Pytanie 67: Jaka jest różnica między delegacją a KVO?

Oba są sposobami tworzenia relacji między obiektami. Delegacja to relacja jeden do jednego, w której jeden obiekt implementuje protokół delegata, a inny używa go i wysyła komunikaty, zakładając, że te metody są zaimplementowane, ponieważ odbiorca obiecuje zgodność z protokołem. KVO to relacja wiele do wielu, w której jeden obiekt może nadawać wiadomość, a jeden lub wiele innych obiektów może słuchać i reagować na nią.

Pytanie 68: Co to jest zawijanie metod? Kiedy go użyjesz?

Swizzling metod to proces zmiany implementacji istniejącego selektora. Jest to technika możliwa dzięki temu, że wywołania metod w Objective-C można zmieniać w czasie wykonywania, zmieniając sposób mapowania selektorów do podstawowych funkcji w tabeli rozsyłania klasy. Załóżmy na przykład, że chcieliśmy śledzić, ile razy każdy kontroler widoku jest prezentowany użytkownikowi w aplikacji na iOS. Każdy kontroler widoku mógłby dodać kod śledzenia do swojej własnej implementacji viewDidAppear :, ale dałoby to mnóstwo zduplikowanego standardowego kodu. Podklasy byłyby kolejną możliwością, ale wymagałyby podklasy UIViewController, UITableViewController, UINavigationController i każdej innej klasy kontrolera widoku - podejście, które również ucierpiałoby z powodu duplikacji kodu.

Pytanie 69: Weź trzy przedmioty: dziadka, rodzica i dziecko.

Dziadek zatrzymuje rodzica, rodzic dziecko, a dziecko rodzica. Dziadek wypuszcza rodzica. Co się dzieje?

Następuje cykl utrzymania. Ponieważ są zachowane, nie będzie można ich uwolnić z pamięci. Można to rozwiązać, osłabiając jedno z odniesień.

Pytanie 70: Jakie są dwa oddzielne i niezależne powody, dla których-retainCount nigdy nie powinno być używane w kodzie wysyłkowym?

Nigdy nie powinieneś używać -retainCount, ponieważ nigdy nie mówi ci on niczego pożytecznego. Implementacja frameworków Foundation i AppKit / UIKit jest nieprzejrzysta. Nie wiesz, co jest zatrzymywane, dlaczego jest zatrzymywane, kto to zachowuje, kiedy zostało zachowane i tak dalej. Oto przykłady, kiedy używać - retainCount:

• [NSNumber numberWithInt: 1] ma teraz retainCount równy 9223372036854775807. Jeśli Twój kod oczekiwał, że będzie to 2, Twój kod został uszkodzony.

• Można by pomyśleć, że @ „Foo” miałoby retainCount równe 1. Tak nie jest. Jest to 1152921504606846975.

• Można by pomyśleć, że [NSString stringWithString: @ "Foo"] miałoby retainCount równe 1. Tak nie jest. Znowu jest to 1152921504606846975.

Zasadniczo, ponieważ wszystko może zachować obiekt (a tym samym zmienić jego retainCount), a ponieważ nie masz źródła większości kodu, który uruchamia aplikację, parametr retainCount obiektu jest bez znaczenia. Jeśli próbujesz wyśledzić, dlaczego obiekt nie jest zwalniany, użyj narzędzia wycieków w Instruments. Jeśli próbujesz wyśledzić, dlaczego obiekt został zwolniony zbyt wcześnie, użyj narzędzia zombie w Instrumentach. Ale nie używaj -retainCount. To naprawdę bezwartościowa metoda.

Pytanie 71: Jak działa pula automatycznego wydania na poziomie środowiska wykonawczego?

