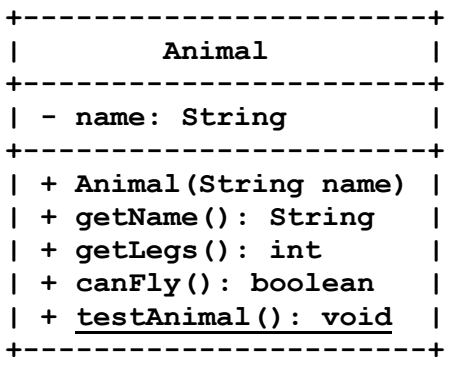
Lab 7 Requirements

Create a new Eclipse workspace named "**Lab7_1234567890**" on the desktop of your computer (replace **1234567890** with your student ID number). For each question below, create a new project in that workspace. Call each project by its question number: "**Question1**", "**Question2**", etc. If you do not remember how to create a workspace or projects, read the "*Introduction to Eclipse*" document which is on iSpace. Answer all the questions below. At the end of the lab, create a ZIP archive of the whole workspace folder. The resulting ZIP file must be called "**Lab7_1234567890.zip**" (replace **1234567890** with your student ID number). Upload the ZIP file on iSpace.

Question 1

Create a class **Animal** with the following UML specification:



where the **name** instance variable stores a name for the animal, the **getLegs** method returns as result the animal's number of legs, and the **canFly** method returns as result a boolean indicating whether the animal can fly or not. The **testAnimal** method is static.

Should the **getLegs** method be abstract? Why or why not?

Should the **canFly** method be abstract? Why or why not?

Should the **Animal** class be abstract? Why or why not?

What kinds of tests can you write inside the **testAnimal** method?

Add the following code to your program to test the **Animal** class:

```
public class Start {
	public static void main(String[] args) {
		Animal.testAnimal();
	}
}
```

Question 2

Add a class **Dog** to your program. Dogs are animals. The constructor for the **Dog** class takes the name of the dog as argument. Dogs have four legs and cannot fly.

Do not forget to change the **main** method of the **Start** class to run the unit tests of the new **Dog** class.

Question 3

Add a class **Bird** to your program. Birds are animals. The **Bird** class must have a private instance variable called **numOfEggs** which is an integer indicating how many eggs the bird has. The constructor for **Bird** takes as arguments a name and a number of eggs. The **Bird** class has a public method called **getNumOfEggs** that takes zero arguments and returns as result the bird's number of eggs. All birds have two legs. Some birds can fly (for example magpies) and some birds cannot fly (for example ostriches).

Do not forget to change the **main** method of the **Start** class to run the unit tests of the new **Bird** class.

Question 4

Add a class **Magpie** to your program. Magpies are birds. The constructor for **Magpie** takes as argument only a name. Magpies always have 6 eggs. Magpies can fly.

Do not forget to change the **main** method of the **Start** class to run the unit tests of the new **Magpie** class.

Question 5

Add a class **Ostrich** to your program. Ostriches are birds. The constructor for **Ostrich** takes as argument only a name. Ostriches always have 10 eggs. Ostriches cannot fly.

Do not forget to change the **main** method of the **Start** class to run the unit tests of the new **Ostrich** class.

Question 6

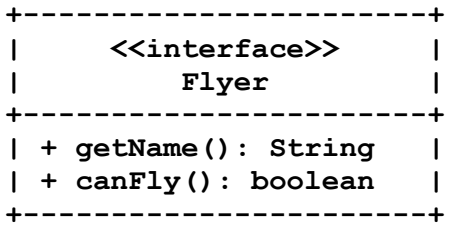
Add a class **Pegasus** to your program. Pegasi are birds (a pegasus has the wings of a bird so for this lab we will assume that pegasi are birds). The constructor for **Pegasus** takes as argument only a name. Pegasi have four legs (not two legs like other birds). Pegasi can fly.

Pegasi do not lay eggs so the **getNumOfEggs** method of the **Pegasus** class should just print a message "**Pegasi do not lay eggs!**" and return zero as a result.

Do not forget to change the **main** method of the **Start** class to run the unit tests of the new **Pegasus** class.

Question 7

Add an interface **Flyer** with the following UML specification:



The **Bird** class implements the **Flyer** interface.

Change the **main** method of the **Start** class to tests magpies, ostriches, and pegasi when viewed through the **Flyer** interface.

Can you test objects from the **`Animal`**, **`Dog`**, or **`Bird`** classes through the **`Flyer`** interface? Why or why not?

Question 8

Add a class **`Airplane`** that implements the **`Flyer`** interface. The constructor for **`Airplane`** takes as argument only a name. An airplane is not an animal.

Do not forget to change the **`main`** method of the **`Start`** class to run the unit tests of the new **`Airplane`** class. Also make sure to test it when viewed through the **`Flyer`** interface.

Question 9

Add a method **`isDangerous`** to the **`Flyer`** interface. This method returns a boolean as result, indicating whether the corresponding object is dangerous or not. Only ostriches are dangerous.

Which classes need to be changed? Which classes do not need to be changed?

Do not forget to add new tests for the **`isDangerous`** method.