Zoo System

🐯A Zoo Management System lets you see different types of animals.

Table of Contents

Overview

Description

This application is a Zoo Management System where you can enter the zoo as a visitor, employee or the owner. As a visitor, you can visit the 6 environment and interact with the animals.

Use Cases

For Humans

Log into the System

1. Human logs in the system
2. The system displays welcome message with different options

   Owner, Visitor, and Employee
3. Human selects the appropriate position/identity
4. The system takes them (the human(s)) into appropriate pages

Owner's page, Visitor's page, and Employees page

Variation #1

1.1 In step 2, if Human does not make a selection

1.2 The system remains on the same page

1.3 Continue with step 2.

For Visitors

Visits the Zoo

1. Visitor Log into the system
2. It enters name and age
3. The system displays Park Visitation Screen which shows the 6 environments
4. It clicks on one of the environment
5. Visitor selects on an animal
6. Visitor is then lead to a page where it can interact with the animal
7. Visitor can go back to the environment page

Variation #1

2.1 In step 2, Visitor enters nothing

2.2 The System will give an error on the screen

2.3 Displays, "Some areas are blank!" in red.

Variation #2

7.1 It can go back to the Welcome page where the options are shown

For Employees

1. Employee enters the zoo

2. The System displays options, Visitor, Employee, Owner
3. The Employee clicks on Employee button
4. System shows email and password textfield.
5. The Employee enters email and password
6. System confirms email and passwords and leads the employee into Employee details page which shows the employees different attributes.
7. The employee then decides to visit the Zoo Environments page
8. The employee selects and environment
9. The employee selects an animal
10. The employee interacts with the animal
11. Then it goes back to the Zoo Welcome page

Variation #1

4.1 Employee enters wrong email or password

4.2 System does not let the employee in

For Owner

Checks the System

1. The owner enters the Zoo
2. Zoo System displays three different options
3. Owner clicks on Owner Option
4. System displays email and password
5. Enters “0” and “0” for both fields
6. System displays owner options
7. Owner can go back to Zoo welcome page