Za każdym razem, gdy obiekt -autorelease jest wysyłany do obiektu, jest on dodawany do najbardziej wewnętrznej puli autorelease. Kiedy basen jest opróżniony, po prostu wysyła -release do wszystkich

obiektów w basenie. Pule autowydzielania to po prostu wygoda, która pozwala na odroczenie wysyłania i zwalniania do „później”. To „później” może się zdarzyć w kilku miejscach, ale najczęściej w aplikacjach GUI Cocoa występuje na końcu bieżącego cyklu pętli uruchamiania.

Pytanie 72: Co jest szybsze: iterować przez NSArray lub NSDictionary?

Gdy kolejność elementów w kolekcji nie jest ważna, zestawy zapewniają lepszą wydajność wyszukiwania elementów w kolekcji. Powodem jest to, że zestaw używa wartości skrótu do wyszukiwania elementów (takich jak słownik), podczas gdy tablica musi iterować po całej swojej zawartości, aby znaleźć określony obiekt.

Pytanie 73: Co jest szybsze: iteracja przez NSArray lub NSSet?

NSArray jest szybszy niż NSSet pod względem prostego trzymania i iteracji. Na niższym końcu konstruowanie jest nawet o 50% szybsze, a na wyższym nawet o 500% szybsze przy iteracji. Jeśli potrzebujesz tylko iteracji zawartości, nie używaj NSSet.

Pytanie 74: Czy musisz realizować wszystkie deklaracje z przyjętego protokołu?

Nie. Jeśli metoda jest zadeklarowana jako @optional, nie musi być implementowana.

@protocol MyProtocol

- (void) requiredMethod;

@optional

- (void) anOptionalMethod;

@required

- (void)anotherRequiredMethod;

@end

- (void) anotherOptionalMethod;

Jeśli programista zostanie poproszony o zidentyfikowanie ryzyka za pomocą tego podejścia, powinien być w stanie rozwinąć koncepcję, że jeśli metoda w protokole jest oznaczona jako opcjonalna, należy sprawdzić, czy obiekt implementuje tę metodę, zanim spróbujesz ją wywołać .

Pytanie 75: Jaki jest skrót do wywoływania funkcji przydzielania i inicjalizacji?

Alokację i init można ogólnie nazwać w ten sposób:

[[Class alloc] init]

W innych fragmentach kodu oraz w literaturze możemy również znaleźć:

[Class new]

Pierwotnie w Objective-C obiekty były tworzone z nowym. Wraz z ewolucją frameworka OpenStep / Cocoa projektanci doszli do wniosku, że alokacja pamięci dla obiektu i inicjalizacja jego atrybutów to odrębne kwestie, a zatem powinny być oddzielnymi metodami (na przykład obiekt może być przydzielony w określonej strefie pamięci). Tak więc styl tworzenia obiektów przydzielać-init zyskał na popularności. Zasadniczo nowy jest stary i prawie - ale nie - całkiem przestarzały. Dlatego zobaczysz,

że klasy Cocoa mają wiele metod inicjujących, ale prawie nigdy nie mają żadnych nowych niestandardowych metod.

Pytanie 76: Jaki rodzaj wskaźnika może pomóc w bezpiecznym uniknięciu wycieku pamięci?

Inteligentne wskaźniki mogą być bardzo pomocne w automatyzacji księgowania okresów życia obiektów. Inteligentny wskaźnik to klasa, która otacza „surowy" (lub „pusty") wskaźnik C ++ w celu zarządzania czasem życia wskazywanego obiektu. Nie ma jednego inteligentnego typu wskaźnika, ale wszystkie z nich próbują wyabstrahować surowy wskaźnik w praktyczny sposób. Wskaźniki inteligentne powinny być preferowane zamiast wskaźników surowych. Jeśli czujesz, że musisz użyć wskaźników (najpierw zastanów się, czy naprawdę to robisz), normalnie chciałbyś użyć inteligentnego wskaźnika, ponieważ może to złagodzić wiele problemów z surowymi wskaźnikami, głównie zapominanie o usunięciu obiektu i wyciek pamięci.

