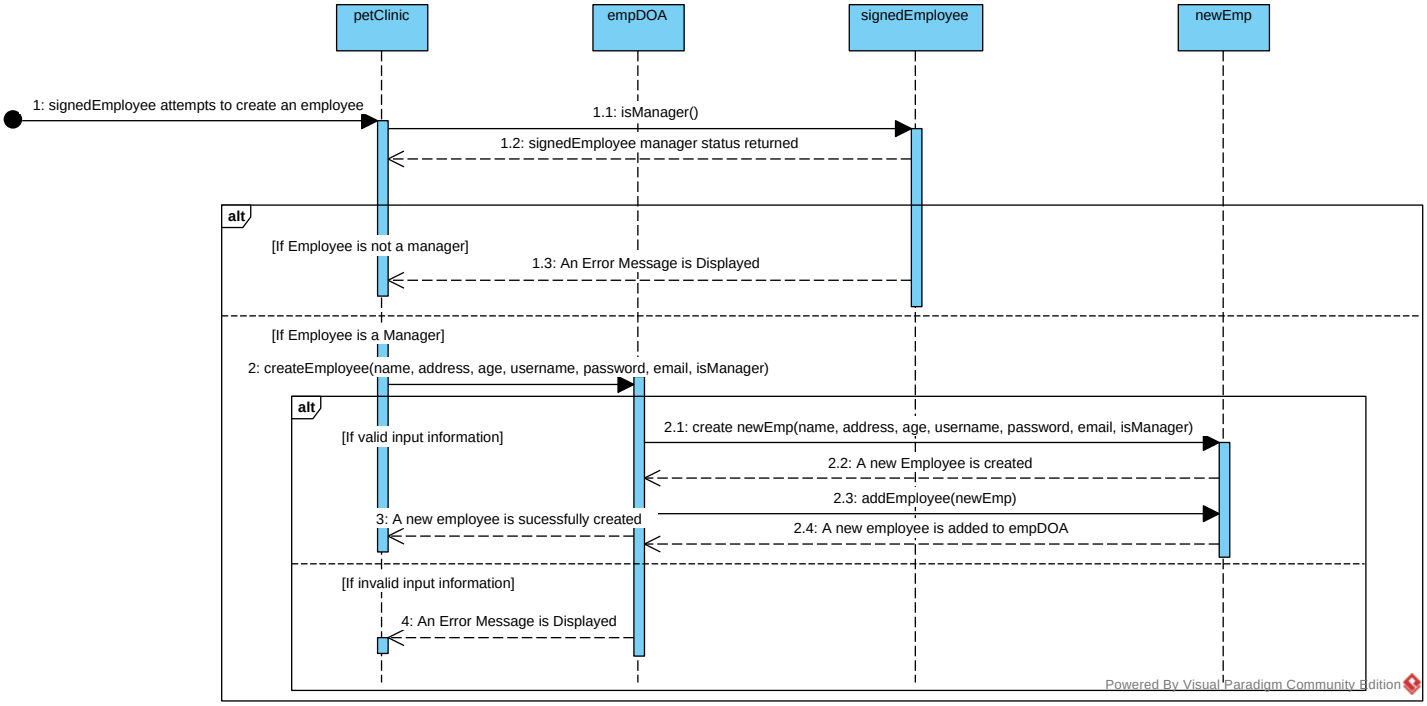
**OPERATIONS**

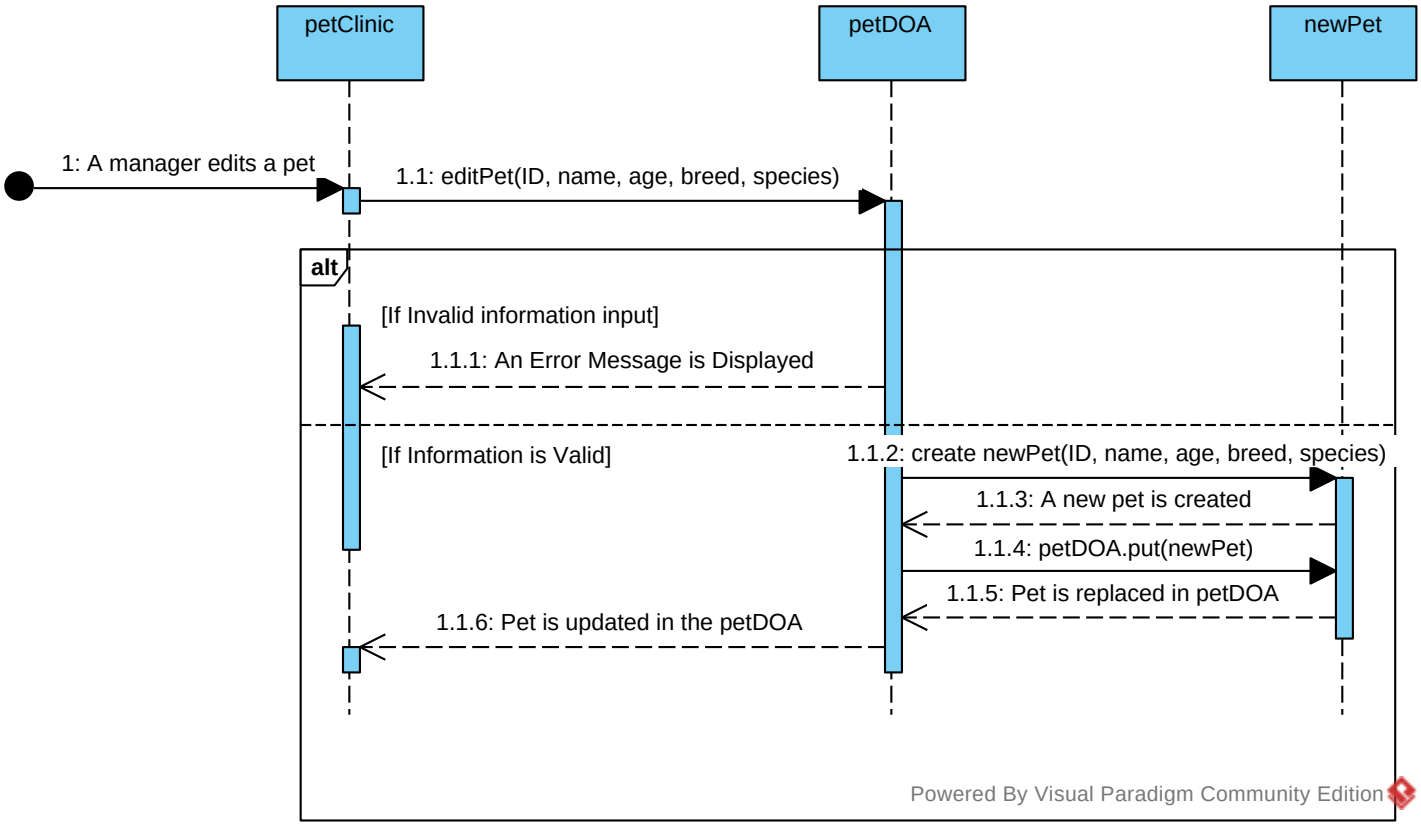
- **Operations**
    1. Import()
        - The responsibility of the PetClinic class is to act as an Information Expert and Creator to the DAO objects as well as provide Indirection to them. Having the Import function allows to maintain high cohesion as the PetClinic can read in and update its DOAs as needed
    2. Export()
        - The responsibility of the PetClinic class is to act as an Information Expert and Creator to the DAO objects as well as provide Indirection to them. Having the Export function allows to maintain high cohesion as the PetClinic can write to a file to later update its DAOs as needed
    3. isEmployeeEmpty()
        - The responsibility of the PetClinic class is to act as an Information Expert and Creator to the DOA objects as well as provide Indirection to them. The PetClinic Class must know if it's EmployeeDAO is empty ensure the DAO never tries to refer to a null object.
    4. findPet()
        - The findPet method allows for indirection from outside controllers by searching through the PetDAO class by going through the PetClinic class first before doing a DML function inside the PetDAO class. This also supports High cohesion between the PetClinic and its petInventory member variable while also supporting low cohesion as the PetDAO will do the desired DML and not the PetClinic Class
    5. findCustomer()
        - The findCustomer method allows for indirection from outside controllers by searching through the CustomerDAO class by going through the PetClinic class first before doing a DML function inside the CustomerDAO class. This also supports High cohesion between the PetClinic and its custDatabase member variable while also supporting low cohesion as the CustomerDAO will do the desired DML and not the PetClinic Class
    6. findEmployee()
        - The findEmployee method allows for indirection from outside controllers by searching through the EmployeeDAO class by going through the PetClinic class first before doing a DML function inside the EmployeeDAO class. This also supports High cohesion between the