Variation #1

6.1 Owner selects collect pay from Visitors

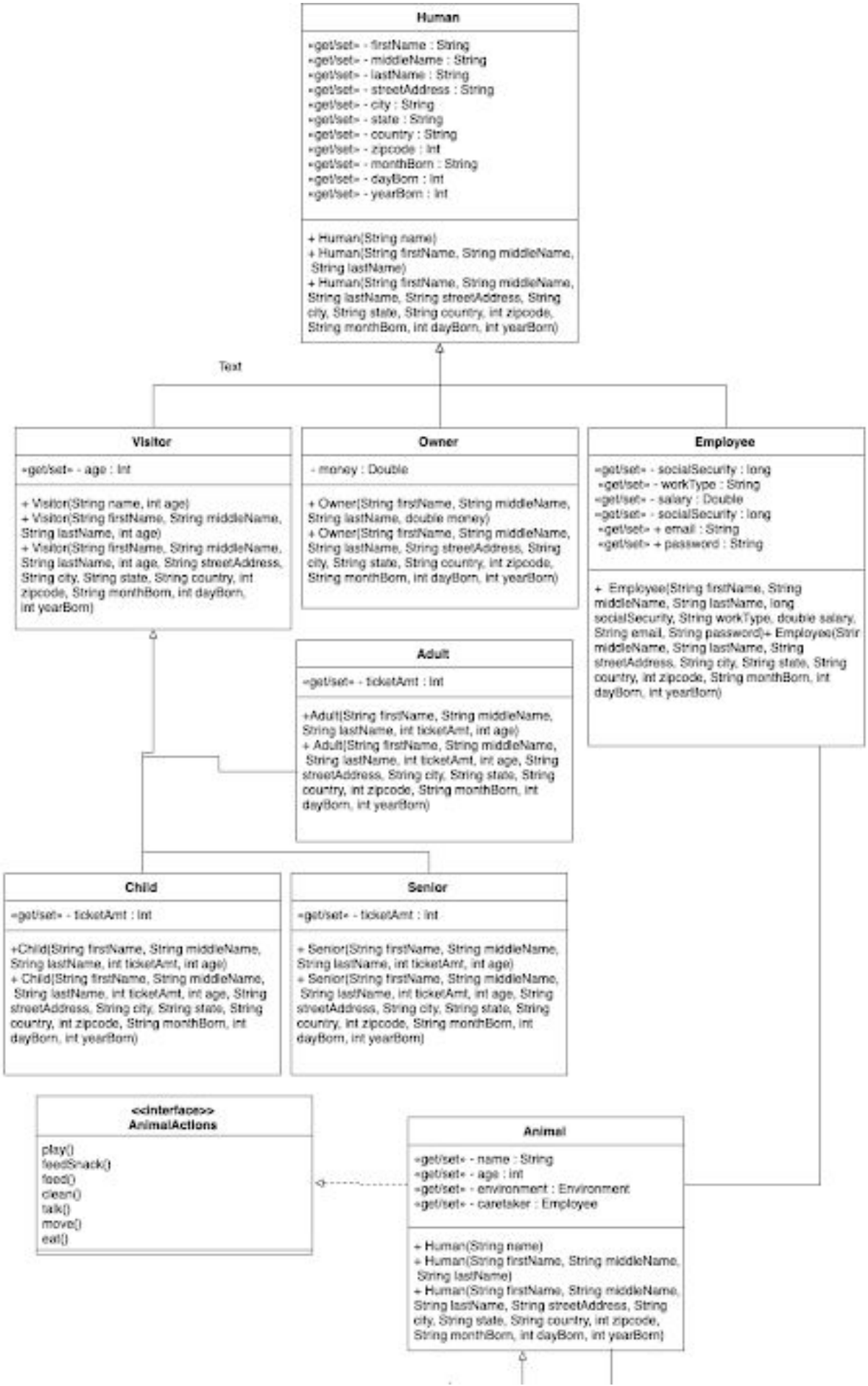
6.2 System displays “$0” and confirms that it has collected money

