**Class Assignment – Strategy Design Pattern**

|  |  |
| --- | --- |
| Group Member Names |  |
| |  |  | | --- | --- | | 1. | Sean Northcutt | | 2. | Andrew Russel | | |  |  | | --- | --- | | 3. | Johnnie Oldfield | | 4. | Carson Davis | |

You will read the handout and answer the 8 questions below. This is better done with your group. Alternate the person reading, discuss and type answers as you go along. Only one person in group needs to submit. You may work in a smaller group, or by yourself.

1. Describe some downsides of inheritance.

|  |
| --- |
| A subclass that was not supposed to inherit a certain attribute will inherit it from the Super class. |

1. Describe the shortcomings of the interface approach mentioned on page 6.

|  |
| --- |
| The interfaces of Flyable and Quackable are being used on multiple subclasses which could be avoided if he just went with making a QuackableDuck subclass that has two subclasses which would have a flyable subclass. Flyable would have the MallardDuck and RedheadedDuck and then RubberDuck would just be a subclass of QuackableDuck. |

1. Suppose that we need to be able to change behaviors at run-time. Describe how to do this (assume you are talking to another developer).

|  |
| --- |
| Program a class that points to an interface for a specific behavior which then makes a supertype that will work as the behavior class that you want. Like for QuackableDuck you could make an interface for QuackBehavior with subclasses that have Quack or Squeak or nothing can be implemented depending on how you want them. |

1. Describe the benefit we gain by adhering to the design principle on page 11.

|  |
| --- |
| Using this method helps make it to where you don’t have to lock in a specific method into a certain class or subclass and instead you will have separate classes that exist for that sole behavior rather than the super class. |

1. Explain two benefits of the design principle on page 23.

|  |
| --- |
| It allows you t oencapsulate a family of algorithms into their own set of classes, and it also lets you change behavior at runtime as long as the object we are composing implements the correct behavior interface. |

1. How is the design principle on page 9 exhibited in the Strategy pattern?

|  |
| --- |
| Divide out the work amongst behavior interfaces and make subclasses from those interfaces to make the code more flexible and easier to change. |

1. Discuss some advantages of *thinking at the pattern level.*

|  |
| --- |
| It makes the workflow and communication between workers go more smoothly and efficiently since they won’t have to think about everything someone was trying to tell them and rather just using a set style to do what they were needed to do. Like how police have codes that mean certain types of crimes have been committed to make the communication simpler. |

1. Discuss how the strategy pattern contributes to extensibility ([http:/en.wikipedia.org/wiki/Extensibility](http://en.wikipedia.org/wiki/Extensibility))?

|  |
| --- |
| Extensibility is about how easy a system is supposed to be to be expanded on and with the use of patterns everything should be uniform which should help with scalability and efficiency. |