Разработка приложений для iOS

Лекция 2

# Objective-C

*Глеб Тарасов*

gleb34@gmail.com

# Похож на C, C++, Java, C#

```
int sum = 0;
for (int i = 0; i < 10; i++)
{
    if (i > 5)
    {
        sum = sum + i;
    }
}
```

# Типы данных

## числа

```
BOOL a = YES;
NSInteger b = -1;
```

## объекты

```
NSString *c = @"string";
NSArray *a = @[ ];
Person *p = ...;
```

# BOOL

```
BOOL a = YES;
if (a) // if (a == YES)
{
    BOOL b = NO;
    if (!b) // if (a == NO)
    {
        //blabla
    }
}
```

# Числа

```
CGFloat a = 0.5;

NSInteger b = -1;

NSInteger c = b + 20;
```

# Объекты

```
NSString *a = @"string";

NSArray *b = @[ @"a", @"b" ];

NSDictionary *c = @[ @"a" : @"1",
                     @"b" : @"2" ];

Person *p = nil;
```

# Объекты

```
(NSString*)a = ...;

(NSString *)a = ...;

NSString *a = ...;

NSString a = ...;
```

# Вызов методов

```
NSString *a = @"string";
```

```
NSString *b = [a copy];
```

```
[b stringByReplacingOccurrencesOfString:@" " withString:@""];
```

```
[c stringByPaddingToLength:20
                withString:@" "
           startingAtIndex:0];
```

# Статические методы

```
NSString *s = [Utils removeSpaces:q];

Person *p = [Person createPerson];
```

# Создание объекта

```
NSString *c = [[NSString alloc] init];
```

```
NSArray *a = [[NSArray alloc] init];
```

```
NSString *c2 = [[NSString alloc]
                initWithString:@"string"];
```

# nil

```
Person *p = nil;
```

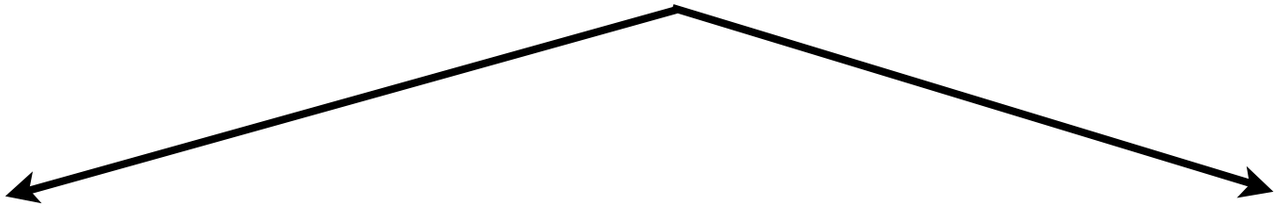
- аналог null из других языков
- можно вызывать любой метод, не будет исключения
- если метод возвращает объект - вернется nil
- если метод возвращает число - вернется 0

# nil

```
NSString *a = nil;
int length = [a length]; // 0
NSString *m = [a mutableCopy]; // nil
if (a) // if (a != nil)
{
    NSLog(@"test");
}
```

# Стандартные классы

# Строки

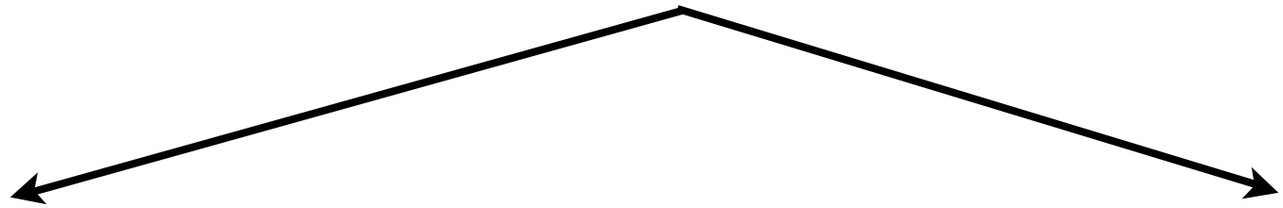


NSString

NSMutableString

```
NSString *a = @"abc";
NSString *b = [a stringByReplacingOccurrencesOfString:@"a"
                                           withString:@"b"];
NSLog(@"b: %@", b);

NSMutableString *m = [b mutableCopy];
NSRange r;
r.length = m.length;
r.location = 0;
[m replaceOccurrencesOfString:@"c"
                   withString:@"b"
                      options:0
                        range:r];
NSLog(@"m: %@", m);
```

