**Урок 8**

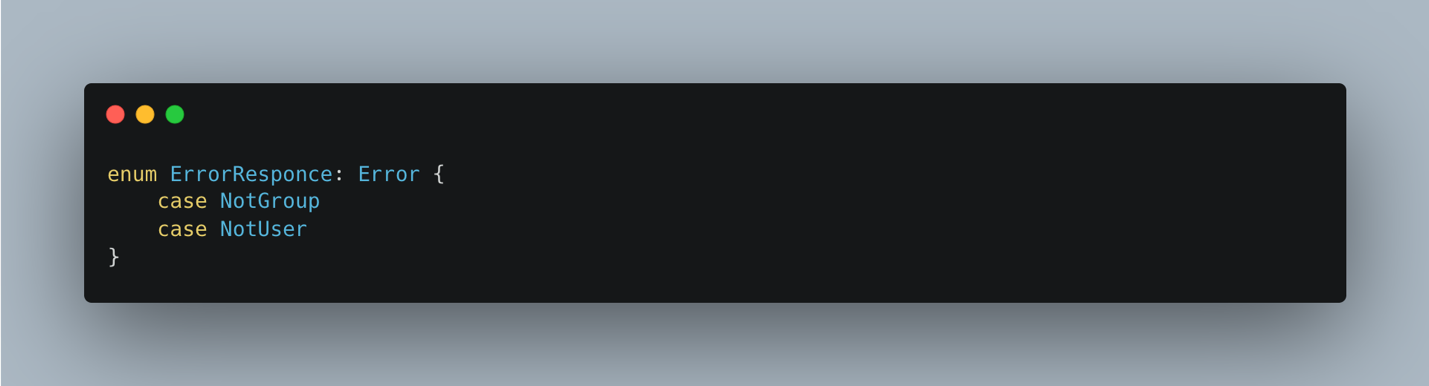
* Обработка ошибок.
* Приведение типов

**Обработка ошибок**

В swift существуют функции, которые не гарантируют, что вернется ответ, но может вернуться ошибка и мы должны быть готовы к тому, что такое произойдет. Такие функции называются throws и очень часто используются при разработке. И раз мы знаем, что может вернуться ошибка, то соответственно мы ее должны обработать, чтобы программа не вылетела.

Существует специальный блок do – catch, который ожидает, что может вернуться ошибка, и в котором мы прописываем необходимые процедуры, на случай такого сценария, посмотри на практике.

Для начала создадим перечисление возможных ошибок



Затем саму функцию и скажем, что она throw. Функция будет принимать номер группы и id пользователя, имитировать работу сервера, который не всегда гарантирует получение ответа, если данные не совпадут, то выведем одну из соотствующих ошибок, если все данные верны, то сделаем обычный принт.

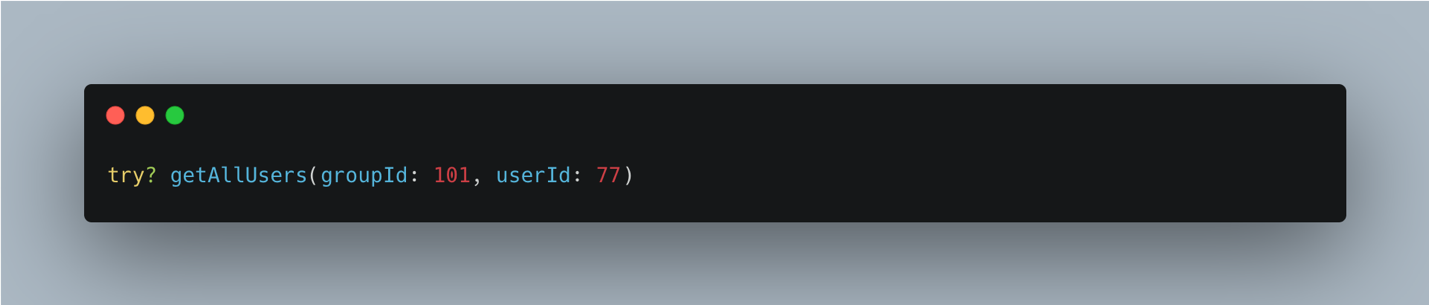


Теперь надо вызвать такую функцию, если мы как обычно начнем ее вызывать, то xcode будет ругаться и выдавать ошибку. Для вызова таких функций мы должны использовать ключевое слово try, т.е. мы должны попытаться вызвать функцию и ожидать, что может функция может вернуть ошибку. Для этого как раз будем использовать блок do – catch.



В блоке do мы через try пытаемся вызвать функцию, если они не вернут одну из ошибок, программа продолжить выполняться дальше, в противном случае сработает один из catch.

Так же существует упрощенная запись, когда нам не нужно обрабатывать ошибки, а в случае, когда все же будет ошибка, просто вернуть nil, в таком случае мы можем написать следящий код.



**Приведение типов**

Еще одной важной возможностью языка является приведение типов. На практике это часто используется, когда несколько классов объединены общим родительским классом, и мы работаем с ним, но бывает нужно обратиться к самому дочернему классу и вызвать нужные методы, разберем на примере.

Допустим у нас есть родительский класс Auto и несколько дочерних.



