COMP319 Open Closed Principle

This principle describes that an object class must be closed for modification but open for extension.

This provides the benefit of keeping the current code base stable and bug free but allowing it to be used as a basis for future code development.

One way to close the Java class is to define its interface in a Java interface definition. If the class is only used via this abstract interface, changes to the class definition itself will not change its interface.

The second way you can lock a class from modification is to use the word final on its method definitions.

Once a method is marked this way the method cannot be overridden.

Finally the use of a protected overridable method can be used as an extension point to actually implement code that can be added to the base class.

Here is an example, starting with the interface definition (note the class will always be used via its public interface).

```
public interface ICardProvider {
        public boolean makePayment(String number);
}
```

Next have a base class which will contain all the default behavior of the service.

```
public class CardProviderBase implement ICardProvider {
        public final boolean makePayment(String number) {// This is closed for modificatin
                // Default code to test for stolen cards etc.
                // If cardStolen(cardNumber)) return(false);
                return(onMakePayment(number));
        }
        protected boolean onMakePayment(String number) {   // This is open for extension
                return(false);              // default implementation just fails
```

```
        }
}
public class NatwestCardProvider extends CardProviderBase {
                protected boolean onMakePayment(String number) {
                        return(true);
                }
        }
}
```