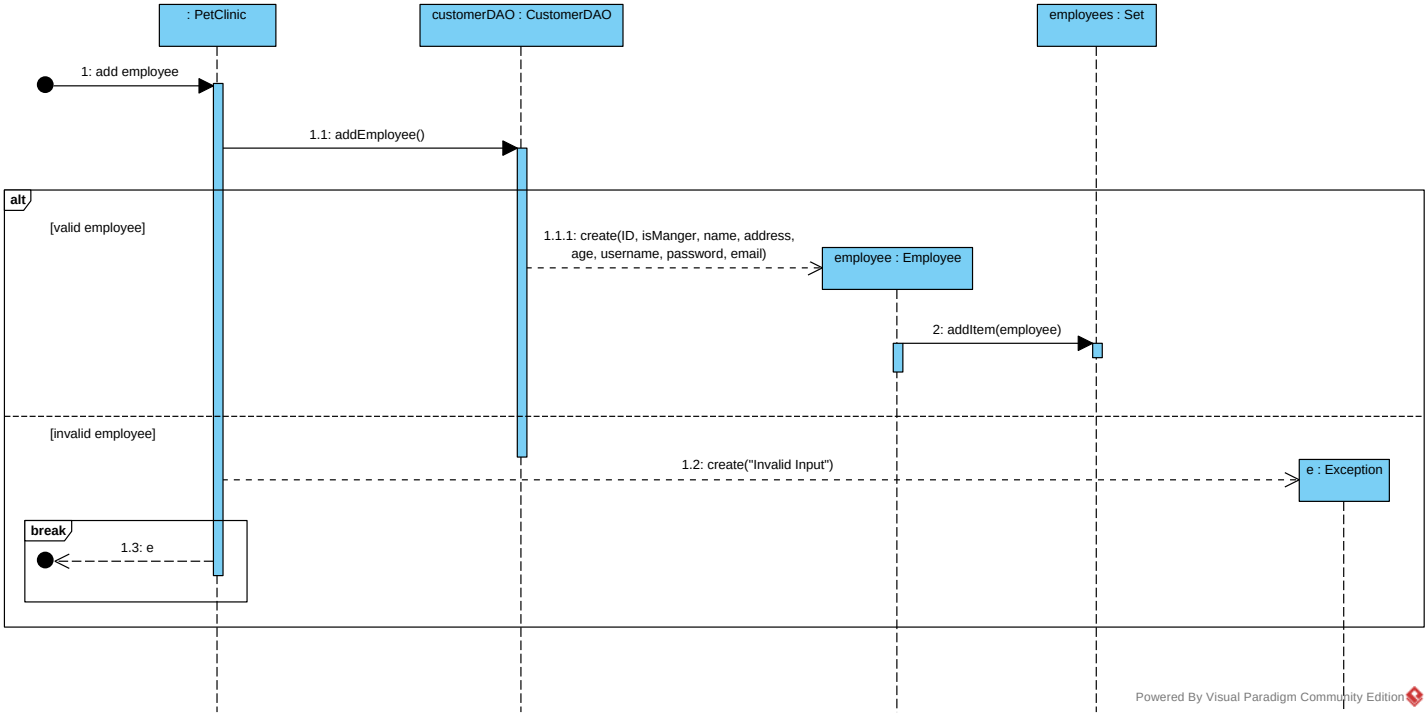
PetClinic and its employDatabase member variable while also supporting low cohesion as the EmployeeDAO will do the desired DML and not the PetClinic Class

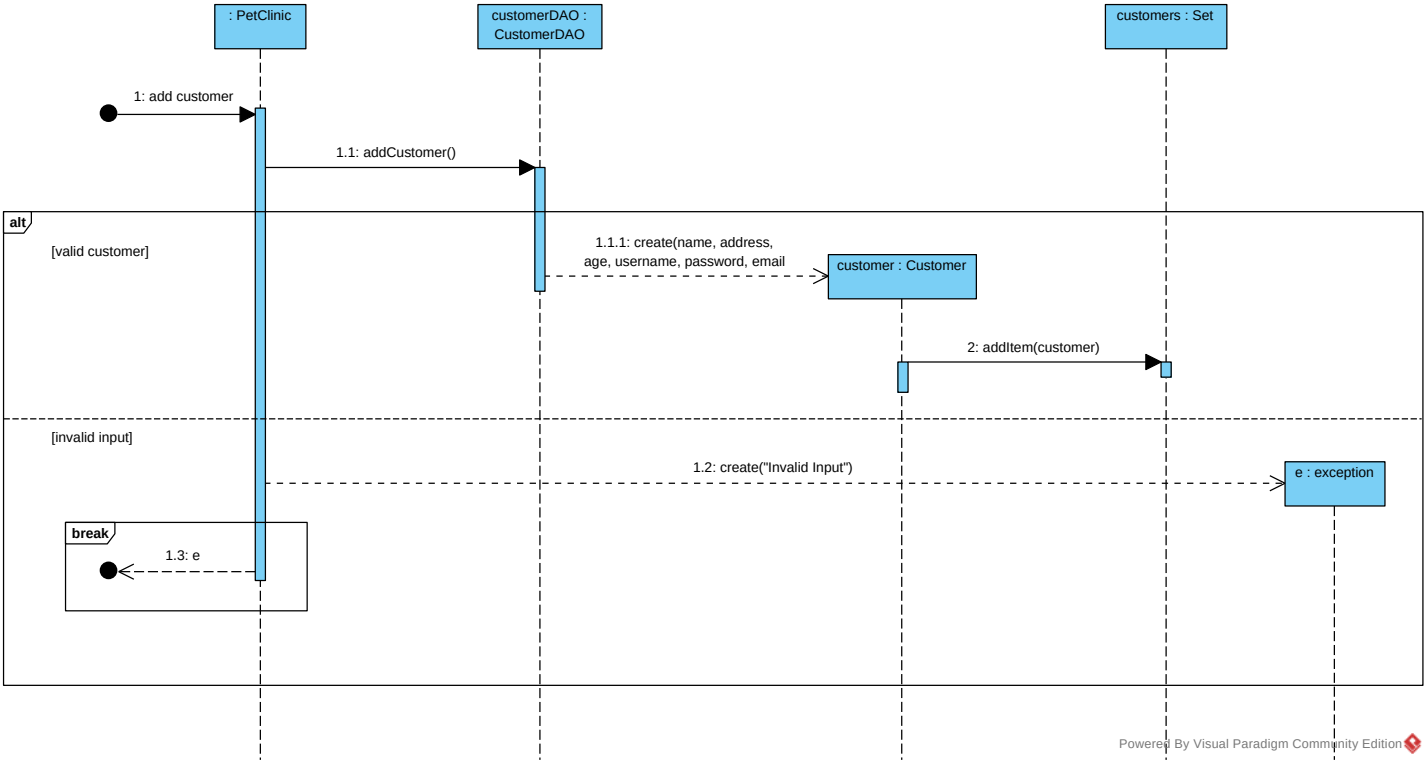
7. Adopt()
    - The Adopt method allows for a pure fabrication approach when manipulating the PetClinic Class's member variables. It allows for Indirection for the Sale class by only being able to create a receipt. By using Indirection and Pure Fabrication in this way, we can maintain a lower coupling for the PetClinic class and also a high cohesion between the PetClinic Class and its DAO member variables

