ABSTRACT CLASSES AND INTERFACES

# Abstract classes

## Objective

The primary objective of this lab is to provide you with the skills necessary to be able to:

* Create and work with abstract classes
* Define and implement interfaces

## Overview

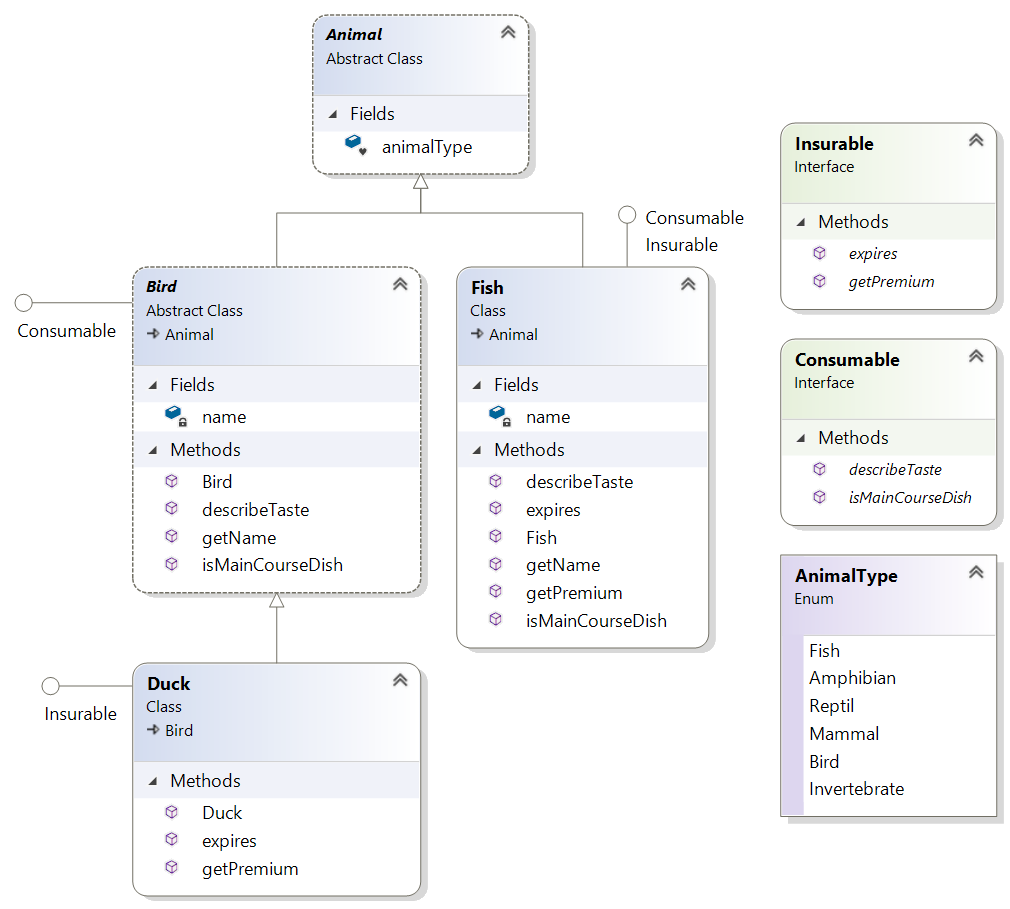
You are going to have Penguins, Ducks and Fish in a collection.

Penguins and Ducks will be derived from the Bird class. We have decided that "Bird" is too abstract for what we want.

All birds are consumable; and Ducks are insurable (Penguins aren't).

Fish (which is not derived from Bird) are also insurable but are consumable.

We'll iterate round the collection and call operations on all consumable things, then on all insurable things regardless of whether they are inherited from Bird or not.  
The class diagram after completing the lab can be seen below.



### Step by step

1. Create a new package called lab19.
2. Create a new class called Program with a main() method in the package
3. Add the following classes to the package
4. Animal, Bird, Fish and Duck
5. Add the following interfaces to the package
6. Insurable, Consumable
7. Create an enum to define the 6 animal type like:

**enum** AnimalType {

***Fish***,

***Amphibian***,

***Reptil***,

***Mammal***,

***Bird***,

***Invertebrate***

}

1. Add code to the Animal class
2. Make this class **abstract**
3. Add a field to define the type, like:  
   **AnimalType** **animalType**;
4. Add code to the Bird class
5. Make this class **abstract**we want to make different kinds of birds, not a generic bird!
6. **extends** the Animal class
7. field called **name** with **getName()** method
8. Create a constructor to set the name (String name as parameter)
9. Set the type of the animal to Bird.
10. Write code for the Duck class
11. **extends** Bird  
    By default a **Duck** will also be an **Animal** Bird extends Animal)
12. You'll need to provide a constructor because the super class (Bird) has constructor with a String parameter to set the name.
13. Write code for the Penguin class
14. **extends** Bird
15. Provide a constructor similar to the Duck class
16. Write code for the interfaces (see below)

**public** **interface** Insurable {

String getPremium();

String expires();

}  
  
**public** **interface** Consumable {

String describeTaste();

String isMainCourseDish();

}

**Note** You would normally expect expires to return a Date object. It returns a String just to make life easier for testing it in this exercise.

1. Check that everything compiles.
2. Implement the interface. Please view the class diagram on page one.
3. For the describeTaste() and isMainCourseDish(),   
   you could write something like:  
     
    **public** String describeTaste() {

**return** getName() + ": " + "Delicate";

}

**public** String isMainCourseDish() {

**return** getName() + ": " + **true**;

}

1. Open class Program.

Now you write some code that invokes the just defined interface methods.

1. Create an array of **Animal** called **animals**Place different Animal instances in the array
2. Write an enhanced ‘for’ loop that iterates over the **animals** array.  
   This is an example of Polymorphism. Every Fish, Penguin, Duck can be referred to as Animal and can therefore be placed in an array of type Animal.
3. Inside the loop test ifthe Animal object is Consumable and if it is, cast it as Consumable in order to call its describeTaste() and isMainCourseDish() methods.
4. Run your code to make sure it works.
5. Return to the main() method.
6. Write another enhanced ‘for’ loop that iterates over the **animals** array.
7. Inside the loop, test ifthe Animal object is **Insurable** and if it is, cast it as Insurable in order to call its **getPremium**() and **expires**() methods.

\*\* End \*\*