# Списки

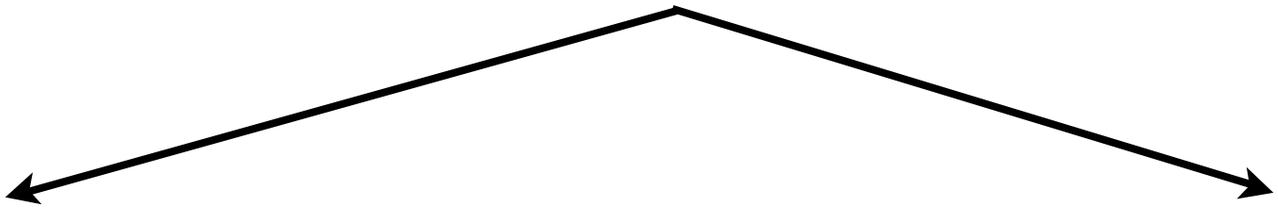


`NSArray` `NSMutableArray`

Обычные упорядоченные массивы

```
NSArray *a = @[@"a", @"b", @"c"];
NSString *first = a[0];
NSString *last = a[[a count] - 1];

NSMutableArray *b = [a mutableCopy];
[b addObject:@"r"];
[b replaceObjectAtIndex:1 withObject:@"q"];
[b removeObjectAtIndex:2];
```

# Словари



`NSDictionary` `NSMutableDictionary`

Хранение пар «ключ-значение».
Быстрый поиск значения по ключу

```
NSDictionary *dict = @{ @"key1" : @"a", @"key2" : @"b" };
NSString *first = dict[@"key1"];

NSMutableDictionary *m = [dict mutableCopy];
m[@"key3"] = @"c";
m[@"key1"] = @"aa";
NSLog(@"m: %@", m);
```

# Перечисление списков

```
NSArray *arr = @[ @"a", @"b", @"c" ];

for (NSString *a in arr)
{
    NSLog(@"%@", a);
}

NSDictionary *dict = @{ @"key1" : @"a", @"key2" : @"b" };

for (NSString *key in dict)
{
    NSString *value = dict[key];
    NSLog(@"%@ - %@", key, value);
}
```

# Добавлять в коллекции можно только объекты!

```
NSArray *arr = @[ 1, 4, 5 ];
NSDictionary *dict = @{ @"key1" : 2, @"key2" : 3 };
```

# Числа

## NSNumber

```
NSNumber *a = @(3);
NSInteger b = [a integerValue];

NSNumber *c = @(2.5);
CGFloat d = [c floatValue];

if ([a isEqualToNumber:c])
    NSLog(@"equals");


  NSArray *arr = @[ @(1), @(4), @(5) ];
```

# NSObject

все объекты наследуются от NSObject

```
NSObject *q = [[NSObject alloc] init];
[q copy];
[q mutableCopy];
NSLog(@"%@", [q description]);
BOOL eq = [q isEqual:@(1)];
```

# Тип id

```
15      NSNumber *a = @(1);
16      NSObject *b = a;
        [b integerValue];
```

```
15      NSNumber *a = @(1);
16      id b = a;
17      [b integerValue];
```

# Селекторы

```
NSString *a = @"a"
SEL sel = @selector(isEqualToString:);
id b = a;
if ([b respondsToSelector:sel])
{
   BOOL e = [b isEqualToString:@"a"];
}
```

# Тип id

```objc
NSObject *o = [[NSObject alloc] init];
NSArray *arr = @[ @"str", @(1), o ];

for (id obj in arr)
{
    if ([obj respondsToSelector:@selector(integerValue)])
    {
        NSLog(@"%d", [obj integerValue]);
    }
}
```

# Классы

```
Class cls = [NSString class];


NSString *a = @"a";
id b = a;
if ([b isKindOfClass:[NSString class]])
{
   BOOL e = [b isEqualToString:@"a"];
}
```

# Собственные классы

заголовочный
файл (.h)

файл реализации
(.m)

