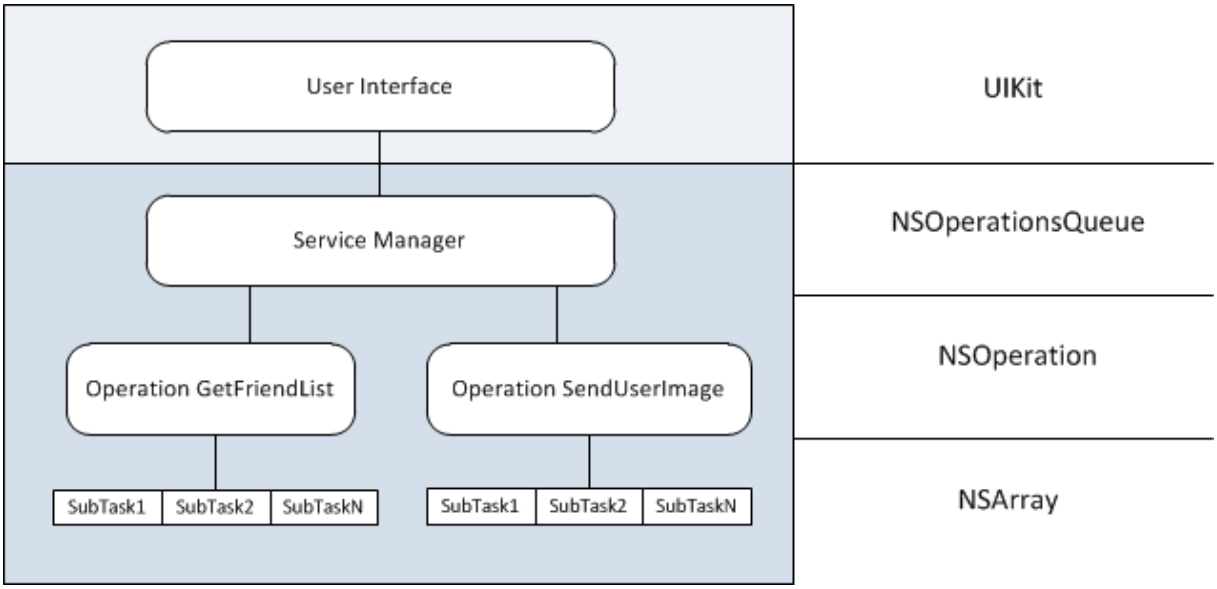
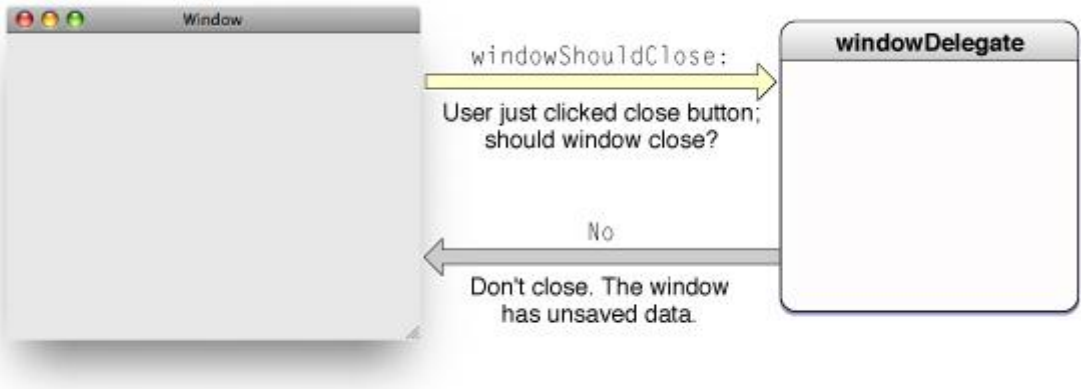
1. Opisz architekturę aplikacji iOS.



2. Wzorzec delegate w iOS, omów i podaj zastosowanie.

Delegation is a simple and powerful pattern in which one object in a program acts on behalf of, or in coordination with, another object. The delegating object keeps a reference to the other object—the delegate—and at the appropriate time sends a message to it.



