Блоки и многопоточность ч.2

GCD, потоки, объекты синхронизации

NSThread

- Класс, который позволяет выполнить метод на определенном потоке.
- Имеет три состояния: **executing** (выполняется), **finished** (окончен), **cancelled** (отменен).

```
- (void)longOperation {
    for (unsigned long i = 0; i < 1000 * 1000 * 1000; i++) {
        if (i % 10 == 0) {
            NSLog(@"%lu", i);
            if ([NSThread currentThread].cancelled) // Отменен ли поток?
                break;
        }
    }
}
…… // Создаем объект потока в другом методе с привязкой к методу longOperation
NSThread *loopThread = [[NSThread alloc] initWithTarget:self selector:@selector(longOperation) object:nil];
loopThread.threadPriority = 0.2; // Низкий приоритет потока (между 0.0 и 1.0)
[loopThread start]; // Запуск потока
// Проверим состояние потока через 10 секунд
dispatch_after(dispatch_time(DISPATCH_TIME_NOW, (int64_t)(10.0 * NSEC_PER_SEC)),
dispatch_get_main_queue(), ^{
        if (loopThread.executing) // Если все еще выполняется
            [loopThread cancel]; // отменяем выполнение
    });
```

Пример работы с NSThread

Объекты синхронизации

- Применяются, если разные потоки пытаются изменить один и тот же ресурс (например, свойство в классе или файл на диске).
- Часто используется **мьютекс** - двоичный семафор (имеет состояния открыт/закрыт).

NSLock

- Класс Cocoa, реализующий мьютекс (по принципу замка́).
- Если один поток успел закрыть мьютекс, второй будет ждать до тех пор, пока первый не откроет его.
- Каждой операции закрытия *обязана* быть сопоставлена операция открытия. Причем из того же потока.

```
- (void)testLock { // Метод, к которому обращаются из разных потоков
    [_arrayLock lock];
    [_mutableArray insertObject:@(arc4random_uniform(1000)) atIndex:0];
    [_arrayLock unlock];
}
..... // В теле какого-нибудь другого метода
    _mutableArray = [[NSMutableArray alloc] init];
    _arrayLock = [[NSLock alloc] init]; // Создание мьютекса

    dispatch_async(dispatch_queue_create("my_queue_0", DISPATCH_QUEUE_CONCURRENT), ^{
        while (YES) { // Запускаем один поток
            [self testLock];
        }
    });
    dispatch_async(dispatch_queue_create("my_queue_1", DISPATCH_QUEUE_CONCURRENT), ^{
        while (YES) { // Запускаем второй поток
            [self testLock];
        }
    });
```

Пример использования NSLock

@synchronized()

- Директива в Objective-C, являющаяся упрощенным аналогом мьютекса.
- В качестве параметра передается указатель на объект, вокруг которого должна строится синхронизация.

```
@synchronized(_mutableArray) {
  [_mutableArray insertObject:@(arc4random_uniform(1000))
atIndex:0];
}
```

atomic / nonatomic

- Спецификаторы свойства в ObjC, указывающие на атомарность доступа к содержимому свойства.
- **atomic** - по умолчанию. Потокобезопасный.
- **nonatomic** - Быстрее, чем atomic, но потоконебезопасный.

```
@property (atomic) NSMutableArray *atomicArray;
@property (nonatomic) NSMutableArray *nonatomicArray;

- (NSMutableArray *)atomicArray {
    @synchronized(_atomicArray) {
        return _atomicArray;
    }
}

- (NSMutableArray *)nonatomicArray {
    return _nonatomicArray;
}
```

Смысл atomic/nonatomic