Введение в шаблоны Cocoa

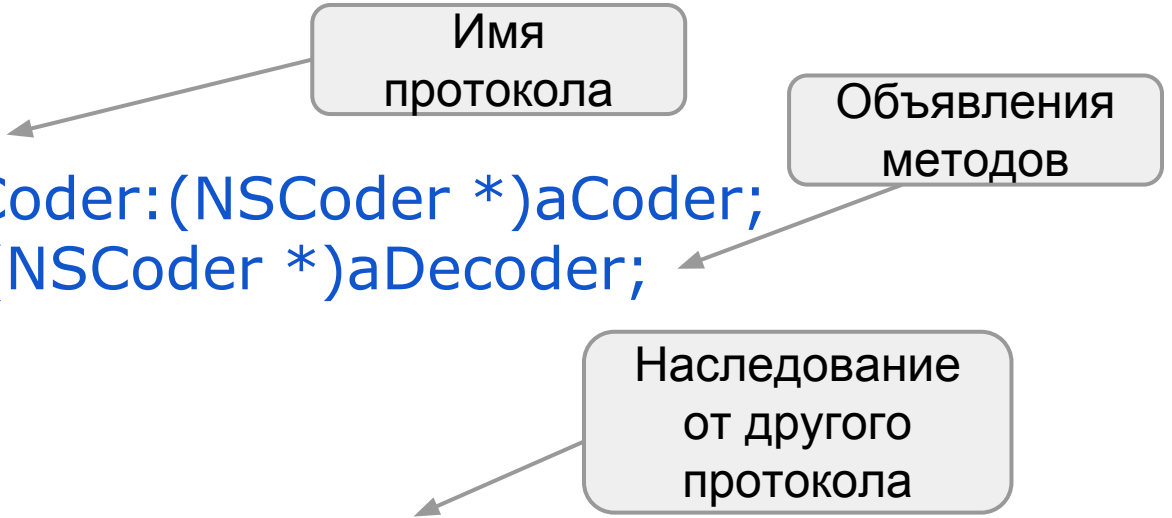
Протокол. Делегирование. UITableView

@protocol

- Проблема множественного наследования (одно из решений).
- Реализует ООП-концепцию “интерфейс” (не путать с @interface).
- Имеет только объявления методов.
- “Подчиниться” протоколу (conform to protocol) - взять на себя реализацию этих методов.
- Протоколам подчиняются другие классы.

Пример 1.

```
@protocol NSCoding
- (void)encodeWithCoder:(NSCoder *)aCoder;
- (id)initWithCoder:(NSCoder *)aDecoder;
@end
```



Пример 2.

```
@protocol NSSecureCoding <NSCoding>
@required
+ (BOOL)supportsSecureCoding;
@end
```

Объявление протокола

```
@interface UIViewController : UIResponder <NSCoding>
@property NSString *title;
@end

@implementation UIViewController

- (id)initWithCoder:(NSCoder *)aDecoder {
   ....
   self.title = [aDecoder decodeObjectForKey:@”title”];
   ....
   return self;
}
@end
```

Наследование от класса

Подчинение протоколу

Реализация одного из методов протокола

Подчинение протоколу

Еще о протоколах

- Множественное наследование протоколов друг от друга

```
@protocol CustomDataProtocol <DataProtocol, CryptedProtocol>
```

- Множественное подчинение разным протоколам у класса

```
@interface NSDictionary : NSObject <NSCopying, NSMutableCopying, NSSecureCoding, NSFastEnumeration>
```

Еще о протоколах (продолжение)

- Ограниченный динамизм через указание подчинения протоколу/протоколам:

```
id<NSCoding, NSCopying> codableVariable;
UIView<CustomDataProtocol> *customizableView;
```

- Проверка объекта любого типа на подчинение протоколу:

```
if ([var conformsToProtocol:@protocol(NSLocking)]) {
   [var lock];
}
```

Делегирование

- Шаблон ООП. Использование объекта-делегата для реализации той или иной функциональности.
- Делегат может быть независимым объектом (агрегация) или существовать только вместе с его владельцем (композиция).

```
@protocol SessionUpdaterDelegate
@required
- (NSString *)newSession;
@end

@interface RequestHandler : NSObject
@property NSString *session;
@property id<SessionUpdaterDelegate> sessionDelegate; // Объект-делегат
- (NSArray *)downloadPictures;
@end

@implementation RequestHandler
- (NSArray *)downloadPictures {
   BOOL sessionOK = …
   if (sessionOK == NO)
      self.session = [self.sessionDelegate newSession]; // Обращение к делегату
}
@end
```

