Lab 7 Requirements

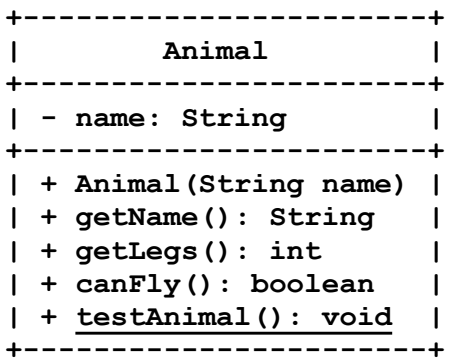
Create a new Eclipse workspace named "**OOP**" on the desktop of your computer. Create a Java Project named "Lab7". Download the sample code **lab7-src-for-student.zip** from iSpace and unzip it. Next to import the sample code(right click the src -> Import… -> File System -> locate to the unzip folder -> import all the questions to your project) . More details, refer to file *How to Use Eclipse and submit to Autolab for student.pdf* on iSpace. Note: you **must** use the sample code, so AutoLab can work correctly.

Answer all questions below. Note: a question may need answers of the previous question(s), just copy paste the previous code to your new package. At the end of the lab, create a ZIP archive of all the src. Refer to previous labs requirements if you forgot how to compress files.

Extract your zip file to confirm that your zip file contains folders. Rename your src.zip to "**Lab7_1234567890.zip**" (replace **1234567890** with your student ID number). Upload the ZIP file onto AutoLab.

question 1 (Reviewed)

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 (Reviewed)

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 (Reviewed)

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 (Mandatory)

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 (Mandatory)

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 (Mandatory)

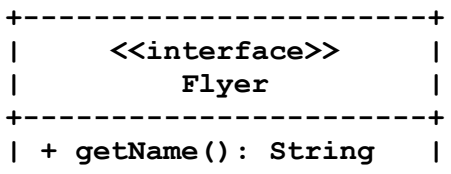
Add a class **Pegasus** to your program. Pegasi are birds (a pegasus has the wings of a bird so for this lab we will assume that pegasi are birds). The constructor for **Pegasus** takes as argument only a name. Pegasi have four legs (not two legs like other birds). Pegasi can fly.

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 (Mandatory)

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



```
      | + canFly(): boolean   |
      +-----------------------+
```

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 (Mandatory)

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 (Optional)

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.

Sample Test Cases:

---

question 1

```
public static void testAnimal() {

}
```

question 2

```
public static void testAnimal() {

}

public static void testDog() {
    Dog d = new Dog("Nice Doggy");
    // The getName method is inherited from Animal.
    // The getLegs and canFly methods come from Dog itself.
    System.out.println(d.getName() == "Nice Doggy");
    System.out.println(d.getLegs() == 4);
    System.out.println(d.canFly() == false);
}
```

question 3

```
public static void testAnimal() {

}

public static void testDog() {
    Dog d = new Dog("Nice Doggy");
    // The getName method is inherited from Animal.
    // The getLegs and canFly methods come from Dog itself.
    System.out.println(d.getName() == "Nice Doggy");
    System.out.println(d.getLegs() == 4);
    System.out.println(d.canFly() == false);
}
public static void testBird() {

}
```

question 4

```
public static void testAnimal() {

}

public static void testDog() {
    Dog d = new Dog("Nice Doggy");
    // The getName method is inherited from Animal.
    // The getLegs and canFly methods come from Dog itself.
    System.out.println(d.getName() == "Nice Doggy");
    System.out.println(d.getLegs() == 4);
    System.out.println(d.canFly() == false);
}
public static void testBird() {

}

public static void testMagpie() {
    Magpie m = new Magpie("Maggie");
    System.out.println(m.getName() == "Maggie");
    System.out.println(m.getLegs() == 2);
    System.out.println(m.getNumOfEggs() == 6);
    System.out.println(m.canFly() == true);
}
```

question 5

```
public static void testAnimal() {

}

public static void testDog() {
    Dog d = new Dog("Nice Doggy");
    // The getName method is inherited from Animal.
    // The getLegs and canFly methods come from Dog itself.
    System.out.println(d.getName() == "Nice Doggy");
    System.out.println(d.getLegs() == 4);
    System.out.println(d.canFly() == false);
}
public static void testBird() {

}
```

```
public static void testMagpie() {
    Magpie m = new Magpie("Maggie");
    System.out.println(m.getName() == "Maggie");
    System.out.println(m.getLegs() == 2);
    System.out.println(m.getNumOfEggs() == 6);
    System.out.println(m.canFly() == true);
}

public static void testOstrich() {
    Ostrich o = new Ostrich("Ossie");
    System.out.println(o.getName() == "Ossie");
    System.out.println(o.getLegs() == 2);
    System.out.println(o.getNumOfEggs() == 10);
    System.out.println(o.canFly() == false);
}
```