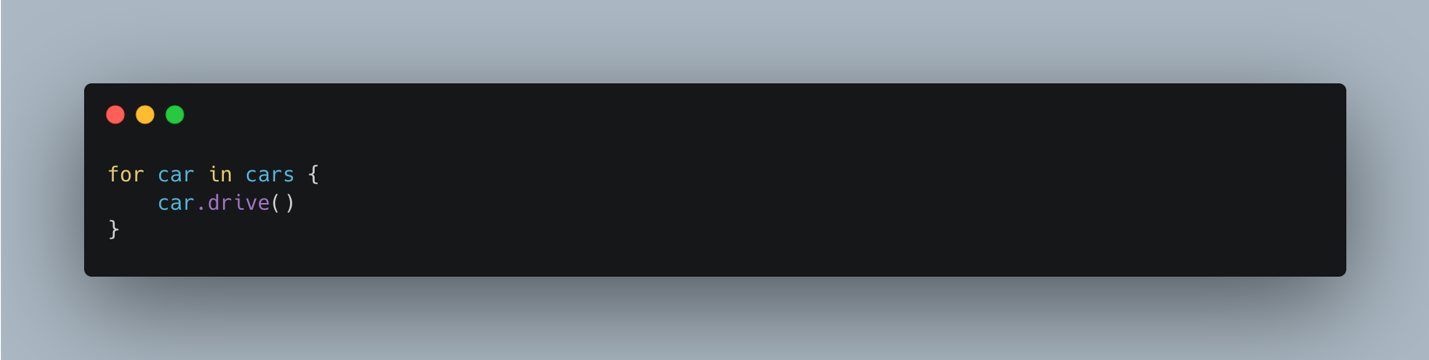
Каждому дочернему классу добавим еще свои методы и свойства.



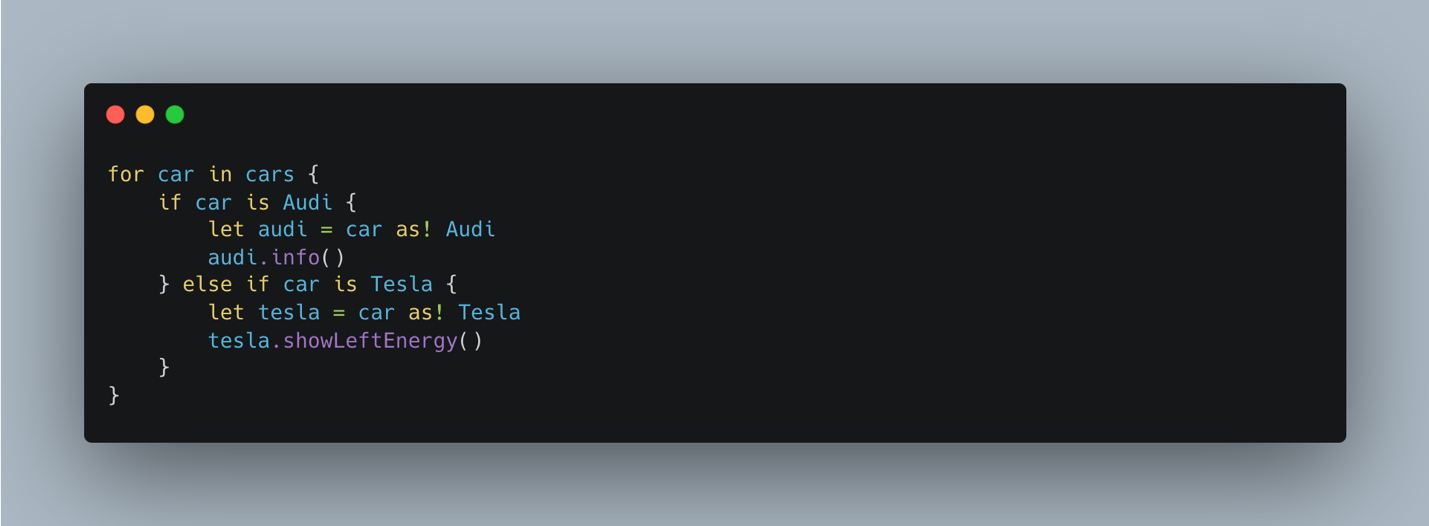
Теперь мы можем создать несколько экземпляров и собрать их в один массив, так как у всех один общий родительский класс. Соответственно массив будет типа [Auto].



А теперь попробуем перебрать элементы массива и вызвать нужные нам методы.



При вызове увидим, что нам доступен только один метод, который был определен именно в родительском классе. Как раз тут нам поможет приведение типов. Мы можем проверять экземпляры на соответствие типов. Для проверки принадлежности типа есть ключевое слово is а для приведения в нужный тип as.



Здесь мы проверяем, соответствует ли элемент массива определенному классу, если да, то при помощи as! говорим, что это экземпляр соответствующего типа. Теперь можно обращаться к его свойствам и методам.

Так же можно проверить соответствие через option binding или switch.



ДЗ.

1. Напишите функцию, которая может вернуть ошибку. Обработайте возможные ошибки.
2. На подобие примера из методички, составьте родительский и дочерние классы и сделайте соответствующие приведения типов