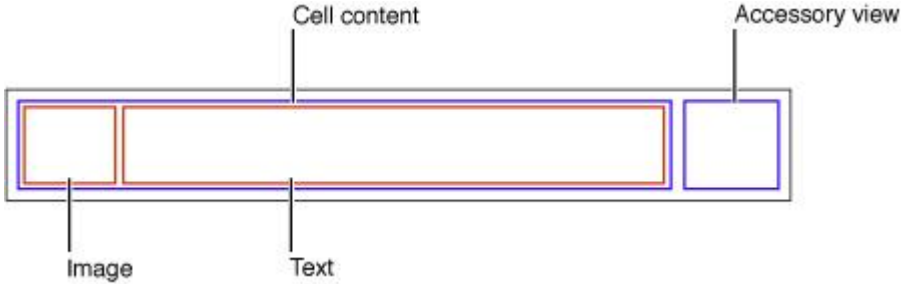
Делегирование с помощью протоколов

UITableView

- Класс в Cocoa, отображающий однотипный упорядоченный контент (таблица).
- Непрямой наследник UIView (работают все методы и свойства UIView).
- Содержимое ячеек заполняется делегатами с помощью протоколов UITableViewDelegate и UITableViewDataSource.

UITableViewCell

- Класс-наследник UIView, ячейка таблицы.
- Имеет несколько стандартных стилей.
- Имеет свойство @property UIView *contentView. Это вью, на который можно добавлять свои вью.
- Для создания кастомных ячеек обычно используют наследование.