Variation #2

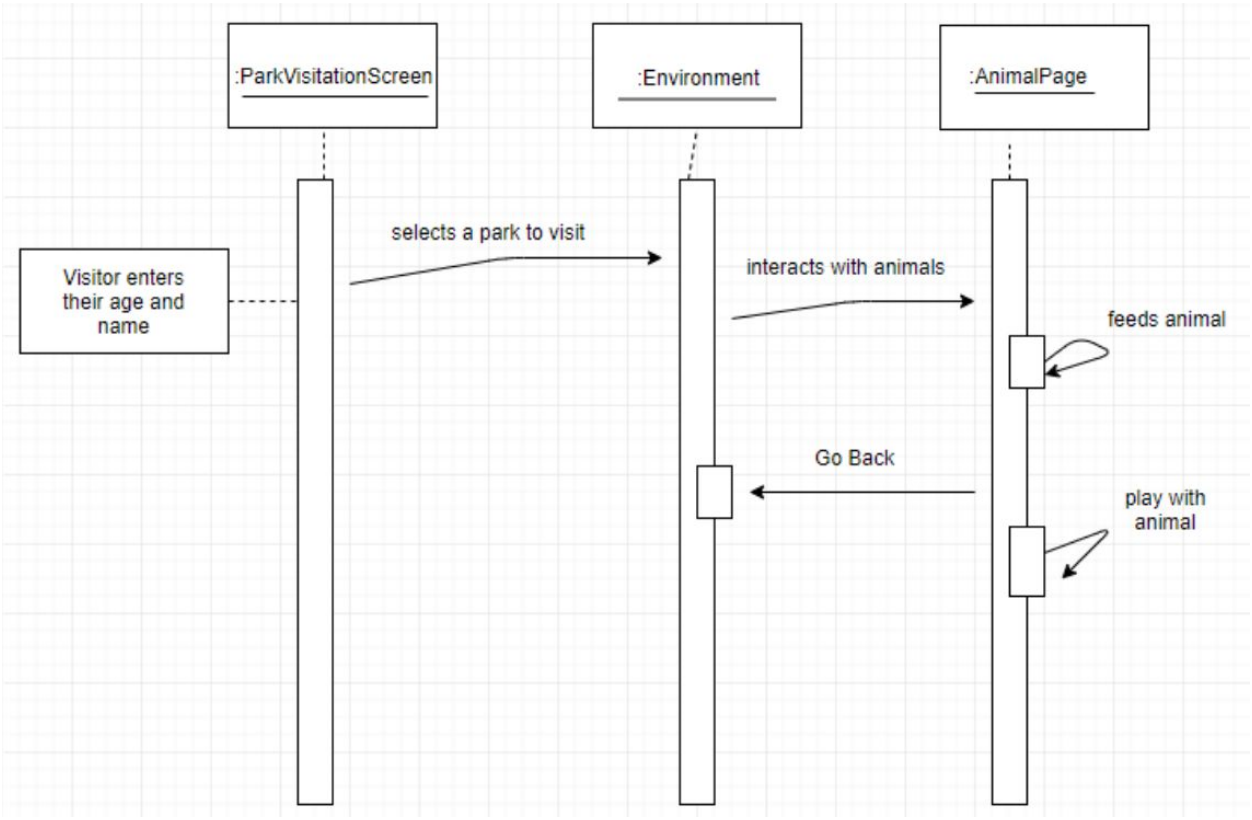
6.1 Owner selects to pays employees

6.2 System displays that it has paid employees

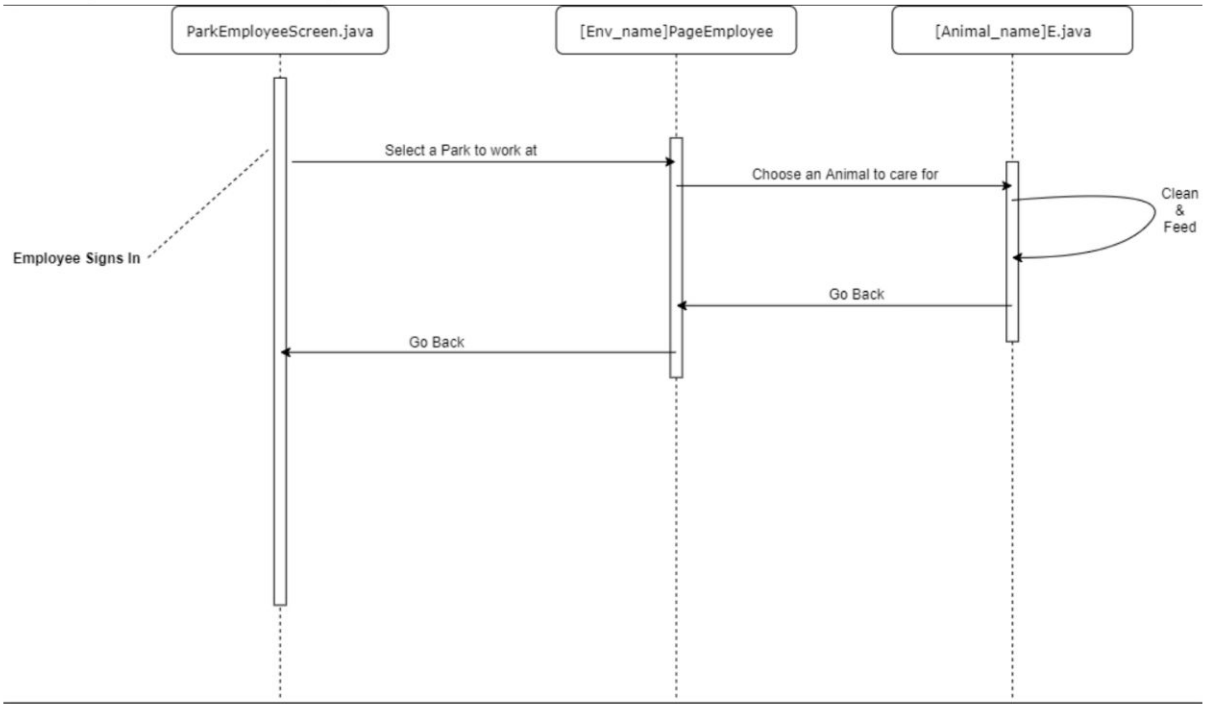
UML Diagrams

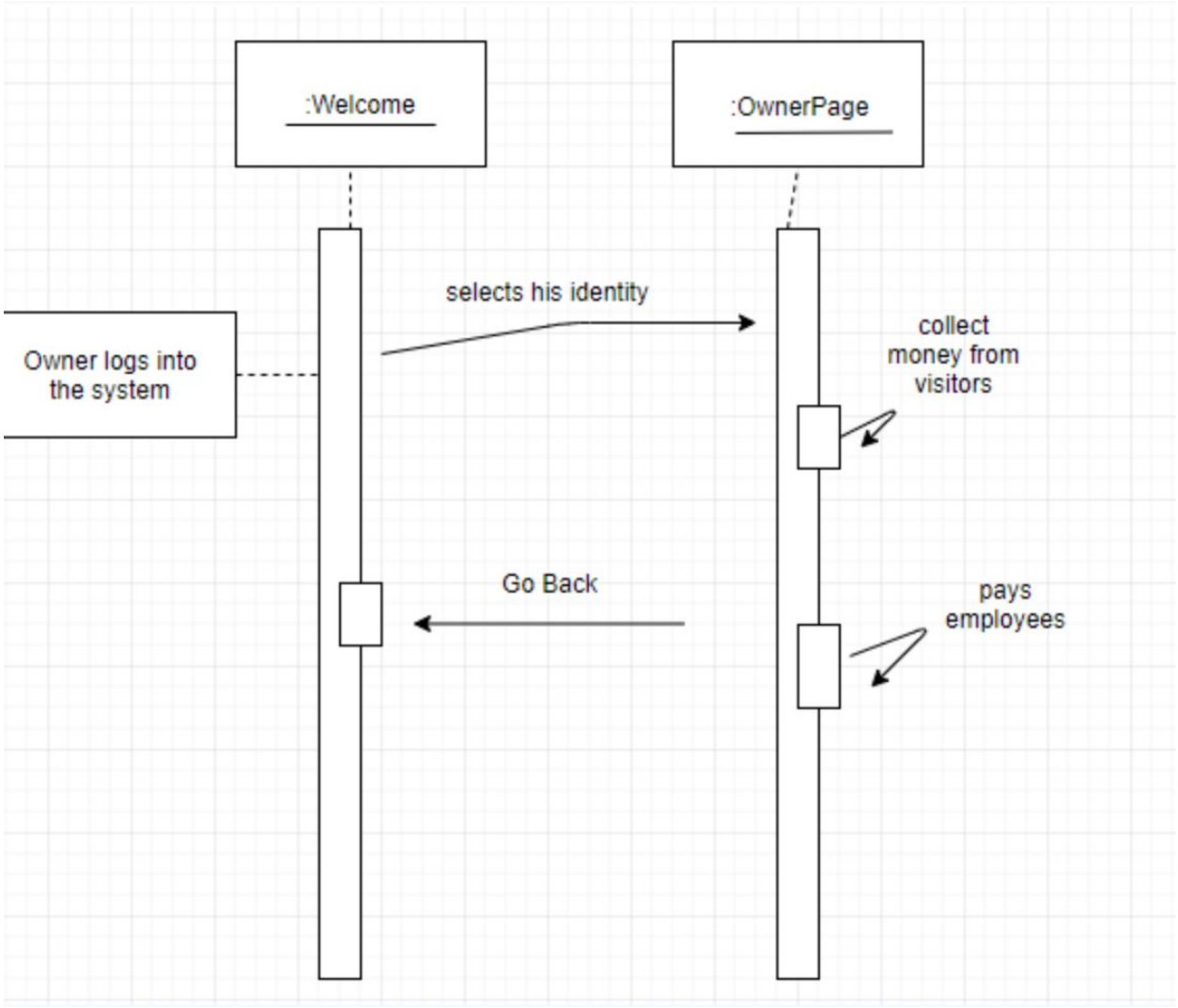

Human
«get/set» - firstName : String
«get/set» - middleName : String
«get/set» - lastName : String
«get/set» - streetAddress : String
«get/set» - city : String
«get/set» - state : String
«get/set» - country : String
«get/set» - zipcode : Int
«get/set» - monthBorn : String
«get/set» - dayBorn : Int
«get/set» - yearBorn : Int
+ Human(String name)
+ Human(String firstName, String middleName, String lastName)
+ Human(String firstName, String middleName, String lastName, String streetAddress, String city, String state, String country, int zipcode, String monthBorn, int dayBorn, int yearBorn)
Text
Visitor
«get/set» - age : Int
+ Visitor(String name, int age)
+ Visitor(String firstName, String middleName, String lastName, int age)
+ Visitor(String firstName, String middleName, String lastName, int age, String streetAddress, String city, String state, String country, int zipcode, String monthBorn, int dayBorn, int yearBorn)
Owner
- money : Double
+ Owner(String firstName, String middleName, String lastName, double money)
+ Owner(String firstName, String middleName, String lastName, String streetAddress, String city, String state, String country, int zipcode, String monthBorn, int dayBorn, int yearBorn)