Pytanie 77: Co może pomóc w zapobieganiu awarii związanej z brakiem pamięci, jeśli masz długotrwałą pętlę wykonywania?

Tutaj jednym kluczem są wycieki pamięci. Upewnij się, że pamięć jest czyszczona i nie przechowujesz żadnych odwołań, które uniemożliwiają zebranie obiektu. Poszukaj również komunikatu WillTerminate aplikacji w delegacie aplikacji. Dzieje się tak, jeśli aplikacja zostanie zamknięta przez system (np. Z powodu małej ilości pamięci), ale nie, jeśli użytkownik opuści aplikację w zwykły sposób, naciskając klawisz home.

Pytanie 78: Jakie są wymagane rozważania podczas pisania

UITableViewController, który wyświetla obrazy pobrane ze zdalnego serwera?

Samo zaprogramowanie tej funkcji może być zadaniem programistycznym dla potencjalnego kandydata. Jeśli jednak pytanie jest zadawane ustnie, kandydat powinien odpowiedzieć, przytaczając kilka ogólnych wskazówek, jak radzić sobie z aspektem przechowywania i asynchroniczności wynikającym z tego problemu. Niektóre z punktów do omówienia to

• Pobieraj obraz tylko wtedy, gdy komórka jest przewijana do widoku, tj. Gdy wywoływana jest funkcja cellForRowAtIndexPath.

• Pobierz obraz asynchronicznie w wątku w tle, aby nie blokować interfejsu użytkownika, aby użytkownik mógł przewijać dalej.

• Po pobraniu obrazu dla komórki musimy sprawdzić, czy ta komórka jest nadal widoczna lub czy została ponownie wykorzystana przez inny fragment danych. Jeśli został użyty ponownie, powinniśmy odrzucić obraz. W przeciwnym razie musimy wrócić do głównego wątku, aby zmienić obraz na komórce.

W zależności od dokładności konwersacji możesz skierować tę dyskusję na sposób buforowania obrazów dla późniejszego użytkownika offline, użycie symboli zastępczych itp.

Pytanie 79: Co to jest KVC i KVO? Jaki jest przykład użycia KVC do ustawienia wartości?

KVC oznacza „kodowanie klucz-wartość". Jest to mechanizm, za pomocą którego można uzyskać dostęp do właściwości obiektu za pomocą ciągów znaków w czasie wykonywania, zamiast statycznej znajomości nazw właściwości w czasie projektowania. KVO oznacza „obserwację klucz-wartość" i pozwala kontrolerowi lub klasie obserwować zmiany wartości właściwości. Na przykład, jeśli klasa ma nazwę właściwości

@property (nonatomic, copy) NSString *name;

możemy uzyskać do niego dostęp za pomocą KVC w następujący sposób:

NSString *n = [object valueForKey:@"name"]

i możemy zmodyfikować jego wartość wysyłając wiadomość:

[object setValue:@"Mary" forKey:@"name"]

Pytanie 80: Jakie mechanizmy zapewnia iOS do obsługi wielowątkowości?

Istnieje wiele różnych technik, których możemy użyć w iOS, aby zapewnić wielowątkowość aplikacji.

• NSThread tworzy nowy wątek niskiego poziomu, który można uruchomić, wywołując metodę start.

• NSOperationQueue umożliwia tworzenie puli wątków i używanie ich do równoległego wykonywania operacji NSO. NSOperations można również uruchomić w głównym wątku, prosząc NSOperationQueue o mainQueue.

• Grand Central Dispatch (GCD) to nowoczesna funkcja Objective-C, która zapewnia bogaty zestaw metod i interfejsów API do obsługi typowych zadań wielowątkowych. GCD zapewnia sposób kolejkowania zadań do wysłania w głównym wątku, w kolejce współbieżnej (zadania są uruchamiane równolegle) lub w kolejce szeregowej (zadania są uruchamiane w kolejności FIFO)

Pytanie 81: Jaki jest łańcuch odpowiedzi?