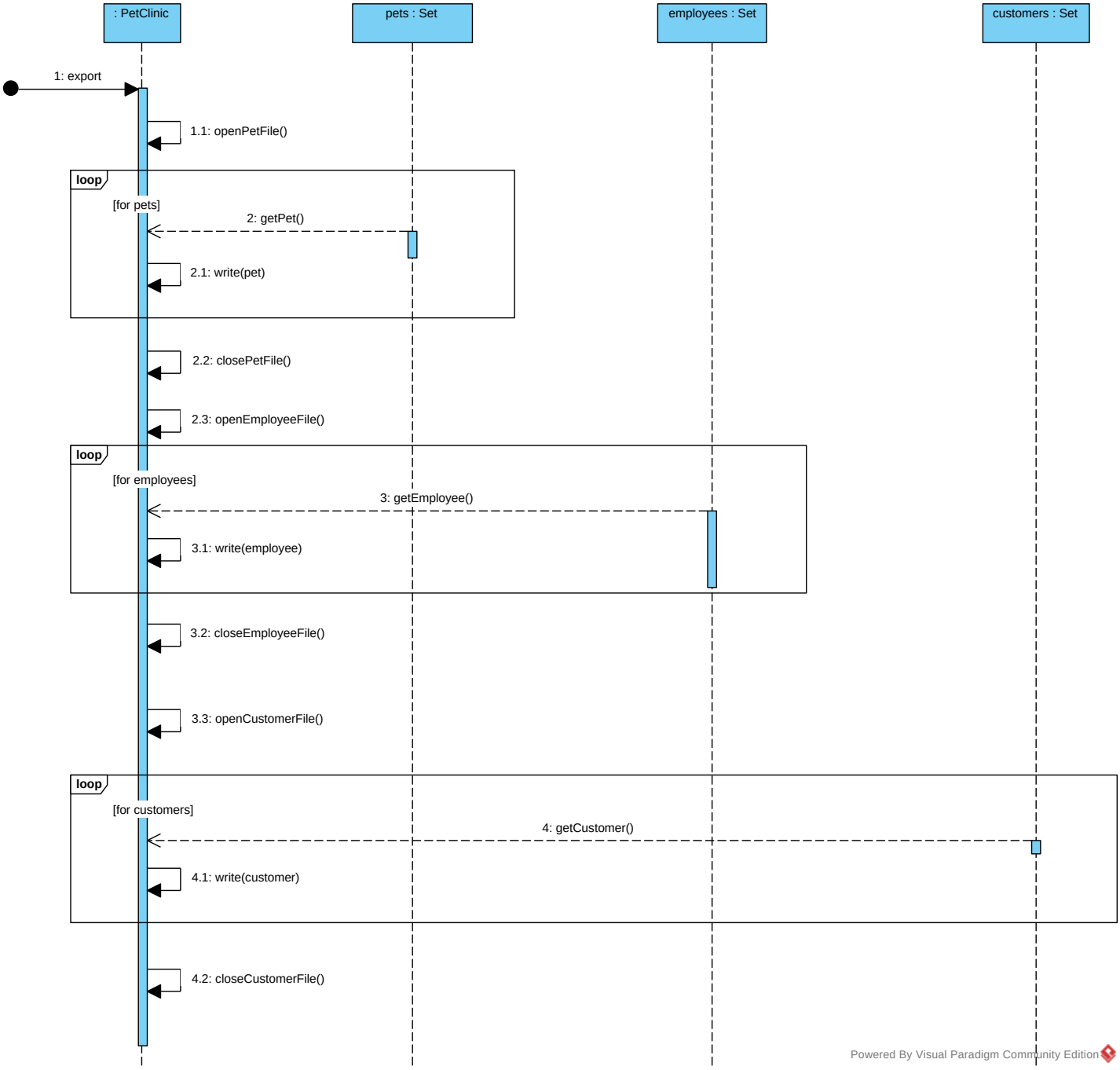

petClinic
: petDAO
: empDAO
: custDAO
: signedEmployee
1: Application launch and creates petClinic
1.1: petClinic() creates petDatabase
1.2: petDatabase created
1.3: petClinic() creates empDatabase
1.4: empDatabase created
1.5: petClinic() creates custDatabase
1.6: custDatabase created
1.7: isEmpEmpty()
1.8: Status of Employee Database returned
opt
[If empDatabase is empty]
1.9: addManager(empID, name, address, age, username, password, email)
1.10: A manager is created in the empDatabase
1.11: empSignIn(username, password)
alt
[If Succesful Sign in]
1.12: signedEmployee()
1.13: An employee is signed into the store
[Else if Incorrect Sign]
1.11.1: An Error Message is Displayed
Powered By Visual Paradigm Community Edition

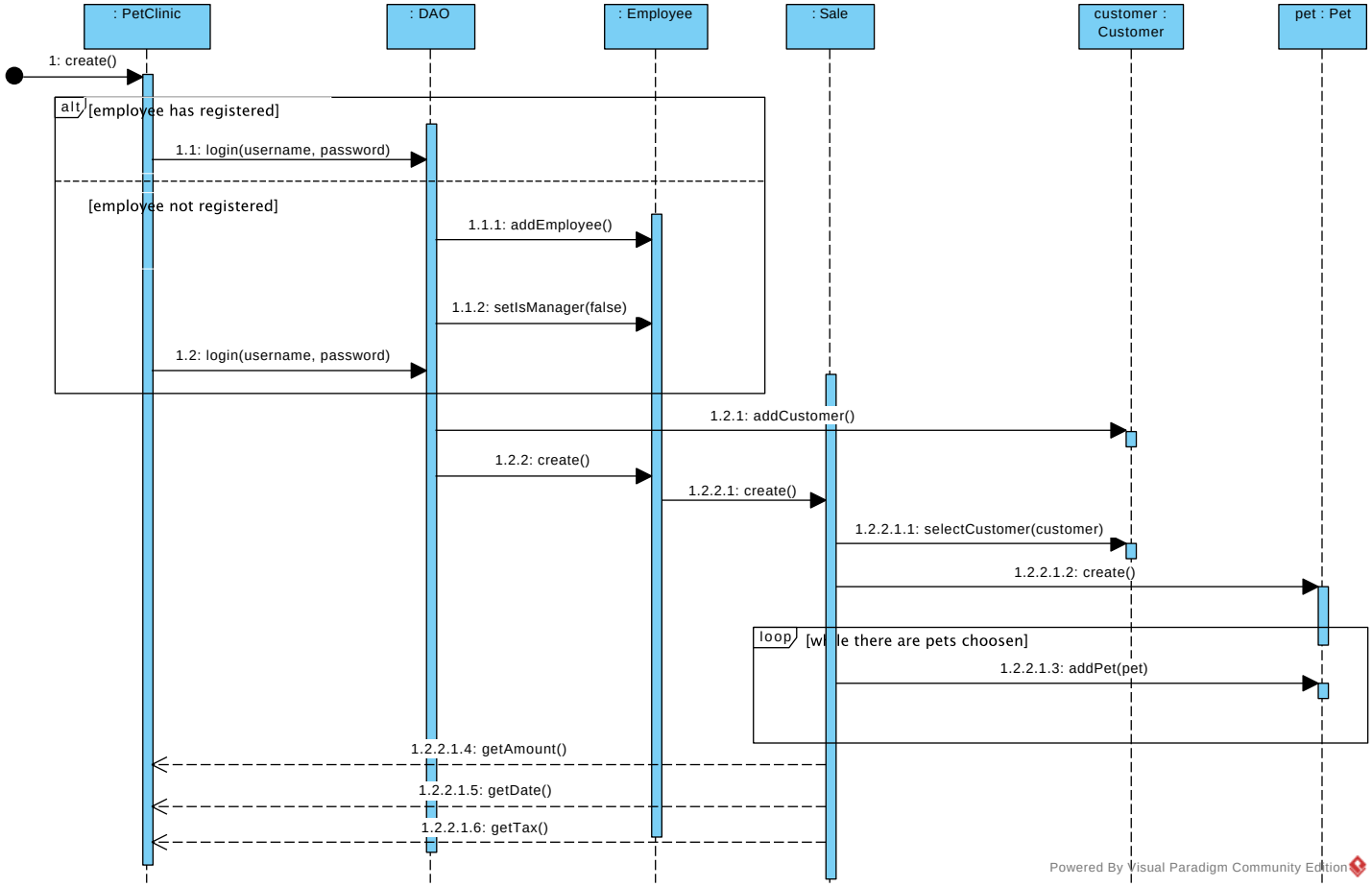

petClinic
empDOA
signedEmployee
newEmp
1: signedEmployee attempts to create an employee
1.1: isManager()
1.2: signedEmployee manager status returned
alt
[If Employee is not a manager]
1.3: An Error Message is Displayed
[If Employee is a Manager]
2: createEmployee(name, address, age, username, password, email, isManager)
alt
[If valid input information]
2.1: create newEmp(name, address, age, username, password, email, isManager)
2.2: A new Employee is created
2.3: addEmployee(newEmp)
3: A new employee is sucessfully created
2.4: A new employee is added to empDOA
[If invalid input information]
4: An Error Message is Displayed
Powered By Visual Paradigm Community Edition

