LAB 1 - BIRDS
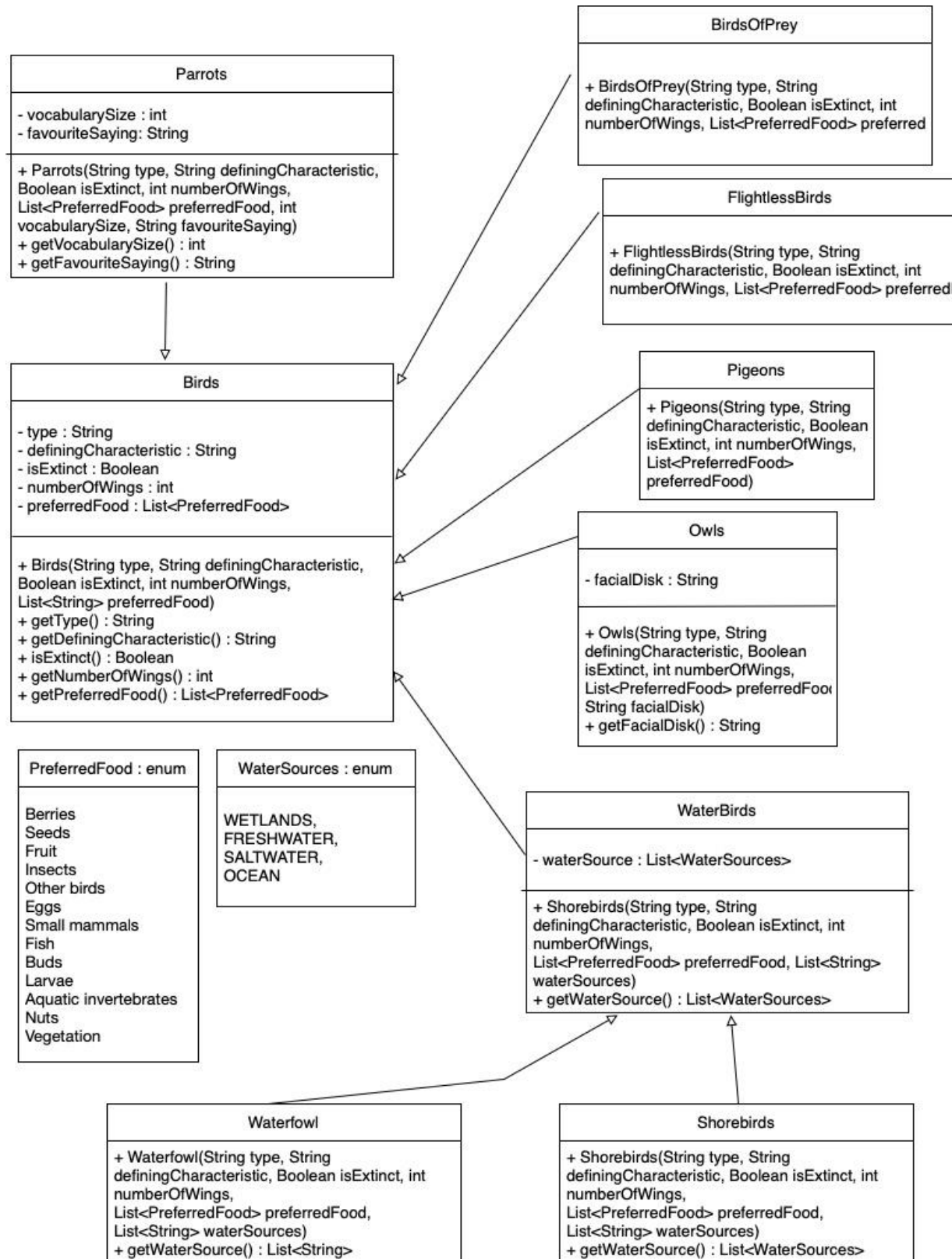
CS5010 – Programming Design Paradigm

Abstract

UML Class diagram and testing plan for Part - 1

Kiruthiga Muralidharan

UML Class Diagram:

Parrots

- vocabularySize : int
- favouriteSaying: String

+ Parrots(String type, String definingCharacteristic, Boolean isExtinct, int numberOfWings, List<PreferredFood> preferredFood, int vocabularySize, String favouriteSaying)
+ getVocabularySize() : int
+ getFavouriteSaying() : String

BirdsOfPrey

+ BirdsOfPrey(String type, String definingCharacteristic, Boolean isExtinct, int numberOfWings, List<PreferredFood> preferred

FlightlessBirds

+ FlightlessBirds(String type, String definingCharacteristic, Boolean isExtinct, int numberOfWings, List<PreferredFood> preferred

Pigeons

+ Pigeons(String type, String definingCharacteristic, Boolean isExtinct, int numberOfWings, List<PreferredFood> preferredFood)

Birds

- type : String
- definingCharacteristic : String
- isExtinct : Boolean
- numberOfWings : int
- preferredFood : List<PreferredFood>

