USING OO PARADIGMS IN ZOO/ANIMAL

Niall Curtis

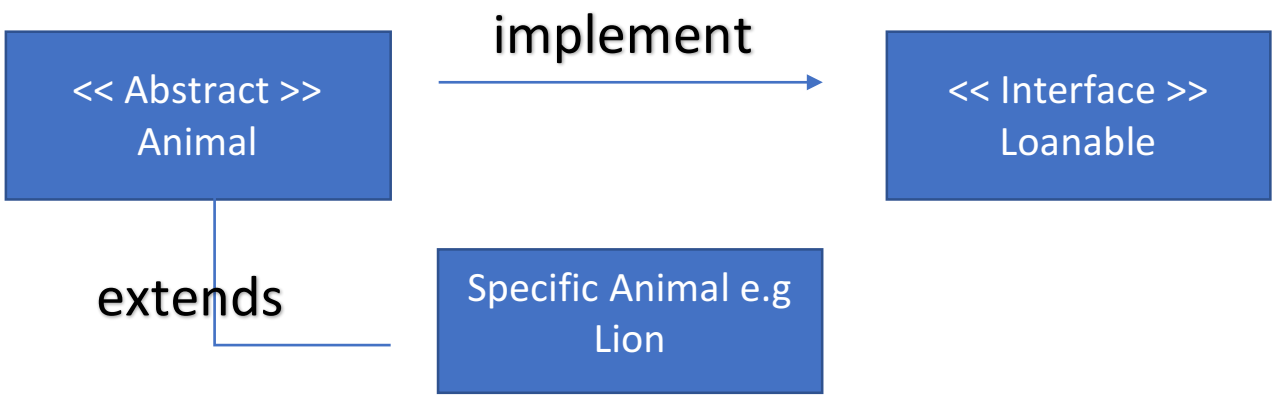
C1623580

The aim of this document is to provide an idea of the flexibility that can be provided by designing the coursework with a much greater emphasis on the object orientation side, as in its current state is just the Zoo class implementing methods from the Animal class. The two tent poles of this design are introducing polymorphism and hierarchy.

Polymorphism will allow me to separate the interface level, and the implementation level, enabling the programmer to evade the difficulties that may arise with a complex system and program at the interface level once all the implementation has been completed.

Hierarchy is another useful element of object oriented programming that permits derivation of classes by ordering them based on an 'is-a' relationship', to avoid duplication and redundancy of code.

Here is a basic structure of how I believe to best implement object orientation into this coursework:



The Animal class will be transformed into an abstract class. The Animal class now basically serves as a template to be further developed by the subclass, which theoretically would be any animal that wanted to be added, for example a lion. Even though practically it wouldn't make sense to do it this way as you would have to make a class for every animal, in a theoretical pure-OO world it makes perfect sense as it's the best way of software design. It allows very easy extension, as all animals share the common interface, especially in this very basic system. A Lion, for example, shares all same methods that any basic animal would have, so it makes sense that it extends the Animal class. The same would apply for a Dog, Cat, etc. This avoids repetition as many methods would be repeated for every specific animal.

The other thing we are doing is reducing complexity by hierarchically implementing a pure interface level class called Loanable. Even though animal is mostly abstract, it will include simple methods like in the current system for

the loaning and returning of animals from other zoos. We can reduce the current complexity of this model by introducing the Loanable class, that only has the definition of the loaning methods to create a ‘standard’ for loaning. As the interface is 100% abstract, all the actual method code is in Animal. The main aim for doing this is to make the system simpler, by breaking it down into its constituent parts which helps in a more complex implementation.

As a basic idea, the class declaration would look like this:

- public class Animal implements Loanable
- public class Lion extends Animal