Snow Day Assignment

Trevor Walker

3/4/2014

**Pt. 1**

-Adding Functionality

Derived classes allow the user to create their own versions of an existing base class. The base class’s characteristics can be changed simply by creating a derived class and adding functionality to the derived class rather than editing the base class.

-Redefining Functionality

A derived class can not only gain new variables and functionality, the functionality of an existing member can be changed so a member will have different functionality when implemented on objects created from the derived class.

-Adding to Existing Functionality

It is also possible to grant extra functionality to existing functions. Simply redefine the function, and within that, call on the existing function with a prefix (scope qualifier) for the base class. Then add whatever additional desired functionality within the redeclaration.

-Hiding Functionality

It is possible to change the access specifiers of class functions inside of a derived class. By redefining a function as a different type, such as defining a class function that is public in the base class as protected in the derived class, one can hide a certain function from a derived class so it cannot be used.

**Pt.2**

-Object Oriented Programming

OOP allows the use of objects to create better organized and reusable code. An object has states and behaviors, and these are easily changed and applied in many different ways. OOP helps to make code more universal and convenient.

-Cohesion

Cohesion is grouping together code that contributes to accomplishing a single task. Every object should have a single responsibility or purpose. Any more than that, and the code becomes convoluted and more complicated to read and edit. Having high cohesion is very important, and helps your code be understandable, readable, and more universal. Every programmer should strive to achieve very cohesive code.

-Coupling

Coupling is the principle that no object should directly affect the state or behaviors of any other object. Coupling is a helpful principle, and is very important to keeping code more flexible and universal, making it easier to update, edit and change existing code that applies OOP methodology.

-Encapsulation

Encapsulation is the idea that the internal workings (clockwork) of an object should be hidden from the rest of the program. Effectively implemented encapsulation techniques give way to loosely couple code. This, of course, helps to make code more universal and easier to play with.

-Abstraction

Abstraction, at its core, is the generalization of any aspect of your code. S good example given was the Tetris example, which shows that the tetriminos and board are both drawn with the same method, both most likely belonging to the same class. This assists in simplifying code and making code easier to understand, alter and implement.

-Inheritance

The principle of a class hierarchy. Certain classes are derivative of other ‘base’ classes, meaning these derivative classes ‘inherited’ the behaviors and states of the base classes they were derived from. It is a very useful principle, and allows the usage of different versions of a specified object which derives its core functionality from an existing object but is capable of different actions entirely, which are defined by the programmer.

**Pt.3**

Version Control

Version control is important in keeping everyone updated to the current state of the code that is being worked on. You can merge your code with the existing code, update your code to the most recent version before making changes, and even track which individual line was changed by who (called blame on github, which is hilarious). As for how this principle would have assisted Fightopia, Im not really sure how to start. It would have helped us keep updated on what is changing and how it is changing other parts of the code, and would have also allowed us to have more cohesive code that was easier to understand. Our code would have been more capable of applying more of the principles that would have made our code more…. Well, more way better in every way. (ie, no more 5000 lines in main).