Gdy zdarzenie wystąpi w widoku, na przykład zdarzenie dotykowe, widok uruchomi zdarzenie w łańcuchu obiektów UIResponder skojarzonych z klasą UIView. Pierwszym UIResponderem jest sam UIView. Jeśli nie obsłuży zdarzenia, kontynuuje łańcuch, dopóki nie obsłuży go UIResponder. Łańcuch będzie zawierał UIViewControllers, nadrzędne UIViews i powiązane z nimi UIViewControllers. Jeśli żadna z nich nie obsługuje zdarzenia, wówczas okno UIWindow jest pytane, czy może je obsłużyć, a na koniec, jeśli to nie obsługuje zdarzenia, jest zadawane pytanie UIApplicationDelegate. Ponieważ wszystko to jest dość hierarchiczne, można by poprosić kandydata o to aby narysował to na tablicy.

Pytanie 82: Jaka jest różnica między używaniem pełnomocnika a powiadomieniem?

Możesz myśleć o delegatach jak o rozmowie telefonicznej. Dzwonisz do swojego kumpla i specjalnie chcesz z nią porozmawiać. Możesz coś powiedzieć, a ona może odpowiedzieć. Możesz rozmawiać, dopóki nie odłożysz słuchawki. Delegaci, w podobny sposób, tworzą łącze między dwoma obiektami i nie musisz wiedzieć, jakiego typu jest delegat; po prostu musi zaimplementować protokół. Z drugiej strony NSNotification jest jak stacja radiowa. Nadaje wiadomości każdemu, kto chce słuchać. Stacja radiowa nie może otrzymywać informacji zwrotnych od swoich słuchaczy, chyba że ma telefon (delegata). Słuchacze mogą zignorować wiadomość lub coś z tym zrobić. NSNotification umożliwia wysyłanie wiadomości do dowolnego obiektu, ale bez połączenia między nimi w celu komunikacji w obie strony. Jeśli potrzebujesz tego typu komunikacji, prawdopodobnie powinieneś zaimplementować delegata; w przeciwnym razie NSNotification jest prostsze i łatwiejsze w użyciu, chociaż może przysporzyć Ci kłopotów.

Pytanie 83: Jak bezpiecznie przechowywać prywatne dane użytkownika offline na urządzeniu? Jakie inne najlepsze praktyki w zakresie bezpieczeństwa należy zastosować?

To pytanie powinno wywołać rozmowę z kandydatem na temat najlepszych praktyk i frameworków. Dobrze zorientowany kandydat będzie mógł mówić o pęku kluczy, czytniku dotykowym i 1 haśle.

Ponadto ogólne rozmowy na temat bezpieczeństwa sieci (używanie połączeń SSL, używanie Charlesa do debugowania itd.) Mogą prowadzić do interesujących miejsc.

Pytanie 84: Czy ataki typu SQL injection są prawidłowe w systemie iOS? Jak byś im zapobiegał?

Ataki typu SQL injection są bardzo prawdopodobne w systemie iOS. Istnieje kilka sposobów, aby temu zapobiec. Jednym z nich jest użycie zapytań parametrycznych.

Pytanie 85: Jakie są typowe przyczyny odrzucania aplikacji w App Store?

Szczególnie podoba mi się to pytanie, chociaż nie jest techniczne. Osoba, która od jakiegoś czasu jest na arenie programistycznej iOS, prawdopodobnie była świadkiem kilku odrzuceń. Na ich podstawie możesz wydobyć interesujące historie od kandydata, a także przeanalizować, w jaki sposób podchodzi do tych problemów i rozwiązuje je

Pytanie 86: Jak zabezpieczyć wątek fragmentu kodu?

Możesz użyć słowa kluczowego @synchronized. Spowoduje to automatyczne zabezpieczenie wątku. Istnieje bardzo interesujące podejście, które niewielu inżynierów może znać, a jeśli kandydat może o tym wspomnieć, zdecydowanie jest w swojej grze: uczynienie wszystkich obiektów niezmiennymi, aby nie można ich było modyfikować.

