Notes :

Birds are defined as warm-blooded, bipedal, vertebrate animals who have two wings instead of arms. They are typically covered in feathers and have a beak instead of a mouth.

- Birds of prey all have sharp, hooked beaks with visible nostrils. They include hawks, eagles, and osprey.
- Flightless birds live on the ground and have no (or undeveloped) wings. They include the emus, kiwis, and moas. Some (but not all) of these birds are extinct.
- Owls are distinguished by the facial disks that frame the eyes and bill.
- Parrots have a short, curved beak and are known for their intelligence and ability to mimic sounds. Many pet parrots can learn a vocabulary of up to 100 words and often adopt a single "favorite" saying. They include the rose-ring parakeet, gray parrot, and sulfur-crested cockatoo.
- Pigeons (or doves) are known for feeding their young "bird milk" very similar to the milk of mammals. Found all over the world, there are several varieties that are extinct.
- Shorebirds include the great auk, horned puffin, and African Jacana. They live near water sources including wetlands, freshwater and saltwater shorelands, even the ocean.
- Waterfowl are another classification that live near water sources (fresh or salt) and include ducks, swans, and geese.

For each classification of bird, your solution should be able to track each of the following:

- **The type of bird** (e.g., duck, horned puffin, etc), their **defining characteristic**, whether they are **extinct**, as well as **the number of wings** they have.
- A description of what 2-4 items they prefer to eat from the following list: berries, seeds, fruit, insects, other birds, eggs, small mammals, fish, buds, larvae, aquatic invertebrates, nuts, and vegetation.
- For **birds that live near water**, **the name of the body of water that they live by**.
- In the case of **parrots**, the **number of words** in their vocabulary as well as their **single "favorite" saying**.

Bird <<abstract>>
-id int
-birdType enum
-definingChar string
-isExtinct boolean
-wingsNum int
-preferredFood enummap<food, integer>
+Bird(BirdType, String, boolean, int, EnumMap)
+getType()
+getDefiningChar()
+isExtinct()
+getWingsNum()
+getPrefFood()
Food <<enumeration>>
Berries
Seeds
Fruit
Insects
Other Birds
Eggs
Small Mammals
Fish
Buds
Larvae
Aquatic Invertebrates
Nuts
Vegetation
BirdType <<enumeration>>
Hawks
Eagles
Osprey
Enus
Kiwis
Moas
Pigeons
Great_auk
Horned_puffin
African_jacana
Ducks
Swans
Geese
Owl
Rose_ring_parakeet
Gray_parrot
Sulfur_crested_cockatoo
Waterbody <<enumeration>>
Wetland
Freshwater_shoreland
Saltwater_shoreland
Ocean
BirdOfPray
+BirdOfPray(BirdType, String, boolean, int, EnumMap)
FlightlessBird
+FlightlessBird(BirdType, String, boolean, int, EnumMap)
Owl
+Owl(BirdType, String, boolean, int, EnumMap)
Parrot
-numberOfWords int
-favoriteSaying string
+Parrot(BirdType, String, boolean, int, EnumMap, String, int)
+getNumOfWords()
+getFavSaying()
Pigeon
+Pigeon(BirdType, String, boolean, int, EnumMap)
WaterBird <<abstract>>
-Waterbody enum
+WaterBird(BirdType, String, boolean, int, EnumMap, Waterbody)
+getLiveBy()
Shorebird
+ShoreBird(BirdType, String, boolean, int, EnumMap, Waterbody)
Waterfowl
+Waterfowl(BirdType, String, boolean, int, EnumMap, Waterbody)