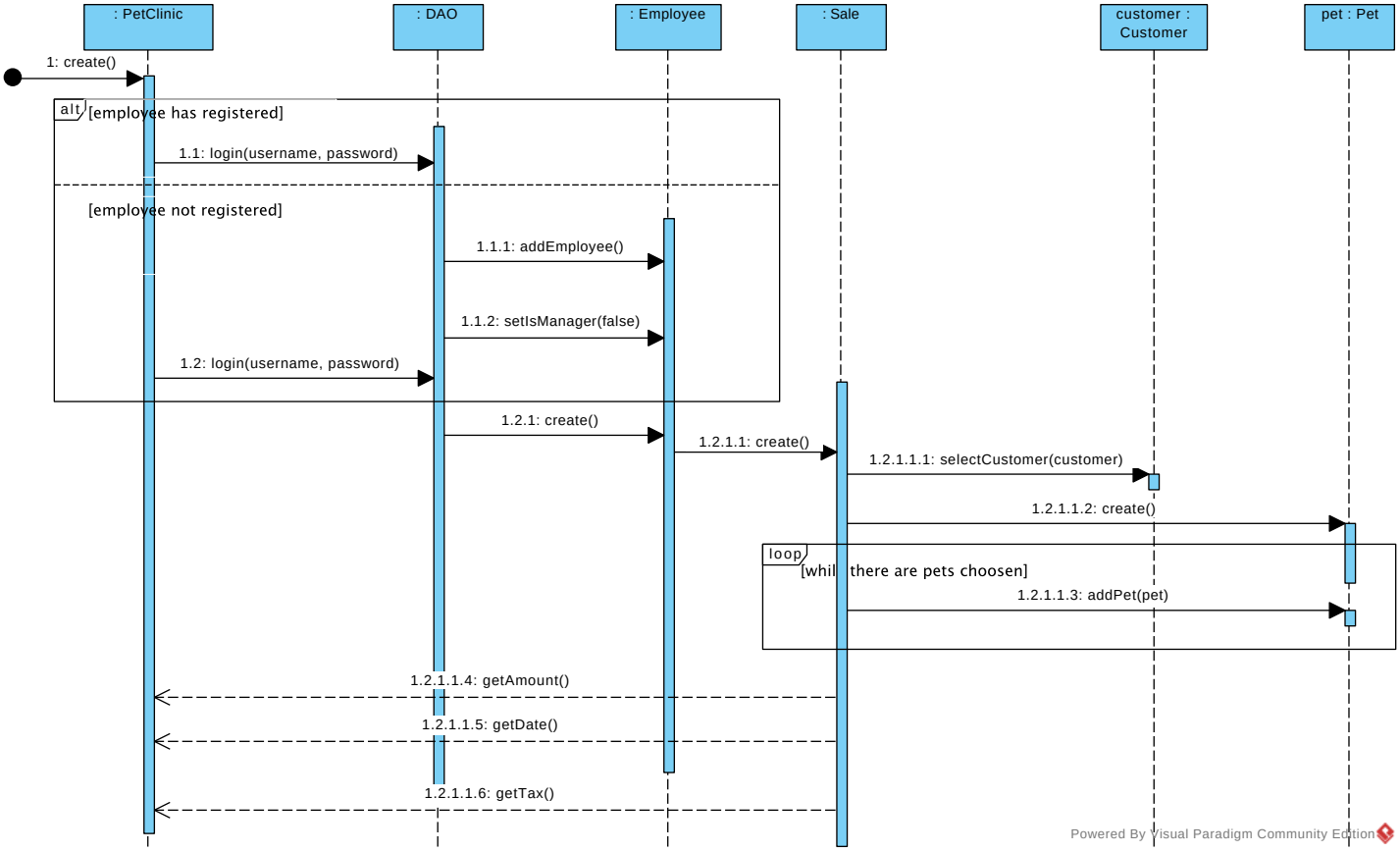

petClinic
petDOA
newPet
1: A manager edits a pet
1.1: editPet(ID, name, age, breed, species)
alt
[If Invalid information input]
1.1.1: An Error Message is Displayed
[If Information is Valid]
1.1.2: create newPet(ID, name, age, breed, species)
1.1.3: A new pet is created
1.1.4: petDOA.put(newPet)
1.1.5: Pet is replaced in petDOA
1.1.6: Pet is updated in the petDOA
Powered By Visual Paradigm Community Edition

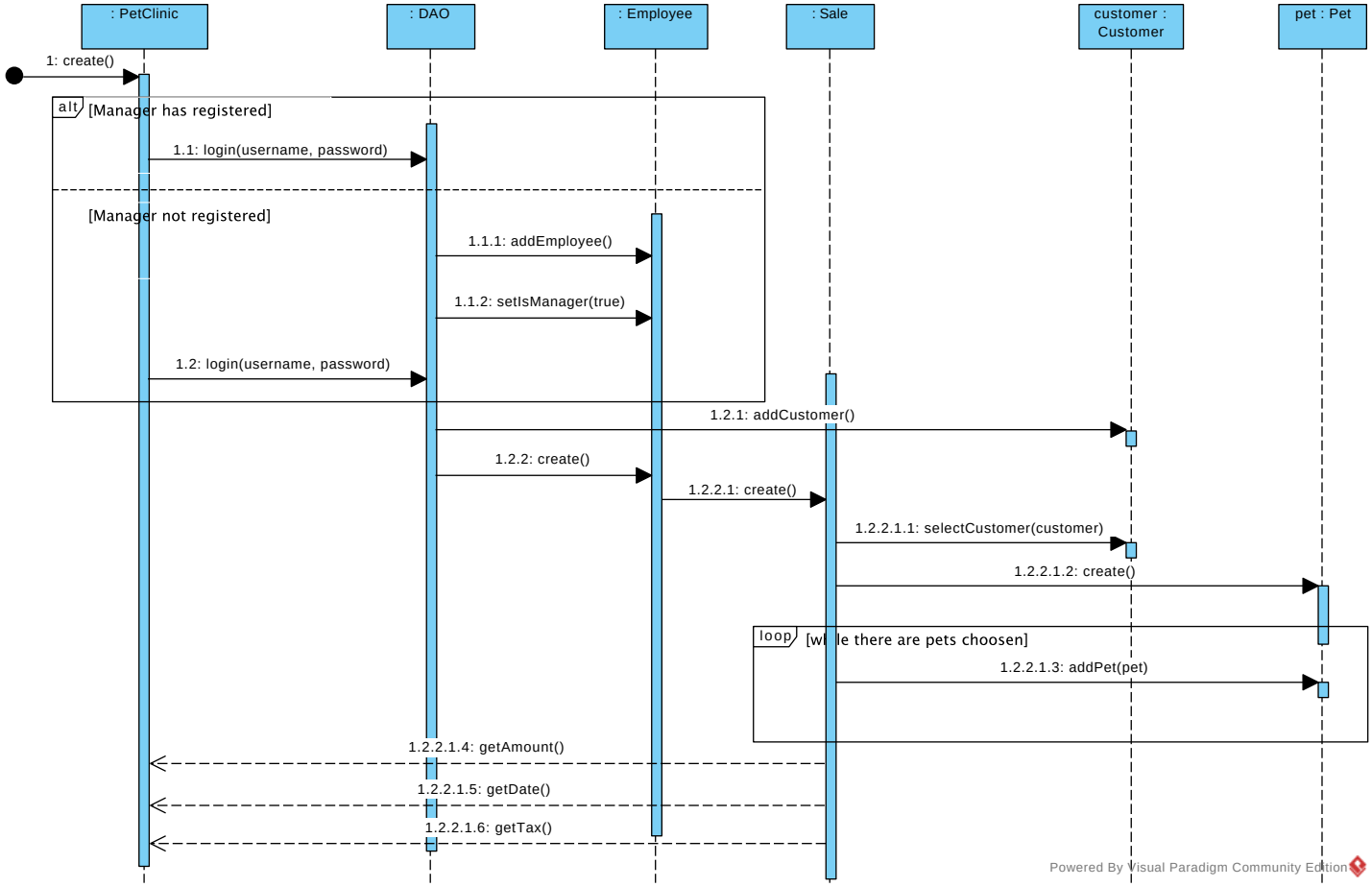

: PetClinic
customerDAO : CustomerDAO
employees : Set
1: add employee
1.1: addEmployee()
alt
[valid employee]
1.1.1: create(ID, isManger, name, address, age, username, password, email)
employee : Employee
2: addItem(employee)
[invalid employee]
1.2: create("Invalid Input")
e : Exception
break
1.3: e
Powered By Visual Paradigm Community Edition