Pytanie 87: Dlaczego powinniśmy zwolnić punkty sprzedaży w viewDidUnload?

viewDidUnload jest ściśle używany do zwalniania IBOutletów z zachowaniem właściwości. Przyczyną tego jest fakt, że UIViewController ma właściwość widoku, którą zachowuje. Sama właściwość widoku zachowuje odniesienia do wszystkich swoich widoków podrzędnych. Te podglądy są dokładnie tym, co zachowujesz w tych właściwościach outletu. Problem polega na tym, że te podziały mają „dodatkowe" zachowanie. Celem -viewDidUnload jest usunięcie niepotrzebnego użycia pamięci. Gdy wywoływana jest opcja -viewDidUnload, właściwość widoku została już wydana, co zwalnia UIView najwyższego poziomu wraz ze wszystkimi jego widokami podrzędnymi. Ponieważ jednak zachowaliśmy niektóre z tych podglądów, pozostają one w pamięci i chcemy je udostępnić, ponieważ nie będą już używane. Nowe kopie tych podglądów podrzędnych zostaną utworzone, gdy (jeśli) widok zostanie ponownie załadowany. Właściwości są również ustawione na zero, ściśle, więc nie mamy wskaźników wskazujących na zwolnioną pamięć.

Pytanie 88: Jaka jest różnica między płytką kopią a głęboką kopią?

Płytkie kopie są powielane tak mało, jak to możliwe. Płytka kopia kolekcji jest kopią struktury kolekcji, a nie elementów. W przypadku płytkiej kopii dwie kolekcje współdzielą teraz poszczególne elementy. Głębokie kopie kopiują wszystko. Głęboka kopia kolekcji to dwie kolekcje zawierające wszystkie elementy z pierwotnego duplikatu kolekcji.

Pytanie 89: Jak można przekazać nieznany typ jako parametr?

Możesz użyć (id).

```
-(void) fooMethod:(id)unknownTypeParameter {

if( [unknownTypeParameter isKindOfClass:[Animal Class]]) {

Animal *referanceObj = (Animal *) unknownTypeParameter;

referanceObj.noOfLegs = 4;

}
```

}

Pytanie 90: Co to jest deinitializer i jak jest napisany w Swift?

Deinitializer jest deklarowany bezpośrednio przed cofnięciem alokacji instancji klasy. Piszesz deinitializer ze słowem kluczowym deinit.

Dezinicjalizator jest napisany bez nawiasów i nie zajmuje

dowolne parametry. Jest napisane w następujący sposób:

deinit {

// wykonaj deinicjalizację

}

Pytanie 91: Co to jest opcjonalne tworzenie łańcuchów?

Proces odpytywania i wywoływania właściwości, indeksów i metod opcjonalnego, który może być zerowy - jest definiowany jako opcjonalne tworzenie łańcuchów. Opcjonalne tworzenie łańcuchów zwraca dwie wartości:

• Jeśli opcja zawiera wartość, wywołanie powiązanej właściwości, metod i indeksów dolnych zwraca wartość.

• Jeśli opcja zawiera wartość nil, wszystkie powiązane właściwości, metody i indeksy dolne zwracają wartość nil.

Ponieważ zgrupowanych jest wiele zapytań do metod, właściwości i indeksów dolnych, awaria jednego łańcucha wpłynie na cały łańcuch i spowoduje uzyskanie wartości zerowej.

Pytanie 92: Co to jest stwierdzenie Fallthrough? Co to robi?

Fallthrough „przechodzi" do następnego przypadku, a nie do następnego pasującego przypadku. Koncepcja jest dziedziczona z instrukcji przełącznika języka C, w których każdy przypadek może być traktowany jako etykieta docelowa przejścia do celu, a instrukcja switch przenosi wykonanie do pierwszej zgodnej.

