EXERCISE 6, ABSTRACT CLASSES AND INTERFACES

Exercise 6, 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

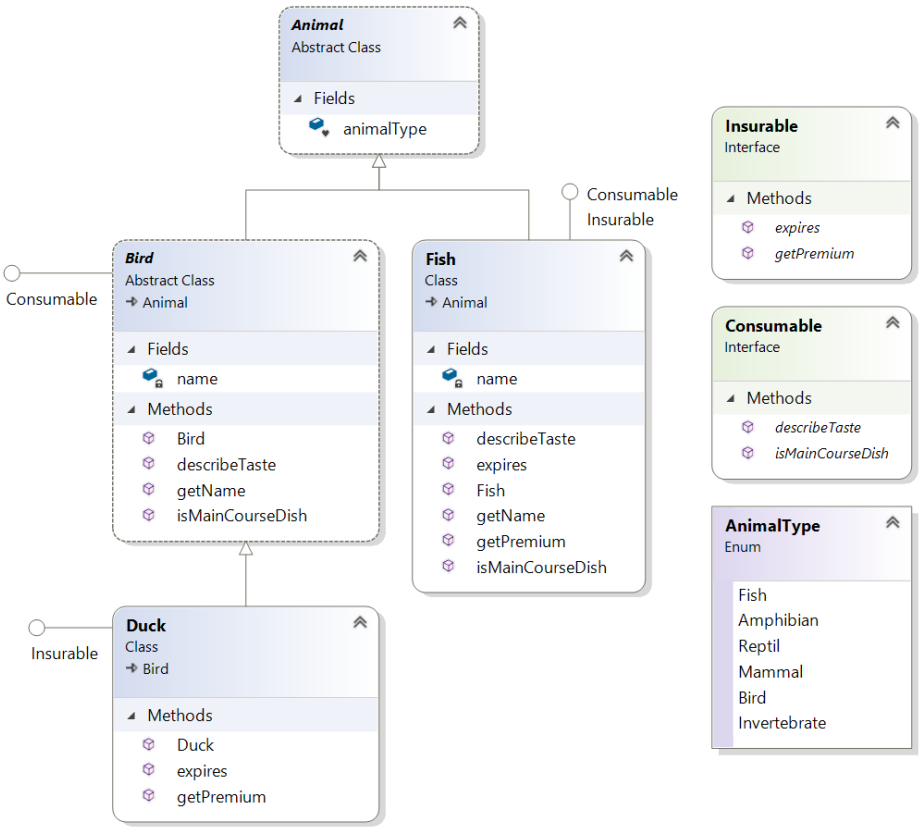
You are going to have Penguins, Ducks and Fish in a collection.

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 lab06.
2. Create a new class called Program with a main() method in the package
3. Add the following classes to the package
   a. Animal, Bird, Fish and Duck
4. Add the following interfaces to the package
   a. Insurable, Consumable
5. Create an enum to define the 6 animal type like:

```
enum AnimalType {
 Fish,
 Amphibian,
 Reptil,
 Mammal,
 Bird,
 Invertebrate
}
```

6. Add code to the Animal class
   a. Make this class **abstract**
   b. Add a field to define the type, like:
      AnimalType animalType;
7. Add code to the Bird class
   a. Make this class **abstract**
      we want to make different kinds of birds, not a generic bird!
   b. **extends** the Animal class
   c. field called name with getName() method
   d. Create a constructor to set the name (String name as parameter)
   e. Set the type of the animal to Bird.
8. Write code for the Duck class
   a. **extends** Bird
      By default a Duck will also be an Animal Bird extends Animal)
   b. You'll need to provide a constructor because the super class (Bird) has constructor with a String parameter to set the name.
9. Write code for the Penguin class
   a. **extends** Bird
   b. Provide a constructor similar to the Duck class

10. 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.

11. Check that everything compiles.
12. Implement the interface. Please view the class diagram on page one.
    a. For the describeTaste() and isMainCourseDish(), you could write something like:

```
public String describeTaste() {
      return getName() + ": " + "Delicate";
}

public String isMainCourseDish() {
      return getName() + ": " + true;
}
```

13. Open class **Program**.

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

    a. Create an array of Animal called animals
    Place different Animal instances in the array

    b. 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.

    c. Inside the loop test **if** the Animal object is **Consumable** and if it is, cast it as Consumable in order to call its describeTaste() and isMainCourseDish() methods.

14. Run your code to make sure it works.
15. Return to the main() method.
    a. Write another enhanced '**for**' loop that iterates over the animals array.

    b. Inside the loop, test **if** the Animal object is **Insuable** and if it is, cast it as Insurable in order to call its getPremium() and expires() methods.

** End **