+ Birds(String type, String definingCharacteristic, Boolean isExtinct, int numberOfWings, List<String> preferredFood)
+ getType() : String
+ getDefiningCharacteristic() : String
+ isExtinct() : Boolean
+ getNumberOfWings() : int
+ getPreferredFood() : List<PreferredFood>

Owls

- facialDisk : String

+ Owls(String type, String definingCharacteristic, Boolean isExtinct, int numberOfWings, List<PreferredFood> preferredFood, String facialDisk)
+ getFacialDisk() : String

PreferredFood : enum

Berries
Seeds
Fruit
Insects
Other birds
Eggs
Small mammals
Fish
Buds
Larvae
Aquatic invertebrates
Nuts
Vegetation

WaterSources : enum

WETLANDS,
FRESHWATER,
SALTWATER,
OCEAN

WaterBirds

- waterSource : List<WaterSources>

+ Shorebirds(String type, String definingCharacteristic, Boolean isExtinct, int numberOfWings, List<PreferredFood> preferredFood, List<String> waterSources)
+ getWaterSource() : List<WaterSources>

Waterfowl

+ Waterfowl(String type, String definingCharacteristic, Boolean isExtinct, int numberOfWings, List<PreferredFood> preferredFood, List<String> waterSources)
+ getWaterSource() : List<String>

Shorebirds

+ Shorebirds(String type, String definingCharacteristic, Boolean isExtinct, int numberOfWings, List<PreferredFood> preferredFood, List<String> waterSources)
+ getWaterSource() : List<WaterSources>

Test Plan:

Test Method	Description	Input Assignment	Expected Output
testRescueBird ()	Tests if a bird is rescued and added to the library	Parrot("Parrot", "Has a cute voice", false, 2, List.of(PreferredFood.SEEDS, PreferredFood.FRUITS), 20, "Hey beautiful!");	Parrot
testAddBirdSuccess ()	Tests if the bird is added. (Aviary is within the capacity and there is no problem of coexistence)	Parrot("Parrot", "Has a cute voice", false, 2, List.of(PreferredFood.SEEDS, PreferredFood.FRUITS), 20, "Hey beautiful!"); Owl("Owl", "Has big eyes", false, 2, List.of(PreferredFood.SEEDS, PreferredFood.BUDS), "Round");	2 (No. of birds added)
testAviaryFull ()	Tests if the Aviary adds 6th bird	Parrot1, Parrot2, Parrot3, Parrot4, Parrot5, Parrot6 Aviary1	IllegalStateException
testCalculateFoodRequirements ()	Tests the food requirements for the conservatory	Parrot("Parrot", "Has a cute voice", false, 2, List.*of*(PreferredFood.*SEEDS*, PreferredFood.*FRUITS*), 20, "Hey beautiful!"); Owl("Owl", "Has big eyes", false, 2, List.*of*(PreferredFood.*FISH*, PreferredFood.*OTHER_BIRDS*), "Round");	Key: SEEDS, Value:1 Key: FRUITS, Value:1 Key: FISH, Value:1 Key: OTHER_BIRDS, Value: 1
testPreferredFoodListSize()	Tests if the food list size	Parrot("Parrot", "Has a cute voice", false, 2,	IllegalArgumentException

	is between 2 and 5	List.*of*(PreferredFood.*SEEDS*), 20, "Hey beautiful!")); false, 2, ["Berries", "Seeds"], ["Freshwater"]);	
testPrintMap()	Tests the map containing aviaries and the birds present in it.	Parrot("Parrot", "Has a cute voice", false, 2, List.*of*(PreferredFood.*SEEDS*, PreferredFood.*FRUITS*), 20, "Hey beautiful!"); Owl("Owl", "Has big eyes", false, 2, List.*of*(PreferredFood.*FISH*, PreferredFood.*OTHER_BIRDS*), "Round");	Aviary 1 – Parrot, Owl
testBirdIndexPrint()	Tests the birds conservatory index	Parrot("Parrot", "Has a cute voice", false, 2, List.*of*(PreferredFood.*SEEDS*, PreferredFood.*FRUITS*), 20, "Hey beautiful!"); Owl("Owl", "Has big eyes", false, 2, List.*of*(PreferredFood.*FISH*, PreferredFood.*OTHER_BIRDS*), "Round");	Parrot located in Aviary1 Owl located in Aviary 1
testAviaryLimit()	Tests the limit of aviaries (20)	List.of(aviary1...aviary21)	IllegalArgumentException
testPrintSign()	Tests the aviary sign	Parrot("Parrot", "Has a cute voice", false, 2, List.*of*(PreferredFood.*SEEDS*, PreferredFood.*FRUITS*), 20, "Hey beautiful!");	Aviary 1 Aviary of Birds!!! Parrot with its unique characteristic Has a cute voice!!

| testCoexistence() | Tests if the input birds can coexist together | BirdOfPrey("Prey1", "Predator", false, 2, List.*of*(PreferredFood.*SEEDS*, PreferredFood.*FRUITS*));

 FlightlessBird("Flightless1", "Flightless", false, 2, List.*of*(PreferredFood.*SEEDS*, PreferredFood.*FRUITS*));

 Only 1 aviary | IllegalArgumentException |