Joshua Hoffmann

Advanced Java

Unit Project 3

When creating programs everything must be done in steps. First you must know what the program is about and how you want to go about it. Understanding all the variables needed is a key component to completing any project. During a projects development live there will always be obstacles. One major obstacle is when the customer wants to change or add ideas to the projects. To compensate for this, programmers use version control to help keep track of all the changes. With each version a project should improve from the previous one. Understanding why and how these changes happen will help any developer help understand the project better. Let’s see how the SimUDuck project changed with each version implemented.

Version 1 of the SimUDuck program starts with the basics of having a main class to create the ducks and run the program. It has a generic Duck class that simulates what a duck can do. This includes quacking, swimming and a display feature so users can see what is happening. Version 1 also creates classes which create different variations of the duck species. Each of which is given a display method so that they can be shown on the screen. Version 2 expands on what version 1 did. The programmer created two more classes, RubberDuck and DecoyDuck, both of which are subclasses of the original duck class. At this level the methods can only be called from the Duck class, this way each different type of duck does not have their own way of quacking, swimming, and flying. Each type of duck must have their own unique quack, right? How about how they fly, is that different to?

SimUDuck version 3 makes the program a little more universal. The programmer has made Interfaces for the quack and fly functions. With this version the programmer has made it, so all ducks are able to swim. The RubberDuck, RedheadDuck and MallardDuck can all quack, and only the ReadHeadDuck and the MallardDuck are able to fly. The DecoyDuck is a class representing a fake duck which is unable to swim, quack or fly. The RubberDuck is exactly what it sounds like, it is a duck that makes a quacking noise, but is unable to fly or swim. The ReadheadDuck and MallardDuck classes represent real life ducks that can swim, quack and fly, hence both classes implement the Quackable and Flyable interfaces. The interfaces were made so that the main Duck class can reference a single interface rather than reaching out to all the different duck classes separately.

Moving onto version 4, we must think about if we want to add different types of ducks. Will they be able fly? What type of sound do they make when they quack? We have three actions that a duck may be able to do. We have quack, fly, and swim. The programmer has decided that all ducks are able to float and leave the swim method within the Duck class, as the method will be the same no matter what duck you are talking about. On the other hand, depending on the type of duck the quack and fly methods will be different. To make sure that the Duck class can reference all the quack and fly methods we create two interfaces, one that will reference quack and on that will reference fly. Our goal is to make sure that we can reuse all our interfaces when it is time to make a new type of duck.

Lastly, we look at version 5. In our Duck class we now see that there are set methods for QuackBehavior and FlyBehavior. We want to be able to distinguish what each separate duck can do, not just what each type of duck can do. In this way we can make one mallard quack while another one is mute. In previous versions all mallards had to be set to one type of quack. The same can be said about flying. The goal is to be able to set unique characteristics for each individual duck that is created. With the newly created fly and quack behaviors the programmer needs methods for the ducks to execute their set behaviors. These are the performQuack and performFly methods. Creating these methods to universally execute the set behaviors that were previously made when the ducks were created.