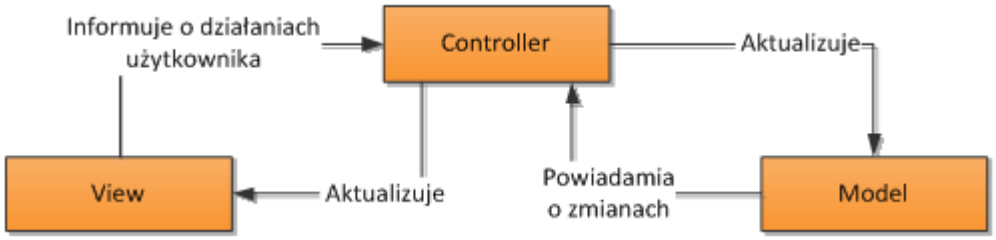
An example of a delegating object is an instance of the NSWindow class of the AppKit framework. NSWindow declares a protocol, among whose methods is windowShouldClose:. When a user clicks the close box in a window, the window object sends windowShouldClose: to its delegate to ask it to confirm the closure of the window.

3. Opisz wzorzec MVC.

Przedstawia on logiczny podział kodu aplikacji z graficznym interfejsem użytkownika na trzy elementy:

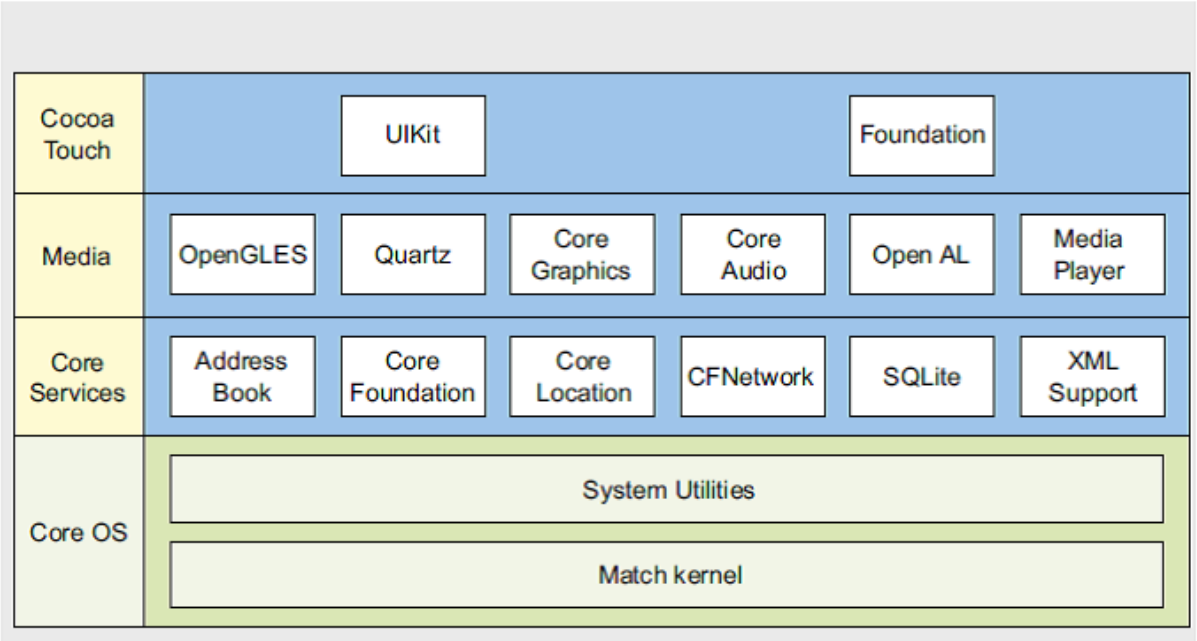
- Model - wszelkie klasy odpowiedzialne za dane naszej aplikacji,
- View - elementy interfejsu graficznego użytkownika, okna, widoki, przyciski, suwaki itd.,

- Controller - kod wiążący interfejs aplikacji z jej danymi.
Głównym celem wzorca jest taka separacja kodu, aby wszystkie wspomniane elementy były od siebie jak najbardziej niezależne.



4. Struktura systemu iOS, wymień warstwy i opisz krótko każdą z nich.

iOS składa się z 4 abstrakcyjnych warstw:



- Core OS – Najniższa warstwa, zapewniająca interakcję między sprzętem a oprogramowaniem, w jej skład wchodzi jądro Darwin.
- Core Services – To rdzenny zestaw podstawowych bibliotek do zarządzania pracą aplikacji i wątków, obsługa sieci, obsługa bazy danych (SQLite) i inne, których działanie nie jest bezpośrednio widoczne dla użytkownika.
- Media – To warstwa zawierająca obsługę obrazu i dźwięku, również odtwarzanie wideo i obsługę formatów plików graficznych, w skład warstwy wchodzą znane biblioteki takie jak OpenGL, OpenAL czy Core Animation.
- Cocoa Touch – Jest to biblioteka interfejsu użytkownika z wykorzystaniem ekranu dotykowego, co różni go od tradycyjnego Cocoa z systemu OS X. W jego skład wchodzi również obsługa akcelerometru.

