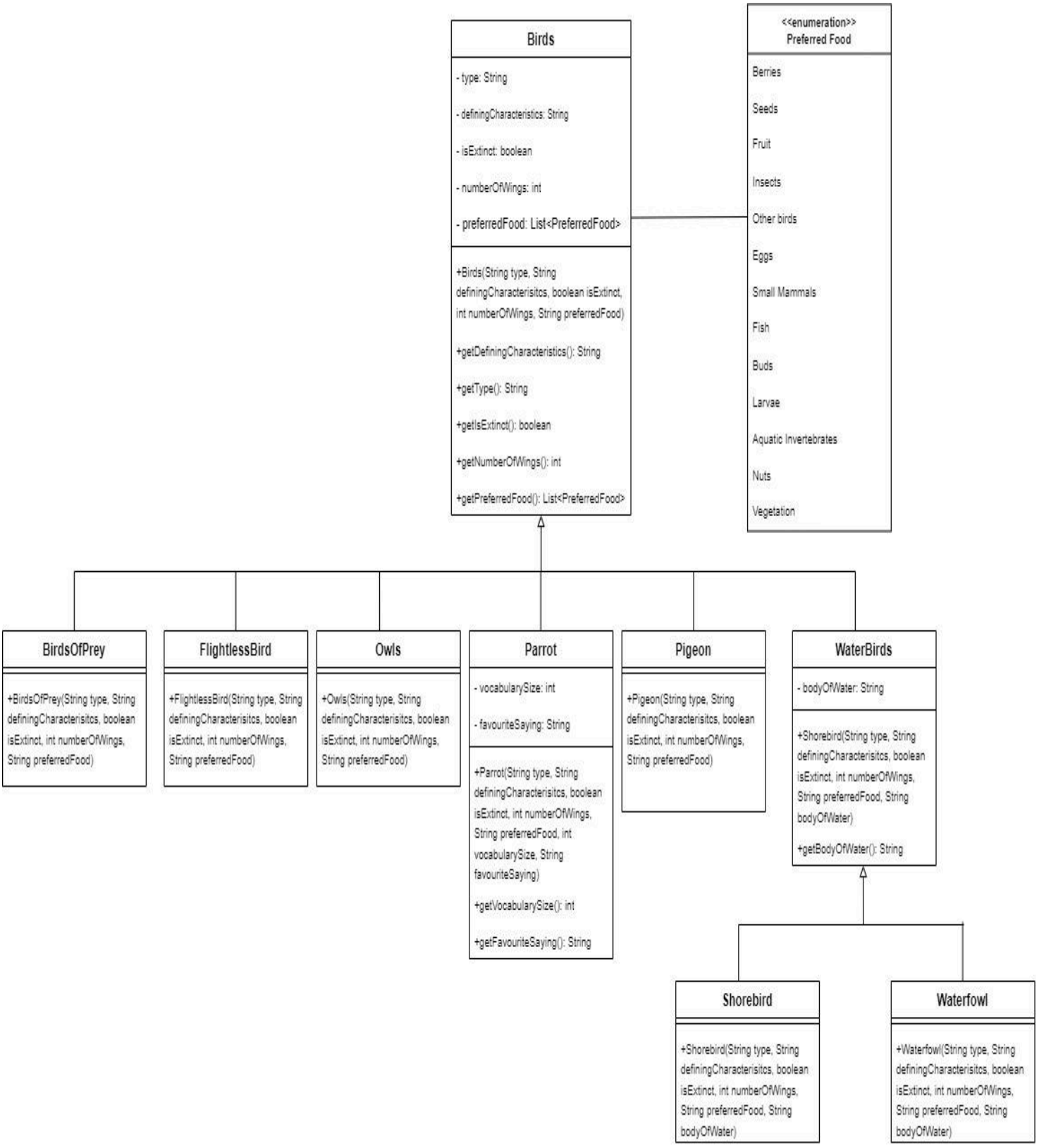
# PDP Lab 1-Birds

## UML Diagram:



# Birds Class Design Walkthrough:

## Encapsulation of Different Types of Birds:

To encapsulate the various bird types, the design employs a hierarchical class structure.

