Class Design Guidelines – Encapsulation

Encapsulation is extremely important in certain programs as privacy can be a huge issue for clients. Certain attributes of a program need to be kept private from direct access. Examples of these could be passwords or codes. Deciding early on what classes you want to make private or public will make the class easy to maintain as your program progresses. The manipulation of data access is a very important tool within java that every programmer needs to know if they wish to provide a basic level of security in their work. An example of how basic encapsulation can be achieved is by declaring all your variables in a class as private and writing your public methods in the class to set and get the values of those private variables. The data within a class is hidden from other classes and can be accessed only by member functions of their own class which they are declared in.

Some of the main advantages I will highlight from encapsulation are as follows:

Hiding data – Users will not be able to see the inner implementation of the private class. Storing values in the variables will be unseen and will only be able to access the values passed to a setter method. Variables can be initialized with the hidden values.

Reusability – This also falls under consistency and cohesion. The new requirements that come with encapsulation allow it to be used multiple times throughout the entire program.

Flexibility – There are quite a few ways to implement these processes, such as making the variables of a class read-only or write-only depending on what we’re trying to accomplish.

If the programmer is aiming to make the variables read-only then you want to avoid setter methods like setSalary() or setWeight(). Or if you’re aiming for the latter, and want to make them write-only then you would avoid getter methods like getSalary() or getWeight().