5. Wymień podstawowe klasy języka Objective-C (NSString, NSNumber, NSDictionary, NSArray, NSSet) i opisz do czego służą.

An NSString object encodes a Unicode-compliant text string, represented as a sequence of UTF–16 code units.

NSNumber provides readonly properties that return the object's stored value converted to a particular Boolean, integer, unsigned integer, or floating point C scalar type.
The NSDictionary class declares the programmatic interface to objects that manage immutable associations of keys and values.
NSArray is Objective-C's general-purpose array type. It represents an ordered collection of objects, and it provides a high-level interface for sorting and otherwise manipulating lists of data.
NSSet object represents a static, unordered collection of distinct objects.

6. Czym charakteryzuje się klasa, w której nazwie znajduje się słowo „Mutable" oraz jaka jest różnica w stosunku do klasy, która tego słowa nie posiada?

A mutable object can be mutated or changed. An immutable object cannot. For example, while you can add or remove objects from an NSMutableArray, you cannot do either with an NSArray.

Mutable objects can have elements changed, added to, or removed, which cannot be achieved with immutable objects. Immutable objects are stuck with whatever input you gave them in their [[object alloc] initWith...] initializer.

7. Podstawowe elementy klasy w języku Objective-C - czym jest interface, implementation, property.

Interfejs w Objective-C jest częścią deklaracji samej klasy.

W interfejsie definiujemy instancje zmiennych, właściwości i metody które nasza klasa będzie miała. W implementacji piszemy ich użycie.

In Objective-C, the class interface specifies exactly how a given type of object is intended to be used by other objects.

The goal of the @property directive is to create and configure properties by automatically generating accessor methods. It allows you to specify the behavior of a public property on a semantic level, and it takes care of the implementation details for you.

8. Jakie rodzaje metod może posiadać klasa w języku Objective-C, opisz różnice pomiędzy nimi oraz napisz przykładowy fragment kodu do każdej z nich, prezentujący sposób wywołania.

a class method is denoted by a plus (+) sign at the beginning of the method declaration and implementation:
+ (void)classMethod;

To send a message to a class, you put the name of the class as the receiver in the message expression:
[MyClass classMethod];

send class messages to subclasses of the class that declared the method:
NSMutableArray *aMutableArray = [NSMutableArray array];

Class methods can't refer directly to instance variables:

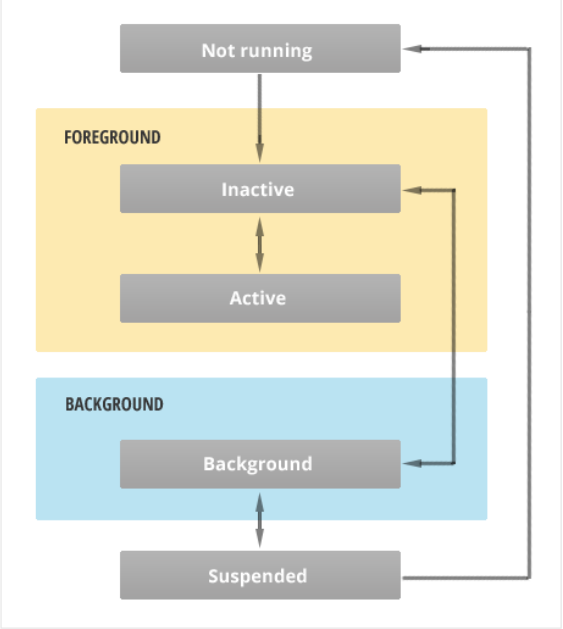
```
@interface MyClass : NSObject {
   NSString *title;
}
+ (void)classMethod;
@end
```

a class method, self refers to the class object itself:

```
+ (id)myClass {
   return [[[self alloc] init] autorelease];
}
```

9. Opisz cykl życia aplikacji iOS.

Wyróżniamy dwa stany, w jakim aplikacja może się znajdować: stan aktywny lub praca w tle. Aplikacja musi wiedzieć, w którym z tych stanów jest i odpowiednio na to reagować.
Za komunikację z systemem operacyjnym i otrzymywanie informacji o zmianie stanu aplikacji jest klasa Application Delegate.