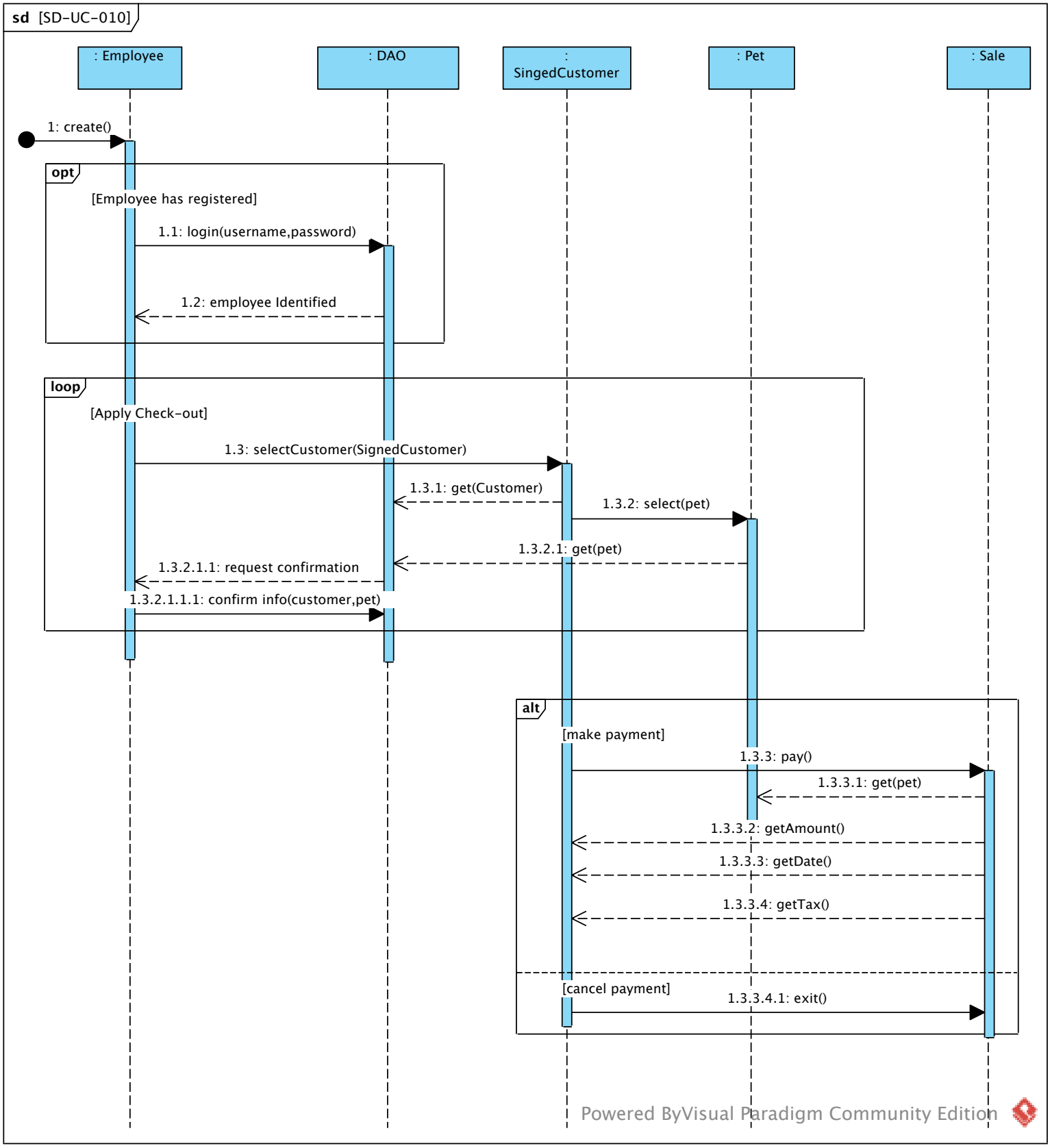

: PetClinic
customerDAO : CustomerDAO
customers : Set
1: add customer
1.1: addCustomer()
alt
[valid customer]
1.1.1: create(name, address, age, username, password, email
customer : Customer
2: addItem(customer)
[invalid input]
1.2: create("Invalid Input")
e : exception
break
1.3: e
Powered By Visual Paradigm Community Edition

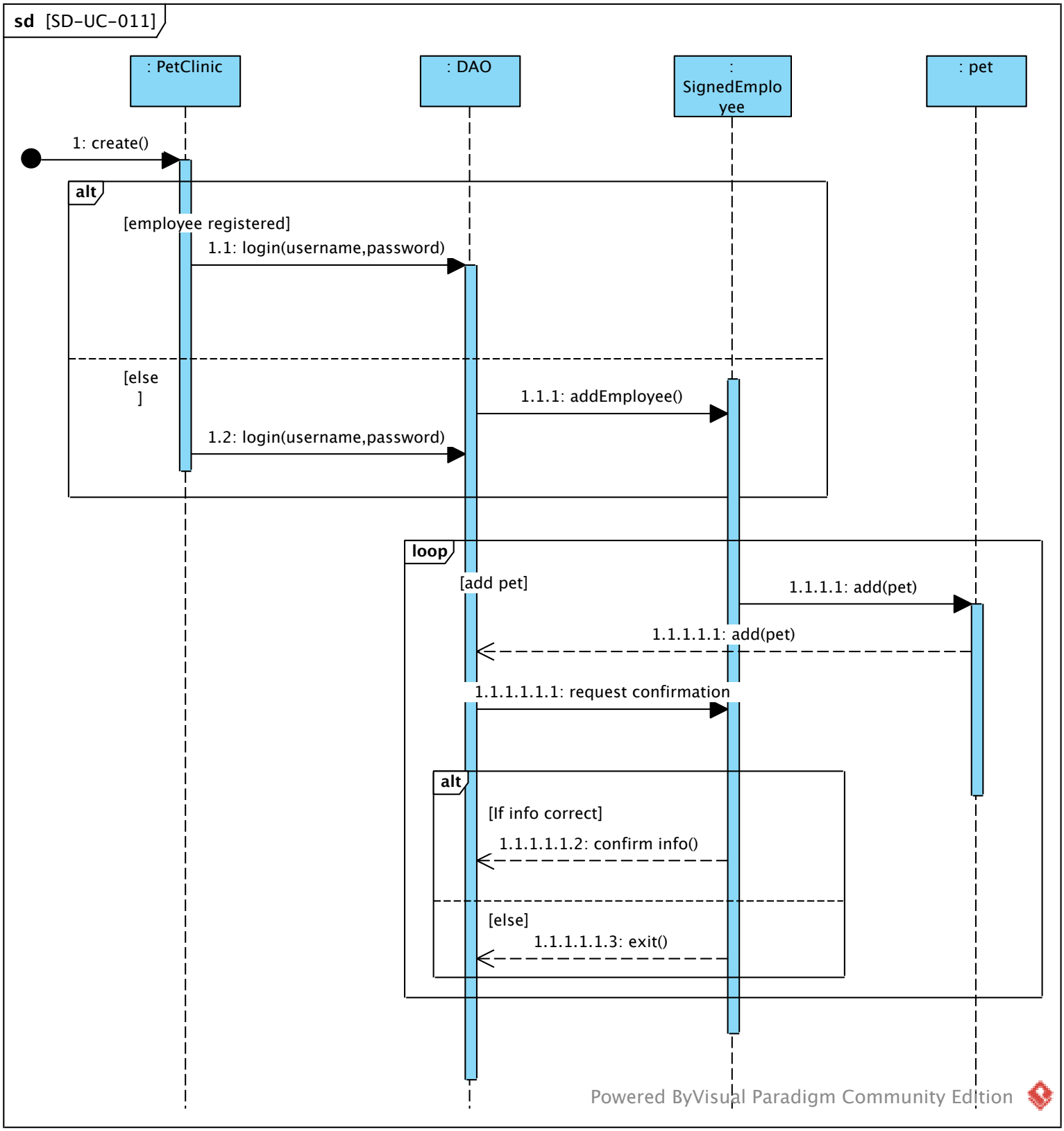

: PetClinic
pets : Set
employees : Set
customers : Set
1: export
1.1: openPetFile()
loop
[for pets]
2: getPet()
2.1: write(pet)
2.2: closePetFile()
2.3: openEmployeeFile()
loop
[for employees]
3: getEmployee()
3.1: write(employee)
3.2: closeEmployeeFile()
3.3: openCustomerFile()
loop
[for customers]
4: getCustomer()
4.1: write(customer)
4.2: closeCustomerFile()
Powered By Visual Paradigm Community Edition

