Advanced Programming

Lab Sheet 9 – Inheritance & Interfaces

Introduction

This lab focusses on the use of inheritance in your code, and the implementation of interfaces. Although you covered inheritance briefly in your first year, effective use of these concepts is crucial to a great number of common tasks in Java. The lab involves the creation of classes representing various pets sold at a shop. The example scenario is light-hearted, but provides a good opportunity to work through the various concepts.

Task 1: Creating the Pet Class

The base class, from which all your more specific types of pet will derive, is going to be named Pet. Create the class such that pets have a name, an age and a cost property. You will also need to create a getter and setter method for each property. You'll need one additional method in the Pet class, named makeNoise(). This method does not need to accept any parameters, and doesn't need to return anything. Calling a Pet's makeNoise() method should cause a message to be printed to the console: "NAME doesn't make any noise", where NAME is replaced with the Pet's name. I would also recommend declaring this class abstract, which means that no instances of the Pet class can be constructed. The only way to create a Pet is to instantiate an object that extends the class.

Task 2: Creating Some Pets

For now, we're going to create three classes that extend Pet: a Cat class, a Dog class and a Goldfish class. You will need to override the makeNoise() method in the Cat and Dog classes such that the correct animal noise is written to the console. Goldfish should use the super keyword in their makeNoise() method to invoke the overridden method in the Pet class, as well as printing "They're a goldfish!" to the console.

At this stage, you should create a class named Controller or Main, with a main() method that instantiates a Cat, a Dog and a Goldfish and checks their makeNoise() methods are working properly.

Task 3: Creating an Interface, and Implementing It

We're going to create an interface that is used to represent Pets that can be stroked. A pet that can be stroked will have a stroke() method, accepting no parameters and returning nothing. Create a Strokeable interface that requires classes implementing it to have such a method.

Modify your Cat and Dog classes so that they implement the Strokeable interface. When stroked, Dogs should write "NAME enjoys being stroked." to the console, and the makeNoise() method should be called. Cats, on the other hand, should write "NAME wanders off and ignores you." without making a noise. Goldfish should not be stroked.

Modify your main method to include some tests for the new stroke() method for both Cats and Dogs, to demonstrate that your newly-added code is working well.

Task 4: Creating a Pet Shop

Add a new class to your eclipse project named PetShop, and write a default (no parameter) constructor for it that creates an array of 10 Pets to represent the shop's stock. You can choose how many Cats, Dogs and Goldfish the shop will have in stock to suit yourself: it doesn't really matter. Once a Pet has been sold, its entry in the array will be replaced with a null.

Your Pet shop will need a number of methods to allow customers to actually buy pets. The buyCat(), buyDog() and buyGoldfish() methods should search through the PetShop's array of stock for the first suitable animal, replace its entry in the stock array with null, and return the pet to the customer. If the store is out of the requested type of animal, it should return null.

The PetShop class will also have a buyPetByCost() method, that accepts a double as a parameter representing the cash a customer can afford to spend. This method will return the most expensive animal from its stock that the customer can afford. If the customer cannot afford any of the animals in stock, the method can return null.

Add code to your main() method to create a pet shop, and demonstrate that each of the buying methods is working properly. Remember to check to see if they returned null before calling any methods on the returned Pets: if you forget you'll get a NullPointerException!

Extension Task: The Stroke-a-Tron 5000

The PetShop has invested in a marvellous new piece of technology: the Stroke-a-Tron 5000, capable of quickly and efficiently stroking many pets with astounding speed. Fig. 1, below, depicts a similar marvel of modern technology, being used to stroke cows on an industrial scale.

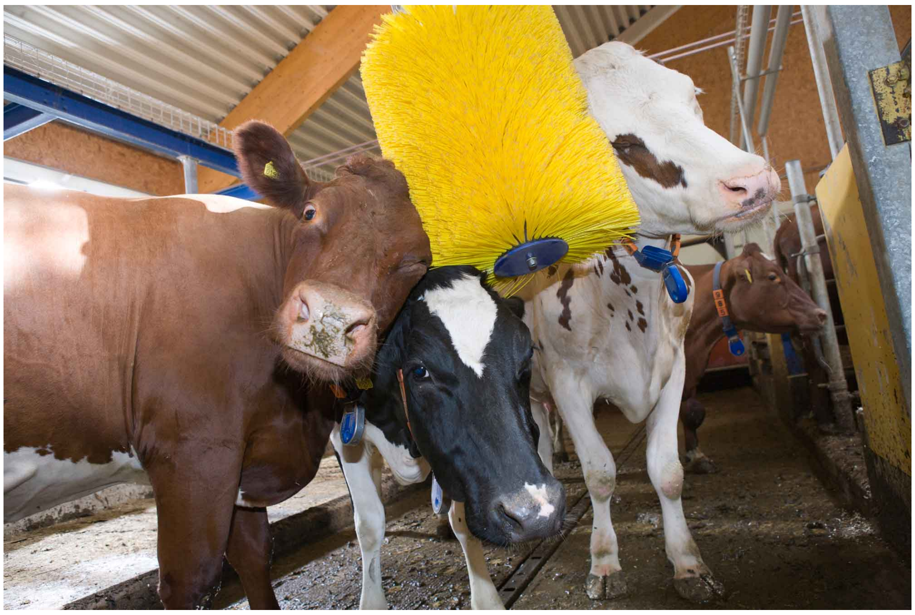
Figure 1: The Stroke-A-Tron 6000, designed for larger animals

Using the Stroke-a-Tron 5000 on a pet that is not Strokeable could be dangerous! Your PetShop will need a getAllStrokeablePets() method that returns an array of only those animals suitable for use in the magnificent manipulator. For this, you'll need to use the instanceof operator.

The Stroke-a-Tron 5000 class itself, named StrokeATron, will need a method called strokeAll(), which accepts an array of Strokeable objects. The method will loop through the supplied array and call stroke() on each of its members. You will also need to modify your main() method to demonstrate that the stroke-a-tron 5000 is working properly, and stroking all of the pets that it should.