Employee
«get/set» - socialSecurity : long
«get/set» - workType : String
«get/set» - salary : Double
«get/set» - socialSecurity : long
«get/set» + email : String
«get/set» + password : String
+ Employee(String firstName, String middleName, String lastName, long socialSecurity, String workType, double salary, String email, String password)+ Employee(Strir middleName, String lastName, String streetAddress, String city, String state, String country, int zipcode, String monthBorn, int dayBorn, int yearBorn)
Adult
«get/set» - ticketAmt : Int
+Adult(String firstName, String middleName, String lastName, int ticketAmt, int age)
+ Adult(String firstName, String middleName, String lastName, int ticketAmt, int age, String streetAddress, String city, String state, String country, int zipcode, String monthBorn, int dayBorn, int yearBorn)
Child
«get/set» - ticketAmt : Int
+Child(String firstName, String middleName, String lastName, int ticketAmt, int age)
+ Child(String firstName, String middleName, String lastName, int ticketAmt, int age, String streetAddress, String city, String state, String country, int zipcode, String monthBorn, int dayBorn, int yearBorn)
Senior
«get/set» - ticketAmt : Int
+ Senior(String firstName, String middleName, String lastName, int ticketAmt, int age)
+ Senior(String firstName, String middleName, String lastName, int ticketAmt, int age, String streetAddress, String city, String state, String country, int zipcode, String monthBorn, int dayBorn, int yearBorn)
<<interface>>
AnimalActions
play()
feedSnack()
feed()
clean()
talk()
move()
eat()
Animal
«get/set» - name : String
«get/set» - age : int
«get/set» - environment : Environment
«get/set» - caretaker : Employee
+ Human(String name)
+ Human(String firstName, String middleName, String lastName)
+ Human(String firstName, String middleName, String lastName, String streetAddress, String city, String state, String country, int zipcode, String monthBorn, int dayBorn, int yearBorn)

Sequence Diagrams



Employee Sequence Diagram





Design Patterns Applied

Iterator

- To iterate the list of Employees to get the Sign In
- Check SignInEmployee.java to see how we loop through

Strategy

- We have an interface AnimalActions.java
- Utilizes static methods to implement the 'STRATEGY' pattern
- Each animal object implements said interface and rewrites the methods

MVC Model View Controller

- We separated the Model, View and the Controller
    - Model is where the data is
    - View, we used JavaFx to display the views and uses the Controller
    - Controller handles the input from the user and manipulates the data and view to update.

Components

We have one file for all the implementation We have another file for images