Pytanie 93: Co to są leniwie przechowywane właściwości i kiedy są przydatne?

Właściwości przechowywane z opóźnieniem są używane dla właściwości, których wartości początkowe nie są obliczane aż do pierwszego użycia. Można zadeklarować leniwą właściwość składowaną, wpisując leniwy modyfikator przed jego deklaracją. Leniwe właściwości są przydatne, gdy początkowa wartość właściwości zależy od czynników zewnętrznych, których wartości są nieznane.

Pytanie 94: Czy słyszałeś o Handoff?

Handoff to nowa funkcja wprowadzona w iOS 8 i OS X Yosemite. Funkcja Handoff umożliwia nieprzerwane kontynuowanie działania podczas przełączania się z jednego urządzenia na drugie, bez konieczności ponownej konfiguracji któregokolwiek z urządzeń.

Pytanie 95: Czy możesz mieć więcej UIWindows w iOS?

To rzadki przypadek, chociaż jest to możliwe. Na przykład dzieje się tak, gdy masz UIAlertView w osobnym oknie.

Pytanie 96: Co to jest metal?

Metal to niskopoziomowa, mniej przenośna, bardziej zoptymalizowana warstwa graficzna dla robienie tego samego rodzaju rzeczy, co w przypadku OpenGL (i OpenCL) lub Direct3D, czyli renderowanie grafiki 3D i równoległe programowanie GPU. Metal próbuje zmniejszyć ilość narzutów wymaganych ze względu na wydajność, a także zmniejszyć wąskie gardła procesora.

Pytanie 97: Czy potrafisz wymyślić strategie zwiększania wydajności sieci?

Doświadczony programista powinien znać niektóre techniki, które nie są domyślnie dostępne w systemie iOS. To poprawi twoją grę. Możesz omówić takie tematy, jak pomiar opóźnienia (ocena typu sieci, do której jesteś podłączony, aby podjąć decyzję o transmisji danych), grupowanie (pakowanie razem żądań i wysyłanie ich razem, gdy są reprezentatywną grupą - coś, co zostało zrobione w wielu analitycznych SDK), wstępne pobieranie (pobieranie informacji, gdy urządzenie jest bezczynne lub połączenie internetowe jest silne), wykładnicze wyłączone (jeśli połączenie nie powiedzie się, poczekaj na wykładniczo rosnące interwały przed wykonaniem następnego żądania), mechanizmy buforowania, użycie nagłówków Last-Modified…

Pytanie 98: Jaki jest najbardziej złożony problem, który musiałeś rozwiązać w swojej poprzedniej pracy?

To pytanie ujawni umiejętności kandydata, które nie są bezpośrednio związane z wiedzą na temat iOS, ale są równie ważne. Może to skierować Cię do omówienia problemów związanych z rozwojem aplikacji, eskalacją systemu, sposobem zrównoważonego rozwoju oprogramowania w dłuższej perspektywie, jak rozwijać zespoły i sprawić, by pracowały wydajnie, a także frameworków, których kandydat użył do rozwiązywania problemów.

Pytanie 99: Co to jest pula automatycznej sprzedaży?

Klasa NSAutoreleasePool jest używana do obsługi systemu zarządzania pamięcią liczoną jako odwołania Cocoa. Pula autowyrealniania przechowuje obiekty, do których jest wysyłany komunikat o zwolnieniu, gdy pula jest opróżniana. Ponadto w przypadku korzystania z automatycznego liczenia odwołań nie można bezpośrednio korzystać z pul automatycznego udostępniania. Zamiast tego używasz bloków @autoreleasepool.

Pytanie 100: Jaka jest hierarchia klas od UIButton do NSObject?

UIButton dziedziczy po UIControl, UIControl dziedziczy po UIView, UIView dziedziczy po UIResponder, UIResponder dziedziczy z klasy głównej NSObject: UIButton ➤ UIControl ➤ UIView ➤ UIResponder ➤ NSObject.