**Single Responsibility Principal**

The order and program classes violate the single responsibility principle. The class should only have one job.

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 storing an orders information.

**Open Closed Principal**

This principle states that “entities should be open for extension but closed for modification”. To allow extension of the Order class without modification, I have created an IOrder Interface which can be used to add properties or functionality.

**Liskov Substitution Principal**

This principle pertains to class inheritance. This program does not have any inheritance, so it doesn’t apply.

**Interface Segregation Principal**

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 Principal**

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.