# Собственные классы

```objc
@interface User : NSObject

@property NSString *name;
@property NSInteger age;

- (BOOL)canBuyDrinks;


@end
```

User.h

# Собственные классы

```objc
@implementation User

- (BOOL)canBuyDrinks
{
    return self.age > 18;
}

@end
```

User.m

# User.h

```
@interface User : NSObject

@property NSString *name;
@property NSInteger age;

- (void)deleteProfile;

- (void)postCommentWithText:(NSString *)text;

- (void)postCommentWithTopic:(NSString *)topic
                     andText:(NSString *)text;


@end
```

# User.m

```
@implementation User

- (void)deleteProfile
{
    NSLog(@"Пользователь %@ удален", self.name);
}

- (void)postCommentWithText:(NSString *)text
{
    [self postCommentWithTopic:@"" andText:text];
}

- (void)postCommentWithTopic:(NSString *)topic andText:
    (NSString *)text
{
    NSLog(@"Пользователь %@ (возраст: %d) с темой %@",
        self.name, self.age, topic);
}

@end
```

# Admin.h

```
#import "User.h"

@interface Admin : User

- (void)deleteComment:(NSInteger)key;

@end
```

# Admin.m

```
#import "Admin.h"

@implementation Admin

- (void)deleteComment:(NSInteger)key
{
    //удаляем из базы
    NSLog(@"Комментарий с ключом %d удален", key);

    // оставляем комментарий, об удалении
    [self postCommentWithTopic:@"От админа"
                       andText:@"Удалил коммент за хамство"];
}

@end
```

# Переопределение методов

```objc
@implementation User

- (NSString *)description
{
    return self.name;
}

@end
```

# Использование объектов

```
User *user = [[User alloc] init];

user.name = @"UserName";


NSString *comment = @"БлаБлаБла";

[user postCommentWithText:comment];

[user deleteProfile];
```

# Собственный инициализатор

```
- (id)init
{
    self = [super init];
    if (self)
    {
        self.name = @"name";
    }
    return self;
}

- (id)initWithName:(NSString *)name
{
    self = [super init];
    if (self)
    {
        self.name = name;
    }
    return self;
}
```

# Методы класса

В файле Admin.h:

```
+ (Admin *)createAdmin;
```

В файле Admin.m:

```
+ (Admin *)createAdmin
{
    Admin *admin = [[Admin alloc] initWithName:@"Админ Админович"];
    admin.age = 34;
    return admin;
}
```

Использование:

```
Admin *admin = [Admin createAdmin];
[admin deleteComment:10];
```

# Протоколы

```
@protocol SendMessageProtocol

- (void)sendMessage:(NSString *)message;

@end

@interface User : NSObject<SendMessageProtocol>

@end

@implementation User

- (void)sendMessage:(NSString *)message
{
    // send message
}

@end


id<SendMessageProtocol> sender = [[User alloc] init];
[sender sendMessage:@"message"];
```

# Категории

NSString+Spaces.h

```
@interface NSString(Spaces)

- (NSString *)stringWithoutSpaces;

@end
```

NSString+Spaces.m

```
@implementation NSString(Spaces)

- (NSString *)stringWithoutSpaces
{
  return [self stringByReplacingOccurrencesOfString:@" "
                                          withString:@""];
}

@end
```

```
  NSString *s1 = @"a b c";
  NSString *s2 = [s1 stringWithoutSpaces];
```

# Приватные свойства

User.m

```
#import "User.h"

@interface User()

@property NSString *name;

@end

@implementation User

...

@end
```

# Еще про свойства

```objc
@interface User : NSObject

@property NSInteger age;

@end
```

```objc
@interface User : NSObject
{
    NSInteger _age;
}

- (NSInteger)age;
- (void)setAge:(NSInteger)age;

@end
```

```objc
@implementation User

- (void)setAge:(NSInteger)age
{
    _age = age;
}

- (NSInteger)age
{
    return _age;
}

@end
```

# Dot notation

```
User *u = [[User alloc] init];
u.name = @"123";
[u setName:@"123"];

NSString *s1 = u.name;
NSString *s2 = [u name];
```

