Franklin Colton Parry

CS 2450 section 002

1/27/2014

Why Getters / Setters are evil

1. The authors argument is that most programs do not need all the getters and setters that in them. He argues that the design of most programs causes getters and setters to cause more complexity. One example that he gives is that if a getter method needs to return an integer and that method is implemented 1000 times, when the type of the variable changes, the method has to be changed 1000 times. He then goes on to say that these methods can be useful in some places, and in some cases are necessary. However, a good designer can remove most of the necessity of these methods. Later he gives an option for designing programs that could help eliminate these methods.

I tend to agree with what the author is saying about how young programmers often put these methods in out of tradition or in unknowing what the purpose of the program will be. I agree with this because when the authors says “You should not program blindly,” I often feel like this is what we do when we don’t have a clear understanding of the purpose of the program, or what the program is supposed to accomplish. Thus, we add these methods in case we need them.

1. Dynamic object programming allows for the changing types of an object at runtime without additional programming. Whereas static object programming is defining the relationships between objects. I think an example of this could be designing an employee form with different types of employees. Each type can have its own class and there is a relationship between each type. However, we want to make sure that we don’t have to change our programming to accommodate each class.

We care about this because of the amount of effort what we have to put in to building and maintaining a program. The author says that “without a clearly defined dynamic model, you’re only guessing… This sort of design-by-guessing strategy is inefficient at best.” No programmer wants to be inefficient or just guessing to try and get a program written.

1. The way to write software with few getters and setters stems from two things. The first thing is to have a clear cut view of what the objects in the program will do. In other words each object is in the program for a reason, and rather than just creating getters and setters because variables in the objects exist, know why they are there and if they are necessary for the program to use.

The second comes from having a proper design to follow. It is important to think about the dynamic and static object relationships. This allows for the objects to control a lot of the information flow, thus reducing the amount of getters and setters needed. If an object class controls the functions then there is no need to write non object functions.