A new pattern to keep Common Interfaces

While interfacing UI with its corresponding functionality, every developer must have come across a common problem, i.e. multiple views may have same UI components demanding for same functionality. For example two views having same menu options invoking same code but since these views are different one can not share the code.

So one solution can be as follows

Create the interfaces in an abstract class

```
class MInterface {
   public:
      void MyInterface() = 0;
}
```

Implement your interface with the AppUi class

```
class CMyApplicationUi : public CAknViewAppUi, public MInterface {
.
.
.
}
```

Now with the help of `AppUi` reference these functions are easily accessible from any view as shown below.

```
// some view
CMyApplicationUi * ui = static_cast<CMyApplicationUi*>(AppUi());
ui->MyInterface();
```

One of the advantages of this method is that all the user interface components are easily accessible in `AppUi` class, so UI related code which is common and needs sharing can be placed here.

But if the UI related code is large in amount which needs to be shared by multiple views then only inference is the design needs improvement and in such case this solutions will be an over kill. Hence this solution need to be used carefully depending on the usage context.

This method might not be a standard practice but it's a common solution to a commonly seen problem.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.