MVP design pattern

The aim of it is to separate the app into 3 independent layers, to keep all the code logic away from the UI.

The three layers are:

1.Model

2.View

3.Presenter

The Model is the data center of the app. All the data is kept here and when the model is changed, instead of updating the view directly, the presenter is notified to update the views accordingly.

The Presenter is where the business logic is kept. It is responsible for keeping the view and model away from each other. View and Model can only talk through the Presenter.

The View is responsible for deciding how the data is to be displayed to the user. Usually it's an activity or fragment.

While the three layers are separated and can't talk to each other, they will communicate using interfaces, so, a set of rules must be defined that the Model, presenter and the view will follow. And here comes the contract, that contains three interfaces, each one for each layer.

The Contract is the set of rules of communication between layers.

1. **The view** will implement the View interface. The view only needs to know about the presenter so an instance of the presenter is initiated in the View.
2. **The presenter** acts as an intermediate for the view and the model. So, the presenter needs to know about both, the view and the model.

Note: As the presenter is instantiated from the View, every presenter should be attached to a view that it will interact with. So, an instance of the view interface is passed into the constructor. When the presenter is instantiated, an instance of the model is also created.

3. **The model** contains only the data and as in our sample app the data is just a variable that is updated each time the button is clicked. So will keep a member variable and implement the Model interface whose callback methods will be called when the data is to be accessed or updated.

***MVVM comments and explanations are provided with the code.