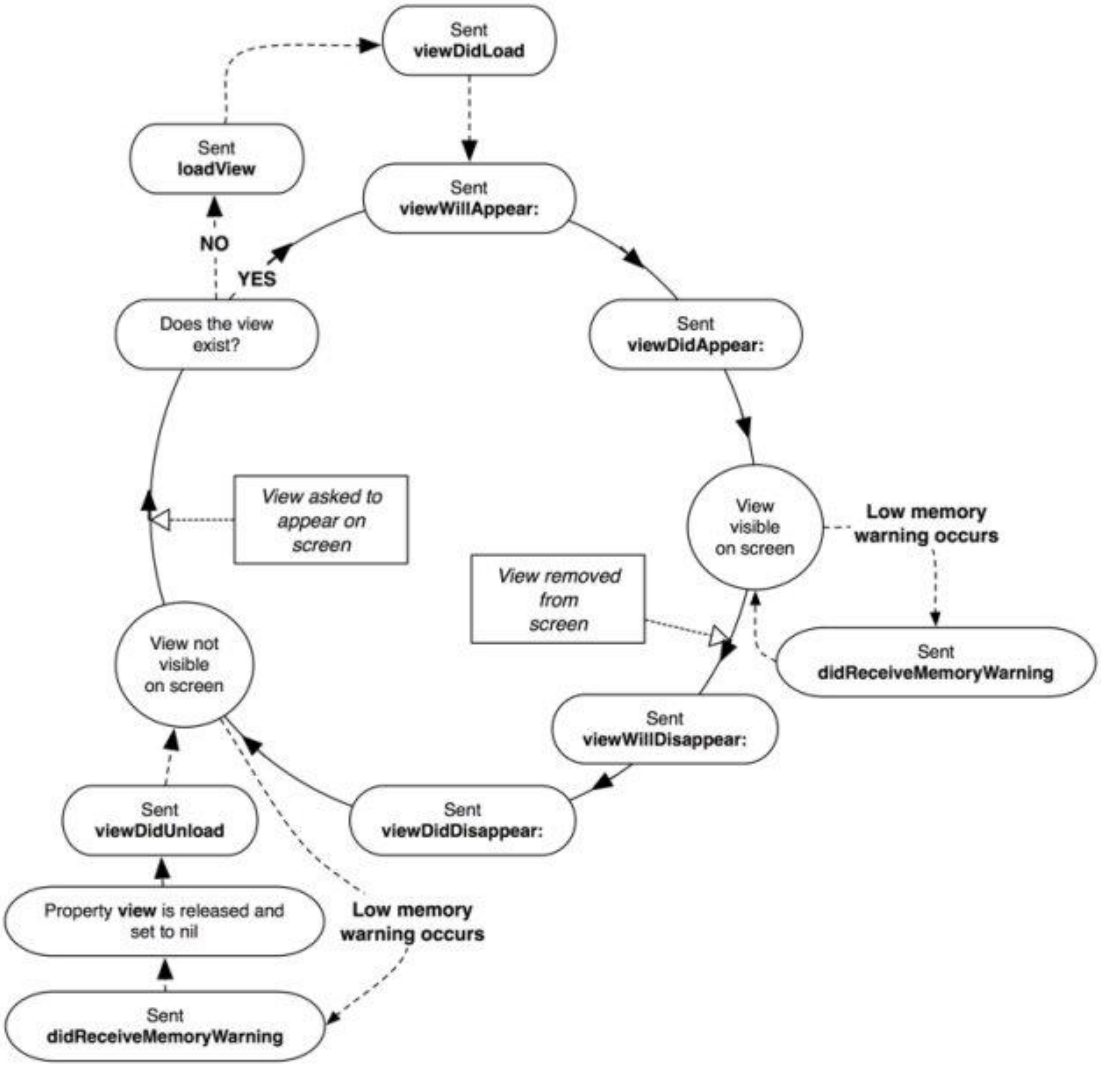


10. Co to jest UIViewController i do czego jest wykorzystywany.

The UIViewController class provides the infrastructure for managing the views of your iOS apps. A view controller manages a set of views that make up a portion of your app's user

interface. It is responsible for loading and disposing of those views, for managing interactions with those views, and for coordinating responses with any appropriate data objects. View controllers also coordinate their efforts with other controller objects

11. Opisz cykl życia klasy UIViewController.



ViewDidLoad - Called after the controller's view is loaded into memory.
ViewWillAppear - Called right before your view appears. this will be called every time your view is about to appear on the screen.
ViewDidAppear - Called after the view appears
ViewWillDisappear/DidDisappear - Same idea as ViewWillAppear/ViewDidAppear.
ViewDidUnload/ViewDidDispose - In Objective C, this is where you do your clean-up and release of stuff, this is handled automatically.