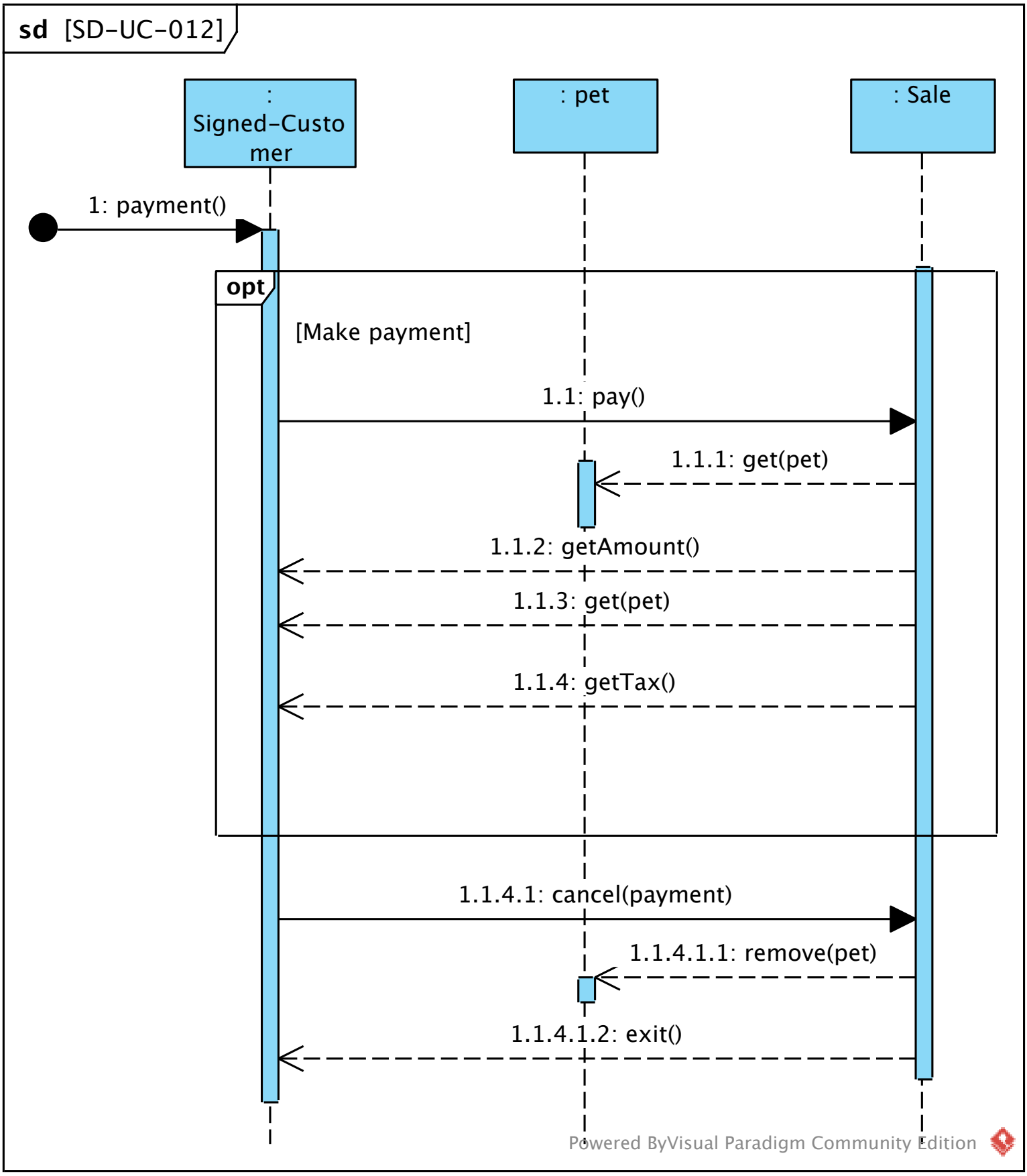

: PetClinic
: DAO
: Employee
: Sale
customer : Customer
pet : Pet
1: create()
alt
[employee has registered]
1.1: login(username, password)
[employee not registered]
1.1.1: addEmployee()
1.1.2: setIsManager(false)
1.2: login(username, password)
1.2.1: addCustomer()
1.2.2: create()
1.2.2.1: create()
1.2.2.1.1: selectCustomer(customer)
1.2.2.1.2: create()
loop
1.2.2.1.3: addPet(pet)
1.2.2.1.4: getAmount()
1.2.2.1.5: getDate()
1.2.2.1.6: getTax()
Powered By Visual Paradigm Community Edition

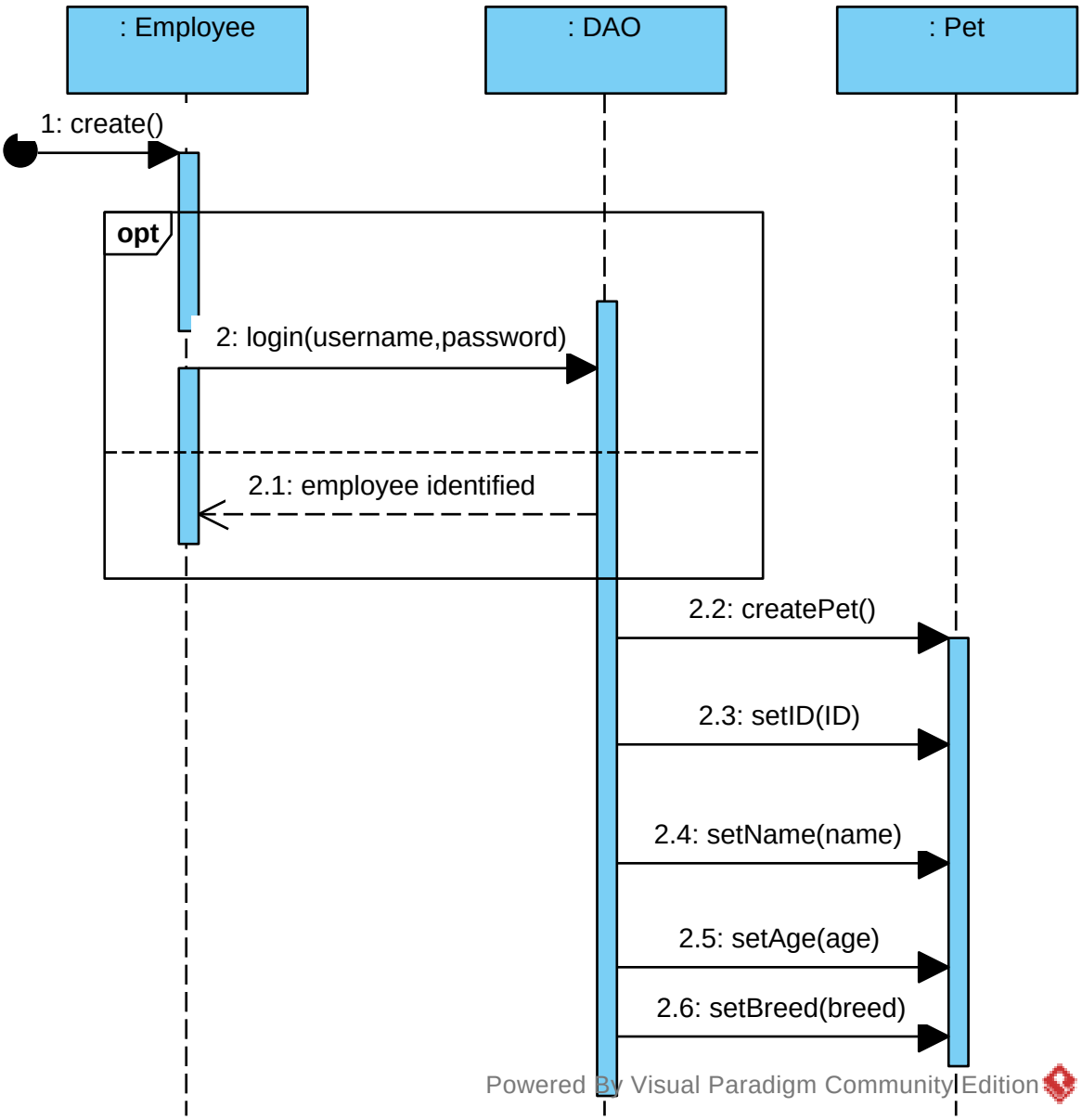

: PetClinic
: DAO
: Employee
: Sale
customer : Customer
pet : Pet
1: create()
alt
[employee has registered]
1.1: login(username, password)
[employee not registered]
1.1.1: addEmployee()
1.1.2: setIsManager(false)
1.2: login(username, password)
1.2.1: create()
1.2.1.1: create()
1.2.1.1.1: selectCustomer(customer)
1.2.1.1.2: create()
loop
[while there are pets choosen]
1.2.1.1.3: addPet(pet)
1.2.1.1.4: getAmount()
1.2.1.1.5: getDate()
1.2.1.1.6: getTax()
Powered By Visual Paradigm Community Edition

