Object Oriented Vocabulary

Private – Variables and methods that are declared private are only accessible within that class.

Public – Variables and methods that are declared public are accessible anywhere.

Encapsulation – A principle in Object Oriented Design that states:

- *Bullet Proof Classes* - You should make your classes unbreakable. For example: denominators in Fraction class were impossible to set to zero.
- *Information Hiding* – just give client classes essential ways to interact with your objects and hide all the messy details in order to make working with your objects as simple as possible.
- *Self-Sufficient* – Your classes are complete, and contain everything client codes needs to interact with objects of this type.

Objects vs Classes: Think of a class as a blueprint, and objects as the things that are built from those blueprints. Objects are *instantiated* from their classes using the *new* keyword.

Static – belongs to the class, not the object. For static variables, there is only one value for all objects of that class. For a static method, you call it from the class not the object.

Non-Static – Belongs to the object, not the class. Each object gets its own unique copies of non-static variables (called *instance variables*). Non-static methods are called from the object itself and treat values of the instance variables as those of the object from which they were called. For example, if you had a Fraction object called f1, if you called f1.getNumerator() then that method would get the numerator belonging to the f1 object. Variables and Methods are by default non-static; you do not need to tell java this.

Final – A variable declared as final cannot change its value. Note the difference between static and final, as students often confuse the two.

Constructor – The method that gets called when you instantiate an object and initializes the instance variables. It must have the same name as the class. It has no return type and is public.

Accessor Methods – "getters" return the values of private instance variables. They don't change the values. Be careful not to return references to private objects! Always public.

Mutator Methods – "setters" that change the values of private instance variables. You can embed error checking and general safety code to make sure your objects are unbreakable. Always public; mutators that are private are called Helper methods.

Helper Methods – Other methods that help accessors and mutators but are not made public to keep interactions with your class simple. They may change instance variables.

Overloading – making multiple methods with the same name but different parameters. For example, our Fraction class had overloaded constructors