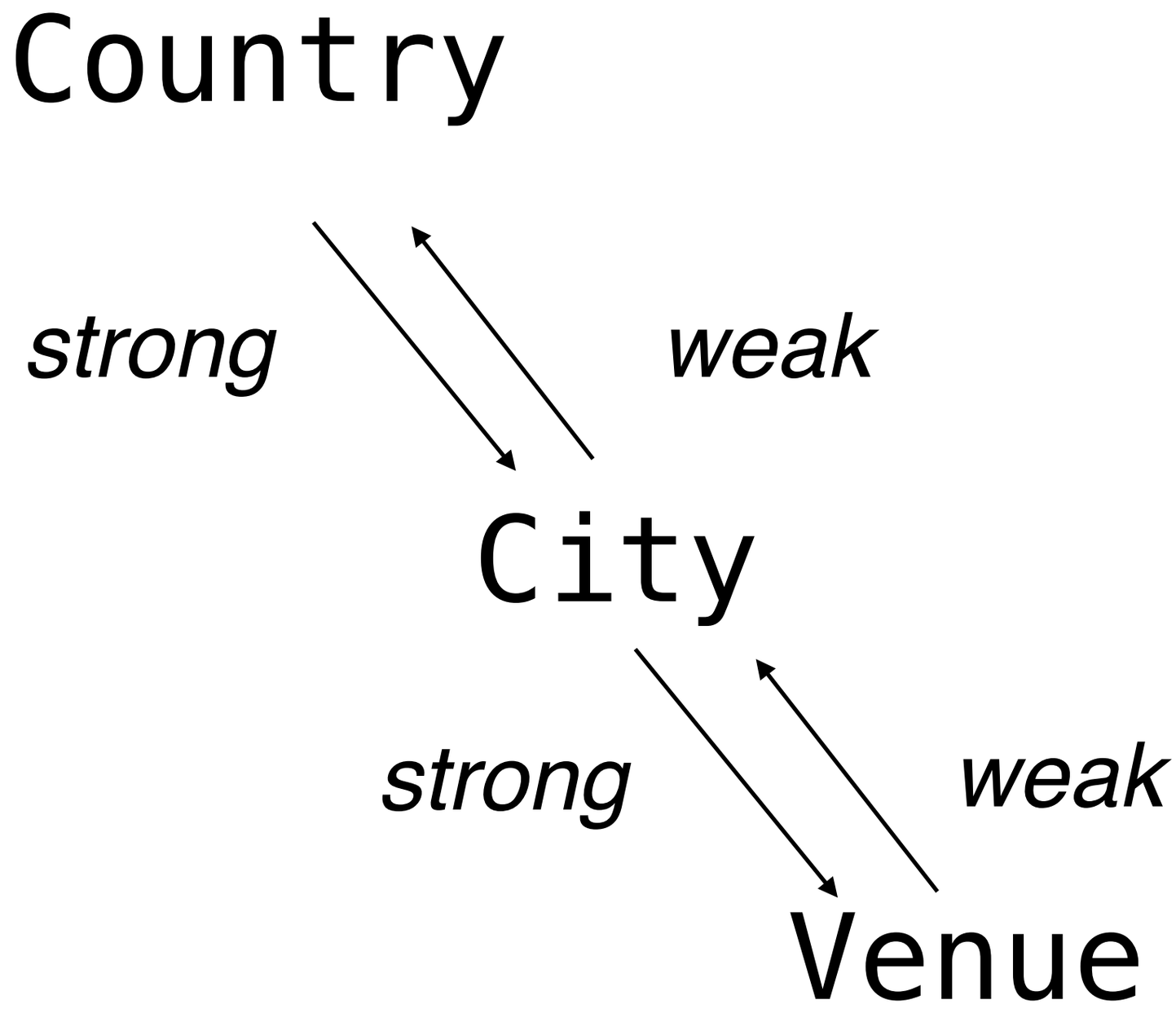
Переопределить геттер или сеттер:

```
- (void)setName:(NSString *)name
{
    _name = name;
    NSLog(@"%@", name);
}
```

# Модификаторы

- strong
- weak

```
@interface Car : NSObject

@property(strong, nonatomic) NSArray *wheels;

@end

@interface Wheel : NSObject

@property(weak, nonatomic) Car *car;

@end
```


Country
strong
weak
City
strong
weak
Venue

# Objective-C style guide

- Локальные переменные: *myLocalVariable*
- Свойства: *myProperty*
- Классы: *MyClass*
- Методы: *doSomethingWith:*