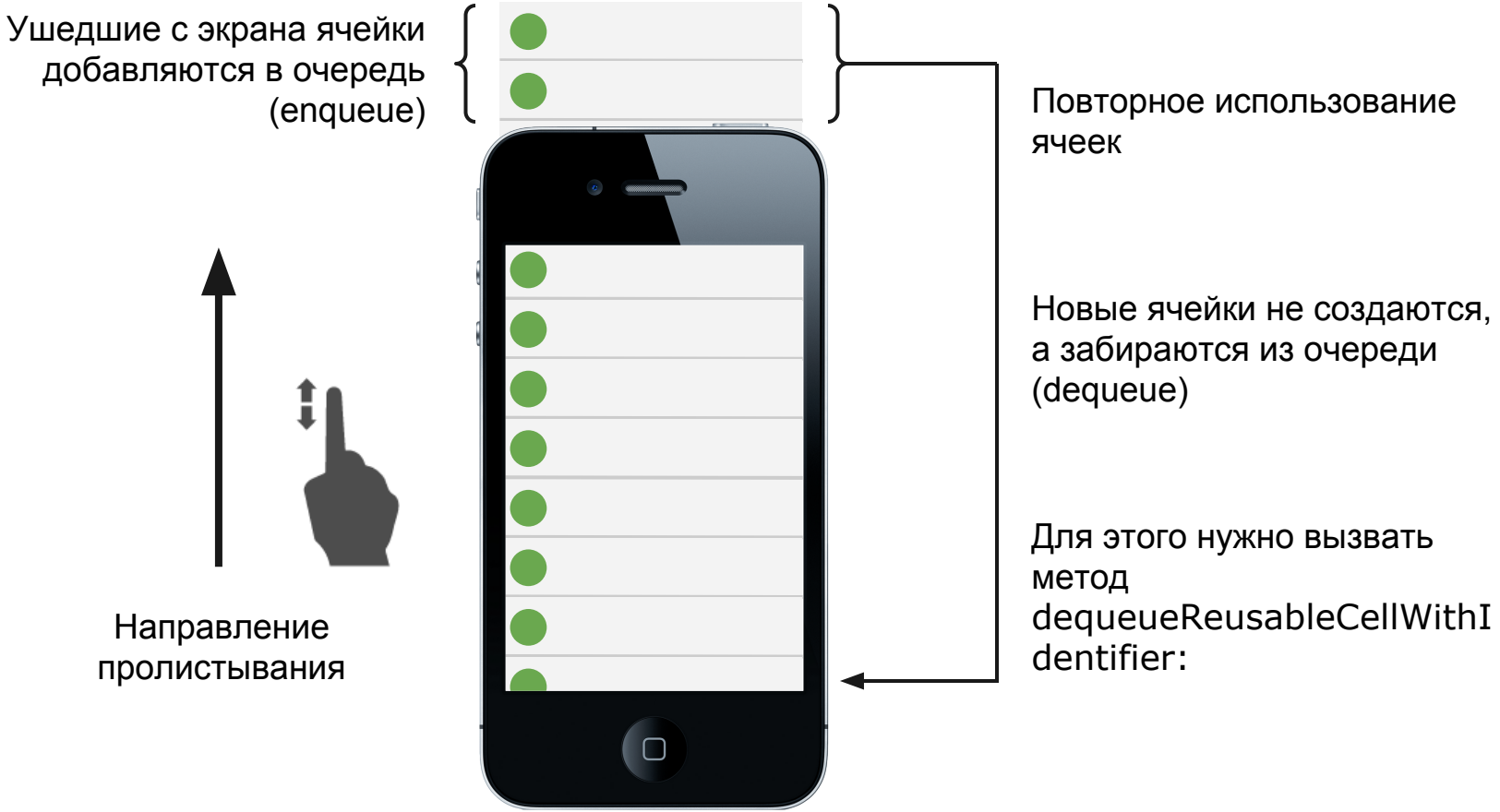


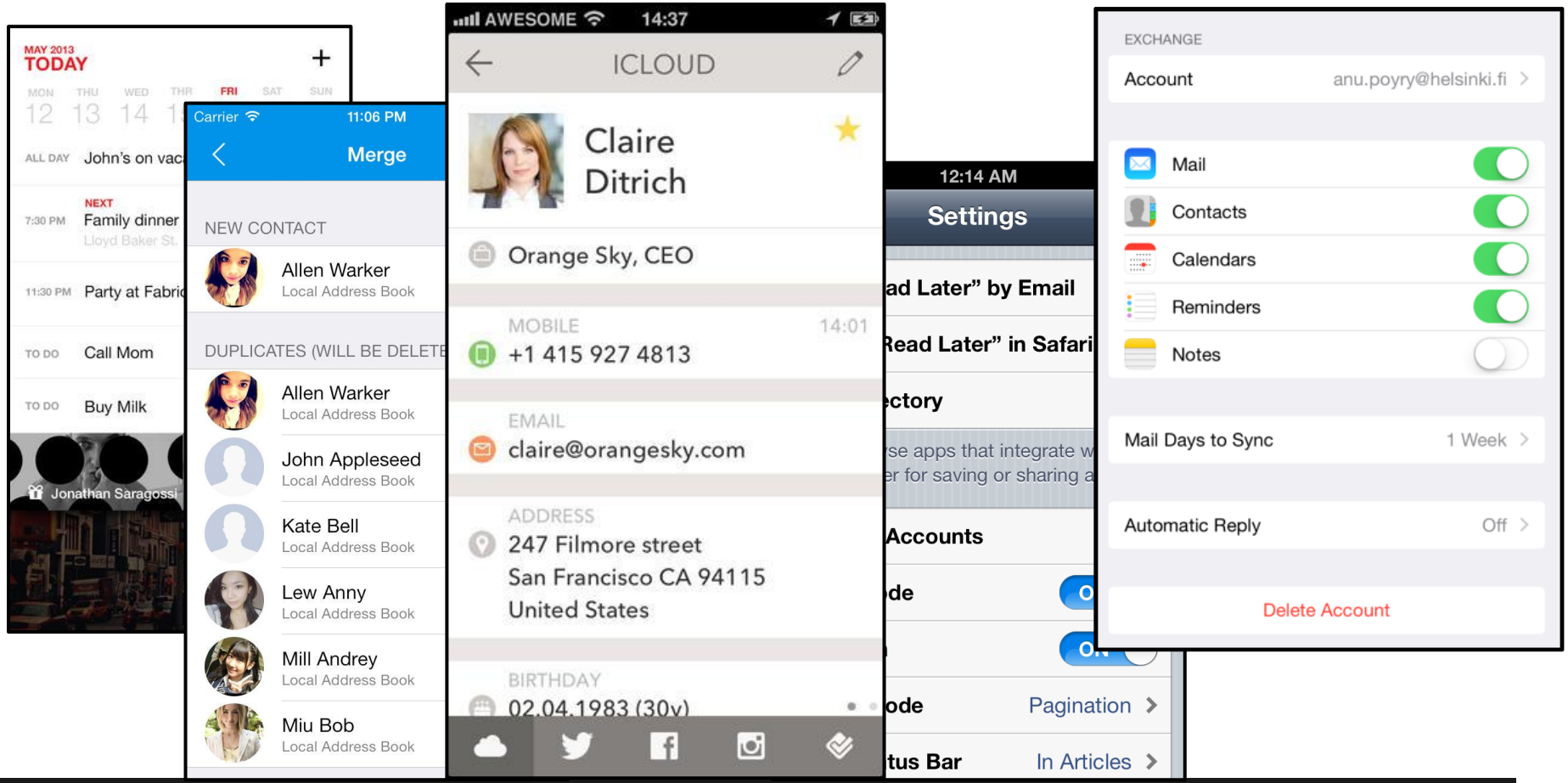
Отображение
UITableViewCell

Стандартные стили
UITableViewCell

Внешний вид UITableViewCell

Жизненный цикл ячейки с reuseIdentifier





Примеры интерфейсов с UITableView