**Single Responsibility Principle**

This principle states that every module, class or function should have a single responsibility. The order and program classes violate this principle.

Code handling logging errors and orders has been moved to the Logger class. The program class now has the single responsibility of generating orders.

Code handling database interactions has been moved out of the Order class to the OrdersDB class. This means the Order class now only has the single responsibility of taking in and storing an orders information.

**Open Closed Principle**

This principle states that entities should be open for extension but closed for modification. To allow extension without modification, each class now has an interface which can be used to extend the functionality of the class.

**Liskov Substitution Principle**

This principle pertains states that functions that use pointers to base classes must be able to use objects of derived classes without knowing it. This program does not have any inheritance therefore it does not apply.

**Interface Segregation Principle**

This principle sates that no client should be forced to depend on methods it does not use. Large interfaces should be split into smaller more specific ones. This program now only has one interface and it can be ignored if it’s not needed.

**Dependency Inversion Principle**

This principle states that higher level modules should not depend on low level modules. I created a Static class called Factory. This class is responsible for newing up classes. Classes such as logger are passed into the constructor of classes that require it. This breaks the dependency between classes.