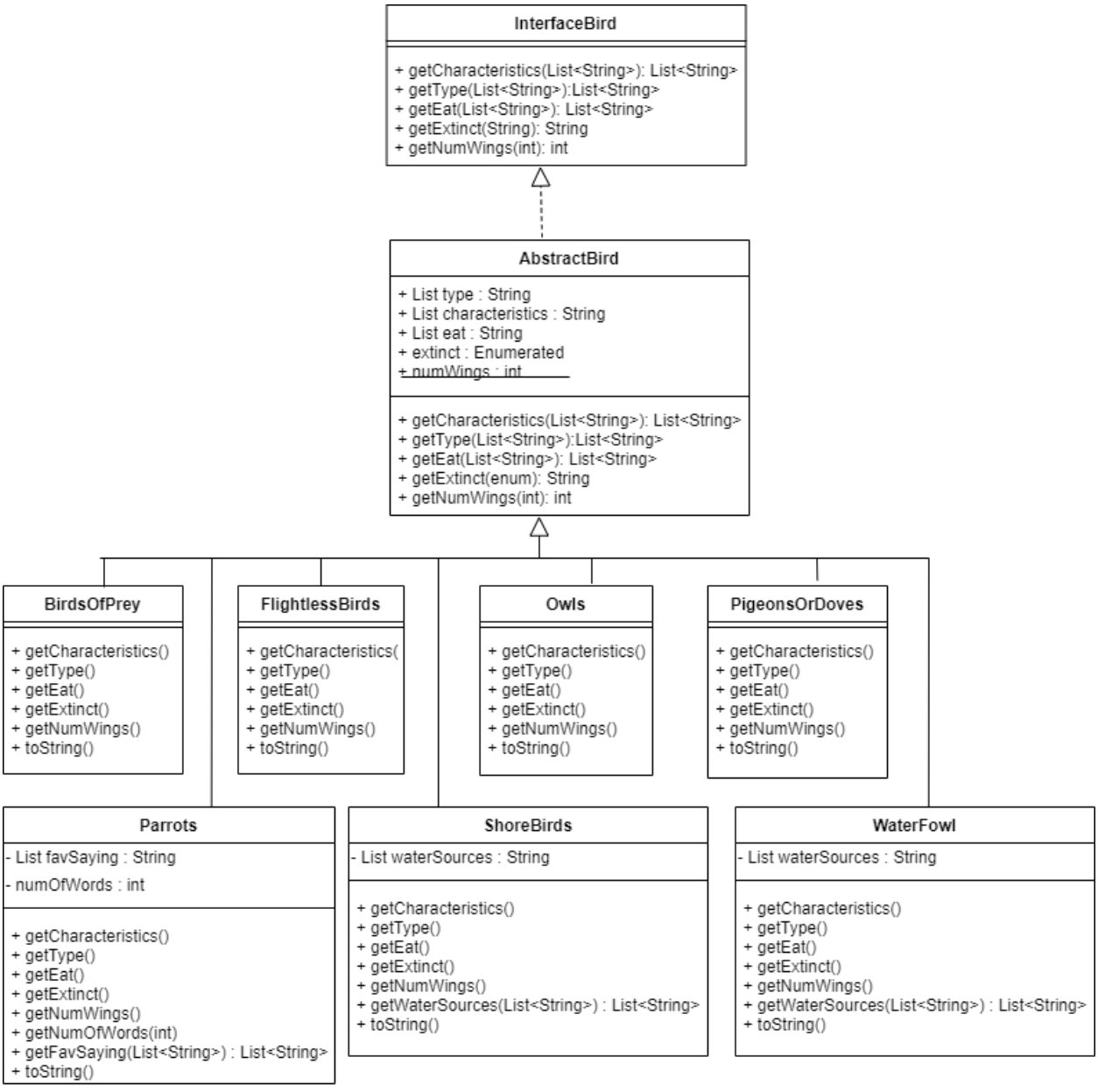
As all bird classification groups have common parameters like characteristics, type, eatList, extinct_or_no and number_of_wings; I have decided to declare the parameters in the abstract AbstractBird class. We create an object for each bird classification group and set the values through it. The values of getter methods can also be accessed using this object. The toString() method in the base classes combines the values from getter methods to display the collective value of the bird classification group.

In testing, we write test cases to see that we are able to retrieve the correct information for each bird classification group. The number of wings cannot be negative, hence it must throw an exception. The List<String> for characteristics, type and eat must also throw an exception if the parameter passed to is not String. Enumerated type Extinct can only take values yes, no and maybe; otherwise it throws an exception.



Testing Plan

@Test for Birds of Prey

assertEquals("Birds of Prey: Characteristics = sharp beak, hooked beak, visible nostrils Type = hawks, eagles, osprey Extinct = no Number of Wings = 2 Eat = other birds, small mammals, fish, aquatic invertebrates", bop.toString())

@Test for Number of wings (expected = IllegalArgumentException.class)
  public void BirdsOfPreyConstructorDisallowsNegativeValueOfWings() { bop.set({sharp beak, hooked beak, visible nostrils },{ hawks, eagles, osprey } , Extinct.no , -2,{ other birds, small mammals, fish, aquatic invertebrates }); }

@Test for List<String>Characteristics (expected = IllegalArgumentException.class)
  public void BirdsOfPreyConstructorDisallowsIntInStringList () { bop.set{1,2 }, { hawks, eagles, osprey } , Extinct.no, 2,{ other birds, small mammals, fish, aquatic invertebrates }); }

@Test for List<String>Type (expected = IllegalArgumentException.class)
  public void BirdsOfPreyConstructorDisallowsIntInStringList () { bop.set({sharp beak, hooked beak, visible nostrils },{ 1,2,3 } , Extinct.no, 2,{ other birds, small mammals, fish, aquatic invertebrates }); }

@Test for List<String>Eat (expected = IllegalArgumentException.class)
  public void BirdsOfPreyConstructorDisallowsIntInStringList () { bop.set({sharp beak, hooked beak, visible nostrils}, { hawks, eagles, osprey } , Extinct.no, 2, { 5,6,7 }); }

@Test for Enumerated type Extinct(yes,no,maybe) (expected = IllegalArgumentException.class)
  public void BirdsOfPreyConstructorDisallowsIntInStringList () { bop.set({sharp beak, hooked beak, visible nostrils },{ 1,2,3 } , Extinct.somewhat, 2,{ other birds, small mammals, fish, aquatic invertebrates }); }

@Test for Parrot

assertEquals("Parrots: Characteristics = short beak, curved beak, intelligent, ability to mimic sounds Type = rose ring parakeet, gray parrot, sulfur crested cockatoo Extinct = no Number of Wings = 2 Eat = berries, seeds, fruit Number of Words = 100 Favorite saying = hello, how do you do, hie ", par.toString())

@Test for Water Fowl

assertEquals("Water Fowl: Characteristics = lives near water sources Water sources = freshwater, saltwater Type =ducks, swans, geese Extinct = no Number of Wings = 2 Eat = seeds, eggs, fish", wf.toString())