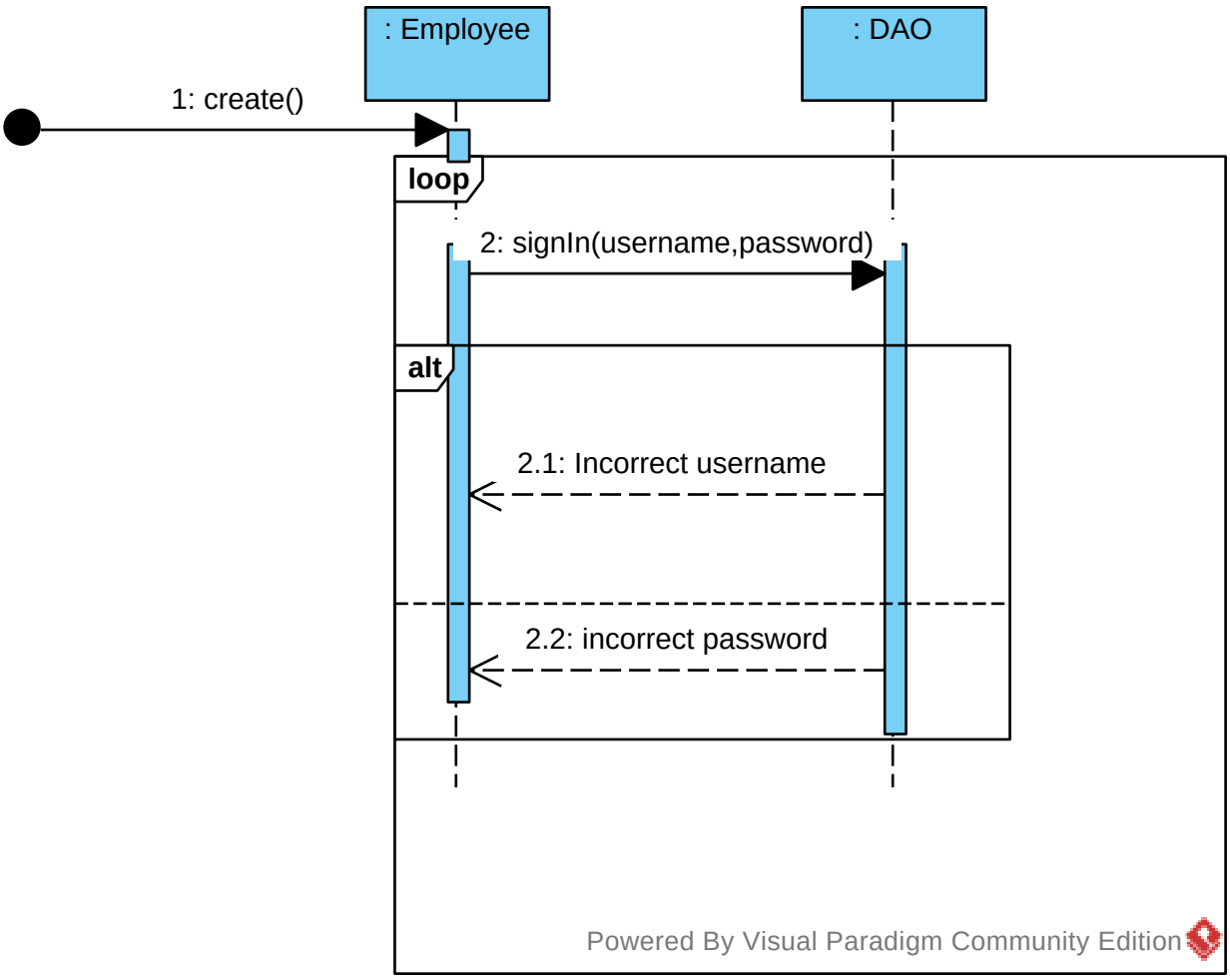

: PetClinic
: DAO
: Employee
: Sale
customer : Customer
pet : Pet
1: create()
alt
[Manager has registered]
1.1: login(username, password)
[Manager not registered]
1.1.1: addEmployee()
1.1.2: setIsManager(true)
1.2: login(username, password)
1.2.1: addCustomer()
1.2.2: create()
1.2.2.1: create()
1.2.2.1.1: selectCustomer(customer)
1.2.2.1.2: create()
loop
1.2.2.1.3: addPet(pet)
1.2.2.1.4: getAmount()
1.2.2.1.5: getDate()
1.2.2.1.6: getTax()
Powered By Visual Paradigm Community Edition

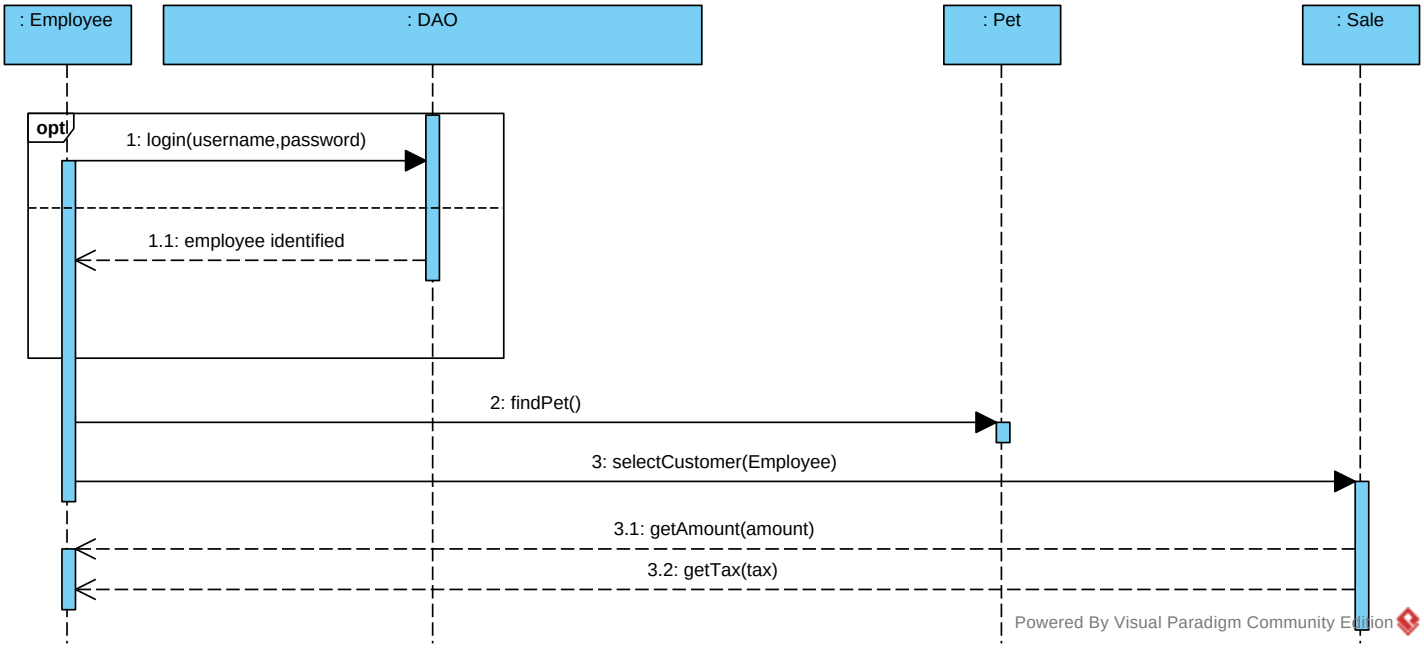

sd [SD-UC-010]
: Employee
: DAO
: SingedCustomer
: Pet
: Sale
1: create()
opt
[Employee has registered]
1.1: login(username,password)
1.2: employee Identified
loop
[Apply Check-out]
1.3: selectCustomer(SignedCustomer)
1.3.1: get(Customer)
1.3.2: select(pet)
1.3.2.1: get(pet)
1.3.2.1.1: request confirmation
1.3.2.1.1.1: confirm info(customer,pet)
alt
[make payment]
1.3.3: pay()
1.3.3.1: get(pet)
1.3.3.2: getAmount()
1.3.3.3: getDate()
1.3.3.4: getTax()
[cancel payment]
1.3.3.4.1: exit()
Powered ByVisual Paradigm Community Edition