question 6

```
public static void testAnimal() {

}

public static void testDog() {
    Dog d = new Dog("Nice Doggy");
    // The getName method is inherited from Animal.
    // The getLegs and canFly methods come from Dog itself.
    System.out.println(d.getName() == "Nice Doggy");
    System.out.println(d.getLegs() == 4);
    System.out.println(d.canFly() == false);
}
public static void testBird() {

}

public static void testMagpie() {
    Magpie m = new Magpie("Maggie");
    System.out.println(m.getName() == "Maggie");
    System.out.println(m.getLegs() == 2);
    System.out.println(m.getNumOfEggs() == 6);
    System.out.println(m.canFly() == true);
}

public static void testOstrich() {
    Ostrich o = new Ostrich("Ossie");
    System.out.println(o.getName() == "Ossie");
    System.out.println(o.getLegs() == 2);
    System.out.println(o.getNumOfEggs() == 10);
    System.out.println(o.canFly() == false);
}

public static void testPegasus() {
    Pegasus p = new Pegasus("Peggie");
    System.out.println(p.getName() == "Peggie");
    System.out.println(p.getLegs() == 4);
    System.out.println(p.getNumOfEggs() == 0); // Message printed here.
    System.out.println(p.canFly() == true);
}
```

question 7

```
public static void testAnimal() {

}

public static void testDog() {
    Dog d = new Dog("Nice Doggy");
    // The getName method is inherited from Animal.
    // The getLegs and canFly methods come from Dog itself.
    System.out.println(d.getName() == "Nice Doggy");
    System.out.println(d.getLegs() == 4);
    System.out.println(d.canFly() == false);
}
public static void testBird() {

}

public static void testMagpie() {
    Magpie m = new Magpie("Maggie");
    System.out.println(m.getName() == "Maggie");
    System.out.println(m.getLegs() == 2);
    System.out.println(m.getNumOfEggs() == 6);
    System.out.println(m.canFly() == true);
}

public static void testOstrich() {
    Ostrich o = new Ostrich("Ossie");
    System.out.println(o.getName() == "Ossie");
    System.out.println(o.getLegs() == 2);
    System.out.println(o.getNumOfEggs() == 10);
    System.out.println(o.canFly() == false);
}

public static void testPegasus() {
    Pegasus p = new Pegasus("Peggie");
    System.out.println(p.getName() == "Peggie");
    System.out.println(p.getLegs() == 4);
    System.out.println(p.getNumOfEggs() == 0); // Message printed here.
    System.out.println(p.canFly() == true);
}
```

question 8

```
public static void testAnimal() {

}

public static void testDog() {
    Dog d = new Dog("Nice Doggy");
    // The getName method is inherited from Animal.
    // The getLegs and canFly methods come from Dog itself.
    System.out.println(d.getName() == "Nice Doggy");
    System.out.println(d.getLegs() == 4);
    System.out.println(d.canFly() == false);
}
public static void testBird() {

}

public static void testMagpie() {
    Magpie m = new Magpie("Maggie");
```

```
    System.out.println(m.getName() == "Maggie");
    System.out.println(m.getLegs() == 2);
    System.out.println(m.getNumOfEggs() == 6);
    System.out.println(m.canFly() == true);
}

public static void testOstrich() {
    Ostrich o = new Ostrich("Ossie");
    System.out.println(o.getName() == "Ossie");
    System.out.println(o.getLegs() == 2);
    System.out.println(o.getNumOfEggs() == 10);
    System.out.println(o.canFly() == false);
}

public static void testPegasus() {
    Pegasus p = new Pegasus("Peggie");
    System.out.println(p.getName() == "Peggie");
    System.out.println(p.getLegs() == 4);
    System.out.println(p.getNumOfEggs() == 0); // Message printed here.
    System.out.println(p.canFly() == true);
}

public static void testAirplane() {
    Airplane a = new Airplane("Aircar");
    System.out.println(a.getName() == "Aircar");
    System.out.println(a.canFly() == true);
}
```

question 9

```
public static void testAnimal() {

}

public static void testDog() {
    Dog d = new Dog("Nice Doggy");
    // The getName method is inherited from Animal.
    // The getLegs and canFly methods come from Dog itself.
    System.out.println(d.getName() == "Nice Doggy");
    System.out.println(d.getLegs() == 4);
    System.out.println(d.canFly() == false);
}
public static void testBird() {

}

public static void testMagpie() {
    Magpie m = new Magpie("Maggie");
    System.out.println(m.getName() == "Maggie");
    System.out.println(m.getLegs() == 2);
    System.out.println(m.getNumOfEggs() == 6);
    System.out.println(m.canFly() == true);
    System.out.println(m.isDangerous() == false);
}

public static void testOstrich() {
    Ostrich o = new Ostrich("Ossie");
    System.out.println(o.getName() == "Ossie");
    System.out.println(o.getLegs() == 2);
    System.out.println(o.getNumOfEggs() == 10);
```

```
    System.out.println(o.canFly() == false);
    System.out.println(o.isDangerous() == true);
}

public static void testPegasus() {
    Pegasus p = new Pegasus("Peggie");
    System.out.println(p.getName() == "Peggie");
    System.out.println(p.getLegs() == 4);
    System.out.println(p.getNumOfEggs() == 0); // Message printed here.
    System.out.println(p.canFly() == true);
    System.out.println(p.isDangerous() == false);

}

public static void testAirplane() {
    Airplane a = new Airplane("Aircar");
    System.out.println(a.getName() == "Aircar");
    System.out.println(a.canFly() == true);
    System.out.println(a.isDangerous() == false);

}
```