1. **Base Class (Birds)**: This serves as a general blueprint for all birds, defining shared attributes like type, characteristics, extinction status, wing count, and diet preferences.
2. **Derived Classes**: Subclasses such as BirdsOfPrey, FlightlessBird, WaterBirds, Parrot, and Pigeon inherit from the base class and add specialized behaviors and attributes. For example:
   - **BirdsOfPrey**: Represents birds with specific hunting traits.
   - **FlightlessBird**: Captures unique features of non-flying birds.
   - **WaterBirds**: Adds a field for the type of water body associated with these birds, further subclassed into Shorebird and Waterfowl.
   - **Parrot**: Adds unique properties like vocabulary size and favorite saying.

## Fields in Each Class:

Each class contains fields tailored to the specific type of bird. Here's a detailed breakdown:

1. **Base Class: Birds**

a. **Fields**:
   - type (String, private): Specifies the bird species or group.
   - definingCharacteristics (String, private): Describes unique traits.
   - isExtinct (Boolean, private): Indicates extinction status.
   - numberOfWings (int, private): Stores the wing count.
   - preferredFood (List<String>, private): Enumerates the bird's diet.

2. **BirdsOfPrey**

a. **Fields**: Inherits all fields from Birds.
   - Adds no additional fields as general bird traits suffice to capture the uniqueness of birds of prey.

3. **FlightlessBird**

a. **Fields**: Inherits all fields from Birds.
   - No new fields; these birds are differentiated by their definingCharacteristics and extinction status.

4. **Parrot**
a. **Fields**:
    - vocabularySize (int, private): Tracks the parrot's vocabulary size.
    - favouriteSaying (String, private): Stores a commonly repeated phrase.

5. **Pigeon**
    - **Fields**: Inherits from Birds.
    - No new fields; pigeons are distinguished by their specific characteristics.

6. **WaterBirds**
a. **Fields**:
    - bodyOfWater (String, private): Indicates the type of water body (e.g., lake, ocean) the bird is associated with.

7. **Shorebird and Waterfowl**
a. **Fields**: Inherit all fields from WaterBirds.

### Access and Justification:

**Private Access**: All fields are private to enforce encapsulation, ensuring that data integrity is maintained and accidental modifications are prevented.

# Test Plan

## 1. Bird Class

### a. Test Case: Creating a Valid Bird

- **Purpose**: Verify that a Birds object can be successfully created when all input parameters are valid.
- **Outcome**: Confirms that valid input produces a properly initialized Birds object.

### b. Test Case: Creating a Bird with an Invalid Type

- **Purpose**: Validate that an exception is thrown when an empty or null string is passed as the bird's type.
- **Outcome**: Ensures that the class enforces the type requirement and prevents the creation of invalid birds.

### c. Test Case: Creating a Bird with Invalid Defining Characteristics

- **Purpose**: Ensure that defining characteristics cannot be empty or null during object creation.
- **Outcome**: Confirms that the defining characteristics field is validated during bird creation.

### d. Test Case: Creating a Bird with a Negative Number of Wings

- **Purpose**: Verify that a bird cannot be created with a negative number of wings.
- **Outcome**: Confirms that the number of wings is validated to prevent invalid data.

### e. Test Case: Creating a Bird with an Invalid Preferred Food List Size

- **Purpose**: Ensure that the preferred food list must contain between 2 and 4 items.
- **Outcome**: Validates that the preferred food list is enforced to meet the specified range of items.

## 2. Additional Tests for Specific Bird Types

### a. Test Case: Vocabulary Size of Parrots

- **Purpose**: Validate that the vocabularySize property for parrots is correctly set.
- **Outcome**: Confirms that the vocabulary size is correctly initialized and accurately returned by the method.

### b. Test Case: Favorite Saying of Parrots

- **Purpose**: Verify that the favouriteSaying property of parrots is correctly set and retrievable.
- **Outcome**: Ensures that parrots' favorite sayings are properly stored and accessible through the appropriate method.

**c. Test Case: Body of Water for Waterbirds**

- **Purpose**: Ensure that the bodyOfWater property is correctly assigned to shorebirds and waterfowls.
- **Outcome**: Verifies that each type of waterbird is assigned the correct body of water and that the property is retrievable.

## 3. Bird Conservatory Tests

**a. Test Case: Adding Aviaries to the Conservatory**

- **Purpose**: Verify that aviaries can be added to the conservatory and that duplicate aviary names are not allowed.
- **Outcome**:The first aviary is added successfully and the duplicate aviary is rejected.

