Section A

1a. Add sing() method inside Bird Class. You can unit test Solution class and ensure when sing() method is called it prints "I am Singing". Alternately you can write JUNIT testcases and use assertion for testing sample code SolutionTest

1b. The first approach is not scalable/maintainable if we have a new animal that cannot walk it wont fit into inheritance hierarchy. To solve this create Animal abstract calss remove its behaviour walk(),sing(),fly() and add it to an interface so that a concrete class ex. Interface decouples the behaviour.

2. .Added new classes Duck and Chicken which Extend class Bird(In later part of the solution Bird extends IFly and ISwim) Interface. Duck and Chicken override the sing() method of Bird class to implement their own sounds.

3 Created a Rooster class that extends Chicken. Rooster is a male chicken . Override the sing() method in. chicken to make sound “Cock-a-doodle-doo”.

4. Created Parrot Class which extends Bird, Whose sing() method is same as Bird . Created separate classes ParrotLivingWithCat , ParrotLivingWithDog and ParrotLivingWithRooster

for parrot living with cat , dog and rooster respectively , which extends the Parrot Class and overrides the sing() method to make the respective sounds.

So that every time we want to model a parrot making some different sound or behaviour we can create a new class and override the sing() method instead of making changes to the Parrot class.

Section B

Fishes extend ISwim Interface and overrides the swim method.

Created a new interface SwimmingAnimals which has methods physicalAppreance() and behaviour() .Created two classes Shark and Clownfish which extends Fishes and implements SwimmingAnimals .

The ClownFish overrides the physicalAppearance() -orange and behaviour()-making jokes .

The Shark overrides the physicalAppearance() -large and grey and behaviour()- eat other fishes.

Created Dolphin class which implements the ISwim interface and overrides its swim() method.

Section D

1. Create an Butterfly class which implements Ifly interface and overrides in canfly() method.
2. Create an Catterpillar class which Implements WalkIF and overrides its walk() method, we can transfor() method that transforms the catterpillar to butterfly.

Section E You can implement factory pattern AnimalFactory which is object creational pattern then add the objects to the list and then count the naimals that can fly, sing, swim,

BONUS.

1. We can consider this as localisation, we can have a map of languages and its translation and depending on native language we can print the corresponding sound for EX Rooster sound in Hindi is Kukudooku. Test Class Solution.
2. Couldn’t implement due to lack of spring framework in home computer.