sd [SD-UC-011]
: PetClinic
: DAO
: SignedEmployee
: pet
1: create()
alt
[employee registered]
1.1: login(username,password)
[else ]
1.1.1: addEmployee()
1.2: login(username,password)
loop
[add pet]
1.1.1.1: add(pet)
1.1.1.1.1: add(pet)
1.1.1.1.1.1: request confirmation
alt
[If info correct]
1.1.1.1.1.2: confirm info()
[else]
1.1.1.1.1.3: exit()
Powered ByVisual Paradigm Community Edition


sd [SD-UC-012]
: Signed-Customer
: pet
: Sale
1: payment()
opt
[Make payment]
1.1: pay()
1.1.1: get(pet)
1.1.2: getAmount()
1.1.3: get(pet)
1.1.4: getTax()
1.1.4.1: cancel(payment)
1.1.4.1.1: remove(pet)
1.1.4.1.2: exit()
Powered ByVisual Paradigm Community Edition


: Employee
: DAO
: Pet
1: create()
opt
2: login(username,password)
2.1: employee identified
2.2: createPet()
2.3: setID(ID)
2.4: setName(name)
2.5: setAge(age)
2.6: setBreed(breed)
Powered By Visual Paradigm Community Edition


: Employee
: DAO
1: create()
loop
2: signIn(username,password)
alt
2.1: Incorrect username
2.2: incorrect password
Powered By Visual Paradigm Community Edition


: Employee
: DAO
: Pet
: Sale
opt
1: login(username,password)
1.1: employee identified
2: findPet()
3: selectCustomer(Employee)
3.1: getAmount(amount)
3.2: getTax(tax)
Powered By Visual Paradigm Community Edition

# Test Coverage Plan

1. Main Menu Testing
    a. Import an existing file.
    b. Import a nonexistent file.
    c. Import a null value.
    d. Export to an existing file.
    e. Export to a new file.
    f. Export to a null value.
2. Home Page Testing
    a. Login with an existing user.
    b. Login with an invalid username and/or password.
    c. Login with null values.
    d. Attempt to open a database/inventory without logging in.
    e. Open the pet inventory.
    f. Open the employee database.
    g. Open the customer database.
3. Pet Inventory Testing
    a. Add a new, valid pet to the inventory.
    b. Attempt to add an invalid/null pet to the inventory.
    c. Edit an existing pet, giving valid edits.
    d. Edit an existing pet, giving invalid/null edits.
    e. Remove a pet from the inventory.
    f. Cancel the removal of a pet from the inventory.
    g. Save the edits.
    h. Cancel all the edits.
4. Employee Database Testing
    a. Add a new, valid employee to the database.
    b. Attempt to add an invalid/null employee to the database.
    c. Edit an existing employee, giving valid edits.
    d. Edit an existing employee, giving invalid edits.
    e. Remove an employee from the database.

    f. Cancel the removal of an employee.
    g. Save the edits.
    h. Cancel the edits.
5. Customer Database Testing
    a. Add a new, valid customer to the database.
    b. Attempt to add an invalid/null customer to the database.
    c. Edit an existing customer's information.
    d. Attempt to edit an existing customer with invalid/null edits.
    e. Remove a customer from the database.
    f. Cancel the removal of a customer.
    g. Save the edits.
    h. Cancel the edits.
6. Sale Testing
    a. Perform a sale with a customer adopting one or more pets.
    b. Perform a sale with an employee adopting one or more pets.
    c. Perform a sale with a manager adopting one or more pets.

CSI 3471 Team project

STAY_NEGATIVE

**Dante Hart**

Time burned: 40 hours

Available time: Every day from 4pm to 11pm

**Garrett Parker**

Time burned: 35 hours

Available time: 4 pm to 6 pm Monday to Saturday.

**Zane Pitzer**

Time burned: 32 hours

Available time: Every day from 4pm to 9pm

**Yihan Zhang**

Time burned: 35 hours

Available time: Every day from 12pm to 9pm

**Ruiqi Zhao**

Time burned: 35 hours

Available time: 4 pm to 10 pm Monday to Saturday.

**Jingke Shi**

Time burned: 35 hours

Available time: Every day from 4pm to 9pm

# Untitled Gantt Project

2022330

http://

| | |
|---|---|
| **Project manager** | Dante Hart |
| **Project dates** | 202228 - 2022419 |
| **Completion** | 91% |
| **Tasks** | 24 |
| **Resources** | 6 |

# Tasks

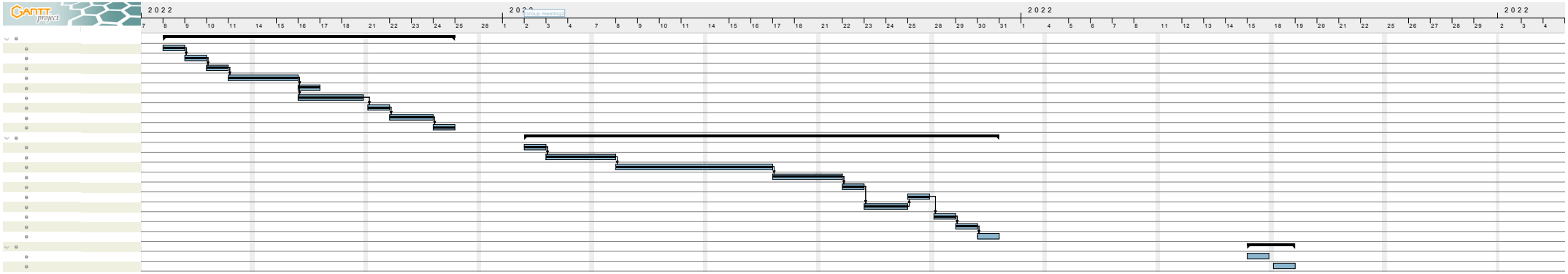
| Name | Begin date | End date |
|---|---|---|
| Iteration 1 | 2022/2/8 | 2022/2/24 |
| Group meeting 1 | 2022/2/8 | 2022/2/8 |
| Project visition | 2022/2/9 | 2022/2/9 |
| requirements | 2022/2/10 | 2022/2/10 |
| Use cases | 2022/2/11 | 2022/2/15 |
| traceability matrix | 2022/2/16 | 2022/2/16 |
| domain model | 2022/2/16 | 2022/2/18 |
| User Interface wireframes | 2022/2/21 | 2022/2/21 |
| Gantt DIagram | 2022/2/22 | 2022/2/23 |
| Presentation | 2022/2/24 | 2022/2/24 |
| | | |
| Iteration 2 | 2022/3/2 | 2022/3/30 |
| Group meeting2 | 2022/3/2 | 2022/3/2 |
| Design class | 2022/3/3 | 2022/3/7 |
| Demo design | 2022/3/8 | 2022/3/16 |
| GRASP | 2022/3/17 | 2022/3/21 |
| Test coverage plan | 2022/3/22 | 2022/3/22 |
| Issur tracking system | 2022/3/25 | 2022/3/25 |
| Gantt update | 2022/3/23 | 2022/3/24 |
| Report pdf | 2022/3/28 | 2022/3/28 |
| Group meeting3 | 2022/3/29 | 2022/3/29 |
| Presentation | 2022/3/30 | 2022/3/30 |
| | | |
| Iteration 3 | 2022/4/15 | 2022/4/18 |
| Group meeting4 | 2022/4/15 | 2022/4/15 |
| Final Presentation | 2022/4/18 | 2022/4/18 |

# Resources

| Name | Default role |
|---|---|
| Dante Hart | project manager |
| Garrett Parker | developer |
| Yihan Zhang | graphic designer |
| Jingke Shi | doc writer |
| Ruiqi Zhao | tester |
| Zane Pitzer | analysis |

# Gantt Chart





# Resources Chart