	Description	Method name	Test case	Expected result	Actual result
BirdOfPray	Testing BirdOfPray with normal input	testConstructor()	birdOfPray = new BirdOfPray(BirdType.HAWKS, "sharp, hooked beaks with visible nostrils", false, 2, preferredFood);	construct BirdOfPray	construct BirdOfPray
	Testing the type of BirdOfPray	testType()	assertEquals(BirdType.HAWKS, birdOfPray.getType());	BirdType is HAWKS	BirdType is HAWKS
	Testing the defining character of BirdOfPray	testDefiningChar()	assertEquals("sharp, hooked beaks with visible nostrils", birdOfPray.	definingChar is "sharp, hooked beaks with visibl	definingChar is "sharp, hooked beaks with visible nostrils"
	Testing whether BirdOfPray is extinct	testIsExtinct()	assertFalse(String.valueOf(false), birdOfPray.isExtinct());	FALSE	FALSE
	Testing the number of wings BirdOfPray	testNumOfWings()	assertEquals(2, birdOfPray.getWingsNum());	2	2
	Testing the preferred food of BirdOfPray	testPreferredFood()	assertEquals(preferredFood, birdOfPray.getPreferredFood());	Preferred food include BERRIES and SEEDS	Preferred food include BERRIES and SEEDS
FlightlessBird	Testing FlightlessBird with normal input	testConstructor()	birdOfPray = new FlightlessBird(BirdType.HAWKS, "sharp, hooked beaks with visible nostrils", false, 2, preferredFood);	construct FlightlessBird	construct FlightlessBird
	Testing the type of FlightlessBird	testType()	assertEquals(BirdType.EMUS, FlightlessBird.getType());	BirdType is EMUS	BirdType is EMUS
	Testing the defining character of FlightlessBird	testDefiningChar()	assertEquals("sharp, hooked beaks with visible nostrils", FlightlessBird. getDefiningChar());	definingChar is "sharp, hooked beaks with visibl	definingChar is "sharp, hooked beaks with visible nostrils"
	Testing whether FlightlessBird is extinct	testIsExtinct()	assertFalse(String.valueOf(false), FlightlessBird.isExtinct());	FALSE	FALSE
	Testing the number of wings FlightlessBird	testNumOfWings()	assertEquals(2, FlightlessBird.getWingsNum());	2	2
	Testing the preferred food of FlightlessBird	testPreferredFood()	assertEquals(preferredFood, FlightlessBird.getPreferredFood());	Preferred food include BERRIES and SEEDS	Preferred food include BERRIES and SEEDS
Owl	Testing Owl with normal input	testConstructor()	birdOfPray = new Owl(BirdType.HAWKS, "sharp, hooked beaks with visible nostrils", false, 2, preferredFood);	construct Owl	construct Owl
	Testing the type of Owl	testType()	assertEquals(BirdType.EMUS, Owl.getType());	BirdType is OWL	BirdType is OWL
	Testing the defining character of Owl	testDefiningChar()	assertEquals("sharp, hooked beaks with visible nostrils", Owl.getDefiningChar());	definingChar is "facial disks that frame the eyes	definingChar is "facial disks that frame the eyes and bill"
	Testing whether Owl is extinct	testIsExtinct()	assertFalse(String.valueOf(false), Owl.isExtinct());	FALSE	FALSE
	Testing the number of wings Owl	testNumOfWings()	assertEquals(2, Owl.getWingsNum());	2	2
	Testing the preferred food of Owl	testPreferredFood()	assertEquals(preferredFood, Owl.getPreferredFood());	Preferred food include BERRIES and SEEDS	Preferred food include BERRIES and SEEDS
Parrot	Testing Parrot with normal input	testConstructor()	birdOfPray = new Parrot(BirdType.HAWKS, "sharp, hooked beaks with visible nostrils", false, 2, preferredFood);	construct Parrot	construct Parrot
	Testing the type of Parrot	testType()	assertEquals(BirdType.EMUS, Parrot.getType());	BirdType is PARROT	BirdType is PARROT
	Testing the defining character of Parrot	testDefiningChar()	assertEquals("sharp, hooked beaks with visible nostrils", Parrot.getDefiningChar());	definingChar is "sharp, hooked beaks with visibl	definingChar is "sharp, hooked beaks with visible nostrils"