12. Czym jest CoreData, wymień z czego się składa i opisz krótko każdy z elementów.

Core Data is a framework that you use to manage the model layer objects in your application.

Create generalized and automated solutions to common tasks associated with object life-cycle and object graph management, including persistence.
Core Data provides object graph management and persistence for Foundation and Cocoa applications.

- NSManagedObjectModel instance describes the data that is going to be accessed by the Core Data stack.
- NSPersistentStoreCoordinator sits in the middle of the Core Data stack. The coordinator is responsible for realizing instances of entities that are defined inside of the model.
- NSManagedObjectContext is the object that your application will be interacting with the most, and therefore it is the one that is exposed to the rest of your application.

13. Komunikacja pomiędzy view controllers, czym jest segue oraz w jaki sposób można przesyłać dane z jednego view controller do innego.

A segue defines a transition between two view controllers in your app's storyboard file. The starting point of a segue is the button, table row, or gesture recognizer that initiates the segue. The end point of a segue is the view controller you want to display.

W aplikacjach wykorzystujących Storyboards do przekazywania aplikacji pomiędzy scenami wykorzystujemy metodę prepareForSegue.

14. Co to jest Storyboard i do czego służy.

A storyboard is a visual representation of the user interface of an iOS application, showing screens of content and the connections between those screens. A storyboard is composed of a sequence of scenes, each of which represents a view controller and its views; scenes are connected by segue objects, which represent a transition between two view controllers.

15. Sposób łączenia elementów interfejsu użytkownika z kodem aplikacji, jaka jest różnica pomiędzy połączeniem typu action a połączeniem typu outlet.

IBOutlet oraz IBAction.

Są to tak zwane kwalifikatory typów i to właśnie dzięki nim kontroler może komunikować się z interfejsem naszej aplikacji. Za ich pomocą Xcode synchronizuje elementy klasy, właściwości i metody z elementami interfejsu użytkownika.

Kwalifikatory zadeklarowane są następująco w pliku UINibDeclarations.h:
#define IBAction void
#define IBOutlet
Aby Xcode synchronizował określoną metodę naszej klasy z Interface Builder-em kwalifikator IBAction musi być użyty, jako typ zwracany przez nasza metodę.

-(IBAction)przeliczNapiwek:(id)sender;

Powyższa deklaracja jest typową dla metod action wykorzystywanych w mechanizmie komunikacji target-action pomiędzy interfejsem a kontrolerem.

Do synchronizowania właściwości klasy (property) wykorzystywany jest kwalifikator IBOutlet. Powinien być on umieszczony bezpośrednio przed typem danej właściwości.

@property (nonatomic, weak) IBOutlet UITextField *rachunek;

16. Co to jest UITableView i do czego służy.
17. Co to jest UICollectionView i do czego służy.
18. W jaki sposób działa mechanizm wyświetlania danych w UITableView oraz UICollectionView.

A table view displays a list of items in a single column. UITableView is a subclass of UIScrollView, which allows users to scroll through the table, although UITableView allows vertical scrolling only.

UICollectionView manages an ordered collection of data items and presents them using customizable layouts. When adding a collection view to your user interface, your app's main job is to manage the data associated with that collection view.

19. Jaka jest różnica pomiędzy UITableView a UITableViewController.

An instance of UITableView (or simply, a table view) is a means for displaying and editing hierarchical lists of information.

The UITableViewController class creates a controller object that manages a table view.

When the table view is about to appear the first time it's loaded, the table-view controller reloads the table view's data. It also clears its selection (with or without animation, depending on the request) every time the table view is displayed.

20. Czym są testy jednostkowe i do czego służą.

Test jednostkowy – metoda testowania tworzonego oprogramowania poprzez wykonywanie testów weryfikujących poprawność działania pojedynczych elementów (jednostek) programu np. metod lub obiektów.

Testy jednostkowe pozwolą zautomatyzować proces testowania aplikacji, zmniejszając liczbę błędów, a już na pewno powinny się przyczynić do tego, że błąd który raz wystąpił w przeszłości i został poprawiony, już nigdy nie powinien się powtórzyć.