**b. Test Case: Adding a Null Aviary Name**

- **Purpose**: Ensure that adding an aviary with a null name throws an appropriate exception.
- **Outcome**:The test confirms that null aviary names are not allowed and that the exception is handled properly

**c.Test Case: Rescuing Birds into an Aviary**

- **Purpose**: Verify that birds can be successfully rescued and added to an aviary, and their details are accurately stored.
- **Outcome**:Birds are successfully added to the aviary and aviary's bird list reflects the correct count and order of birds.

**d. Test Case: Rescuing a Bird with Null Values**

- **Purpose**: Ensure that rescuing a bird with invalid input (null aviary name or bird) throws an appropriate exception.
- **Outcome**:Confirms that invalid inputs are handled correctly, maintaining system integrity.

**e. Test Case: Rescuing an Extinct Bird in an Existing Aviary**

- **Purpose**: Validate that extinct birds can be added to existing aviaries.
- **Outcome**:Extinct birds can be added to existing aviaries without errors and the bird list accurately updates with the new addition.

**f. Test Case: Rescuing an Extinct Bird in a New Aviary**

- **Purpose**: Validate that rescuing an extinct bird into a new aviary does not allow the bird to be added.
- **Outcome**:Extinct birds cannot be rescued into new aviaries and the aviary remains empty after the rescue attempt.

**g. Test Case: Rescuing a Mix of Bird Types**

- **Purpose**: Ensure that rescuing multiple bird types into the same aviary works as expected.
- **Outcome**:The aviary can hold multiple bird types, provided capacity constraints are met.

**h. Test Case: Rescuing a Mixed Bird into a New Aviary**

- **Purpose**: Validate that mixed bird types can be added to newly created aviaries.
- **Outcome**:Birds are correctly allocated to the intended aviary and mixed bird types can be accommodated in new aviaries.

**i. Test Case: Calculating Food Requirements for Rescued Birds**

- **Purpose**: Ensure the conservatory can calculate the aggregated food requirements for all rescued birds.
- **Outcome**:Food requirements are accurately calculated and aggregated across aviaries.

**j. Test Case: Finding an Aviary by Bird Type**

- **Purpose**: Verify that the system can locate an aviary containing a specific bird type.
- **Outcome**:The correct aviary is identified for present birds and appropriate messages are returned for birds not found in any aviary.

**k. Test Case: Generating an Aviary Sign**

- **Purpose**: Ensure aviary signs are generated correctly, displaying bird information.
- **Outcome**:Aviary signs correctly display the list of birds and edge cases like empty or non-existent aviaries are handled gracefully.

**l. Test Case: Generating a Map of Aviaries**

- **Purpose**: Verify that the conservatory generates a detailed map of all aviaries, including their names and birds.
- **Outcome**:The map includes all aviary names and their birds, the birds that do not meet constraints are not added.

**m. Test Case: Generating an Index of Birds in Aviaries**

- **Purpose**: Ensure that the conservatory can create an index showing the locations of all birds across aviaries.
- **Outcome**:The index accurately lists all birds and their respective aviary locations.

## 4. Aviary Tests

**a. Test Case: Adding More Aviaries Than the Limit**

- **Purpose**: Ensure the conservatory enforces the maximum number of aviaries allowed.
- **Outcome**:The conservatory allows only the predefined maximum number of aviaries and the additional aviaries are not added beyond this limit.

**b. Test Case: Checking the Capacity of an Aviary**

- **Purpose**: Verify that the capacity of an aviary is correctly calculated and enforced.
- **Outcome**:The capacity is correctly reported as full when 5 birds are added and invalid inputs are gracefully handled.

**c. Test Case: Counting the Current Number of Birds in an Aviary**

- **Purpose**: Ensure that the current count of birds in an aviary is accurately tracked.
- **Outcome**:The current bird count matches the number of birds added to the aviary.

**d. Test Case: Adding Birds Exceeding Aviary Capacity**

- **Purpose**: Confirm that the aviary does not accept birds beyond its maximum capacity.
- **Outcome**:The aviary rejects additional birds once its capacity is full.

**e. Test Case: Getting a Description of the Aviary**

- **Purpose**: Validate that the aviary description accurately lists its bird occupants.
- **Outcome**:The description correctly reflects the bird occupants in the aviary.