	Testing whether Parrot is extinct	testIsExtinct()	assertFalse(String.valueOf(false), Parrot.isExtinct());	FALSE	FALSE
	Testing the number of wings Parrot	testNumOfWings()	assertEquals(2, Parrot.getWingsNum());	2	2
	Testing the preferred food of Parrot	testPreferredFood()	assertEquals(preferredFood, FlightlessBird.getPreferredFood());	Preferred food include BERRIES and SEEDS	Preferred food include BERRIES and SEEDS
	Testing the number of words	testNumOfWord()	assertEquals(50, parrot.getNumOfWords());	50	50
	Testing the favorit saying	testFavSaying()	assertEquals("Lovin’ it", parrot.getFavSaying());	Lovin’ it	Lovin’ it
Pigeon	Testing Pigeon with normal input	testConstructor()	birdOfPray = new Pigeon(BirdType.HAWKS, "sharp, hooked beaks with visible nostrils", false, 2, preferredFood);	construct Pigeon	construct Pigeon
	Testing the type of Pigeon	testType()	assertEquals(BirdType.EMUS, Pigeon.getType());	BirdType is Pigeon	BirdType is Pigeon
	Testing the defining character of Pigeon	testDefiningChar()	assertEquals("sharp, hooked beaks with visible nostrils", Pigeon.getDefiningChar());	definingChar is "feeding their young "bird milk\'	definingChar is "feeding their young "bird milk\"
	Testing whether Pigeon is extinct	testIsExtinct()	assertFalse(String.valueOf(false), Pigeon.isExtinct());	FALSE	FALSE
	Testing the number of wings Pigeon	testNumOfWings()	assertEquals(2, Pigeon.getWingsNum());	2	2
	Testing the preferred food of Pigeon	testPreferredFood()	assertEquals(preferredFood, Pigeon.getPreferredFood());	Preferred food include BERRIES and SEEDS	Preferred food include BERRIES and SEEDS
Shorebird	Testing Shorebird with normal input	testConstructor()	birdOfPray = new FlightlessBird(BirdType.HAWKS, "sharp, hooked beaks with visible nostrils", false, 2, preferredFood);	construct Shorebird	construct Shorebird
	Testing the type of Shorebird	testType()	assertEquals(BirdType.EMUS, Shorebird.getType());	BirdType is GREAT_AUK	BirdType is GREAT_AUK
	Testing the defining character of Shorebird	testDefiningChar()	assertEquals("sharp, hooked beaks with visible nostrils", Shorebird. getDefiningChar());	definingChar is "live near water sources"	definingChar is "live near water sources"
	Testing whether Shorebird is extinct	testIsExtinct()	assertFalse(String.valueOf(false), Shorebird.isExtinct());	FALSE	FALSE
	Testing the number of wings Shorebird	testNumOfWings()	assertEquals(2, Shorebird.getWingsNum());	2	2
	Testing the preferred food of Shorebird	testPreferredFood()	assertEquals(preferredFood, Shorebird.getPreferredFood());	Preferred food include BERRIES and SEEDS	Preferred food include BERRIES and SEEDS
Waterfowl	Testing Waterfowl with normal input	testConstructor()	birdOfPray = new Waterfowl(BirdType.HAWKS, "sharp, hooked beaks with visible nostrils", false, 2, preferredFood);	construct Waterfowl	construct Waterfowl
	Testing the type of Waterfowl	testType()	assertEquals(BirdType.EMUS, Waterfowl.getType());	BirdType is DUCK	BirdType is DUCK
	Testing the defining character of Waterfowl	testDefiningChar()	assertEquals("sharp, hooked beaks with visible nostrils", Waterfowl. getDefiningChar());	definingChar is "live near water sources"	definingChar is "live near water sources"
	Testing whether Waterfowl is extinct	testIsExtinct()	assertFalse(String.valueOf(false), Waterfowl.isExtinct());	FALSE	FALSE
	Testing the number of wings Waterfowl	testNumOfWings()	assertEquals(2, Waterfowl.getWingsNum());	2	2
	Testing the preferred food of Waterfowl	testPreferredFood()	assertEquals(preferredFood, Waterfowl.getPreferredFood());	Preferred food include BERRIES and SEEDS	Preferred food include BERRIES and SEEDS