SecondViewController.sourceViewController = self;

@property(nonatomic,copy)NSString\* message

@property(weak,nonatomic)

sourceViewController

self.sourceViewController.message= XXXXX

C1

C2

**Communication Inversion**

M2

V1

M1

V2

**C1 following C2’s protocol while C2 remains independent**

**@protocol “required methods”…**

**@property (nonatomic,weak)id<C2>Delegate**

C1

**Protocol**

C2



**Communication Inversion 2**

**(delegate)**

V2

M2

M1

V1

**C2.delegate = self;**

**-(XXX) requiredMethods{XXX}**

**Desired Action**

C1

C2

**Action In Reality**

**Direct Communication**

M1

V1

V2

M2

**NavigationController**

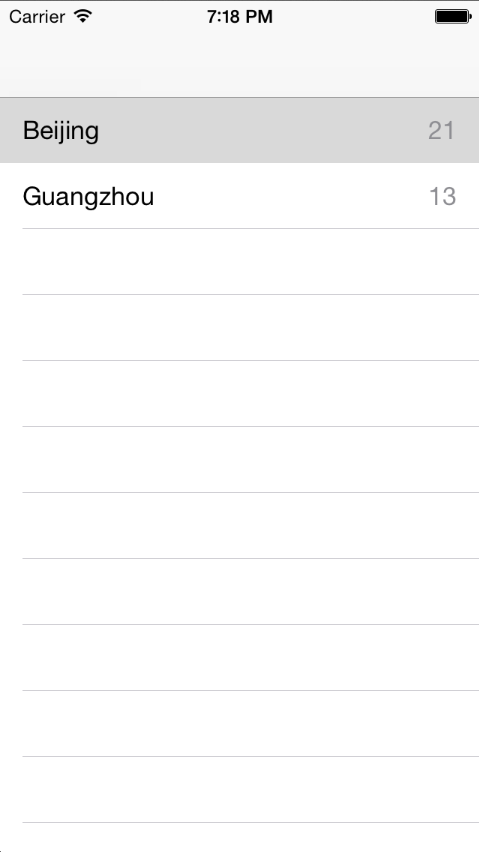
VC2

VC1

Item1

The Direction You Watch

Item2



C1

M1

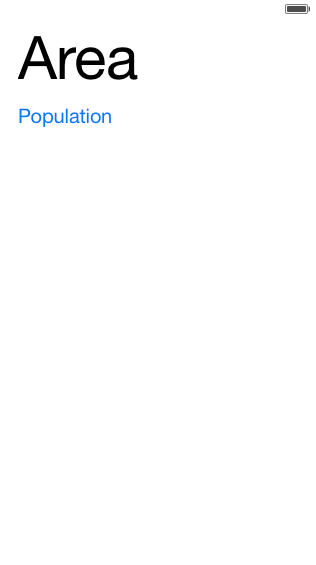


**Property**





**Data**



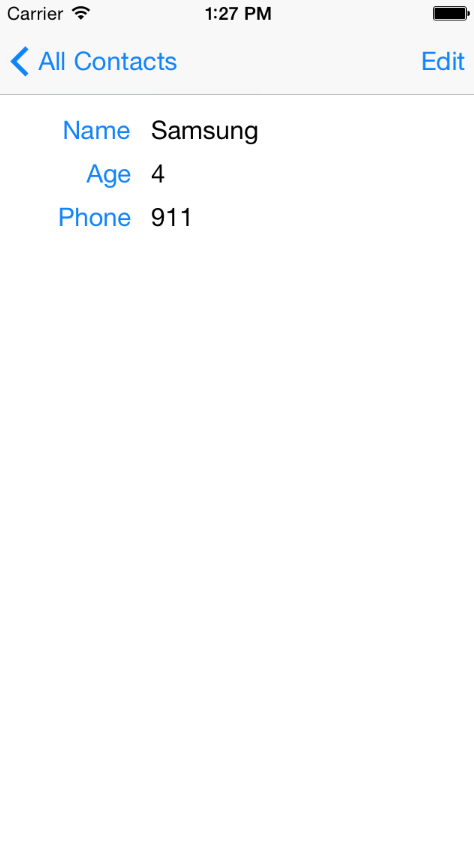
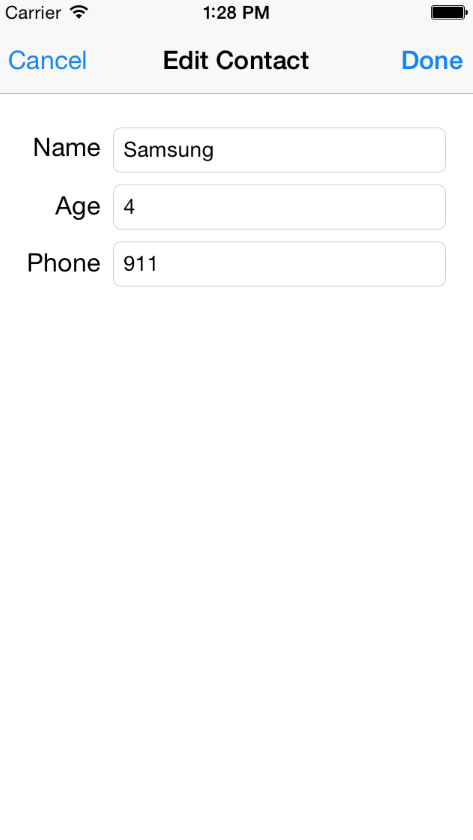
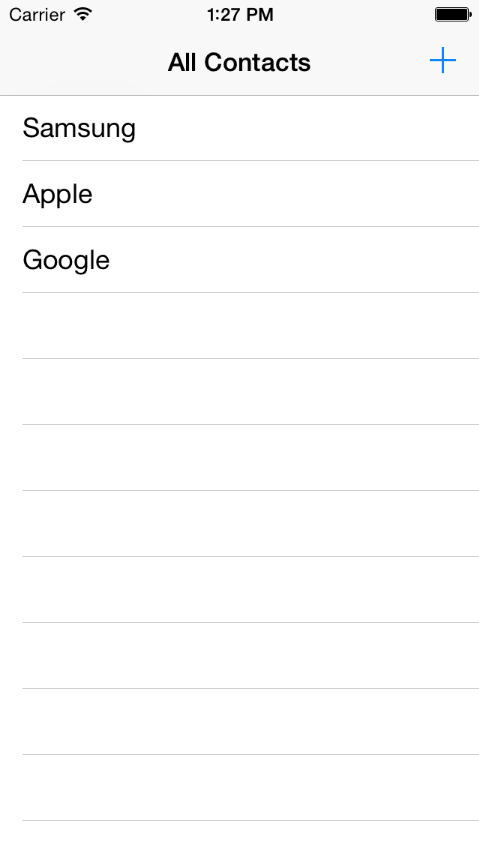




C2

Always send the object to other view controllers. If you change one of the properties later, then you don’t worry about changing all the properties in others views. Because you have sent the object (the whole package) to them.





They are the same object sharing the same pointer

**NSMutableArray**



**TRContact**

**TRContact**

C3

C1

C2

This inverse communication is done in a different way. Because C2 passed the pointer to C3. You just need to Edit the object’s data(not creating), and assign them to your own property. In other words, you are manipulating the object itself(In model layer)

This inverse communication is delegate. Because you passed the pointer that needs to create new data and to be transferred back to its initiator



TRArea

NSString

NSArray

TRArea

TRArea

NSString

NSString

NSArray

NSArray

TRArea

TRArea

TRArea

TRArea

NSString

NSString

NSString

NSString

NSArray

NSArray

NSArray

NSArray

TB

NSString

TRViewController

@property int sunCount

TRSunFlower

you wanna access **sunCount** in the class from the left.

@property (nonatomic,weak) TRViewController \*viewController

then you can access the property **sunCount** through **viewController** property

Normally, we use the name **delegate** instead of **viewController**

@property (nonatomic,weak) TRViewController \*delegate



E

A

C

B

D



In this case, you have a button in View E, however, you want to design a feature that ,by taping button in View E, you can change all the labels separated in View A,B,C,D. You need the feature notification to implement this.