# Демонстрация

# Домашнее задание

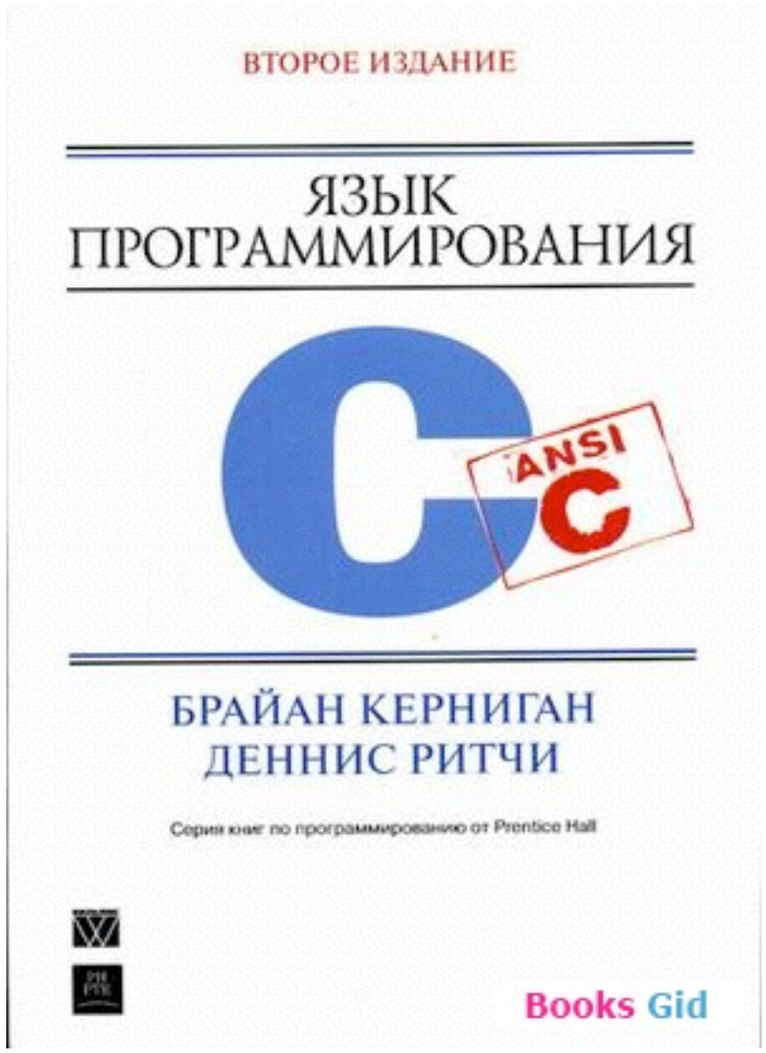
- продумать, какие классы будут нужны в модели вашего приложения
- реализовать их (продумать свойства, прикинуть методы)
- создать проект *Empty Application*
- В методе *didFinishLaunching* в *AppDelegate* создать эти классы, заполнить, распечатать в консоль

# Всё!

*Глеб Тарасов*
*gleb34@gmail.com*
*twitter.com/pilot34*

если успеем

# Немного Си




http://ru.wikipedia.org/wiki/Си_(язык_программирования)

# Функции

```
double add(double a, double b)
{
    return a + b;
}
```

# Структуры

```
CGPoint p;
p.x = 10;
p.y = 20;
p = CGPointMake(10, 20);

CGSize s;
s.width = 100;
s.height = 100;
s = CGSizeMake(100, 100);

CGRect r;
r.size = s;
r.origin = p;
r = CGRectMake(10, 20, 100, 100);
```

# Структуры

```
typedef struct
{
    float x;
    float y;
} Location;

Location createLocation(float x, float y)
{
    Location l;
    l.x = x;
    l.y = y;
    return l;
}

int main()
{
    Location l = createLocation(1.5, 0.5);
    printf("location: {%g, %g}", l.x, l.y);
}
```

# Перечисления

```
typedef enum
{
    UIViewAnimationCurveEaseInOut,
    UIViewAnimationCurveEaseIn,
    UIViewAnimationCurveEaseOut,
    UIViewAnimationCurveLinear
} UIViewAnimationCurve;
```

```
UIViewAnimationCurve a = UIViewAnimationCurveLinear;
printf("%d", a);
```