Swift предлагает два специальных псевдонима, позволяющих работать с неопределенными типами:

* AnyObject соответствует произвольному экземпляру любого класса;
* Any соответствует произвольному типу данных.

Данные псевдонимы позволяют корректно обрабатывать ситуации, когда конкретное наименование типа или класса неизвестно либо набор возможных типов может быть разнородным.

Псевдоним AnyObject позволяет указать на то, что в данном месте должен или может находиться экземпляр любого класса.

Инициализаторы класса и структуры, производящие установку значений свойств, называются назначенными (designated).

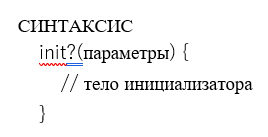
При этом должен существовать хотя бы один назначенный инициализатор, производящий установку значений всех свойств (если они существуют)

Convienent init

Помимо назначенных, в Swift существуют вспомогательные (convenience) инициализаторы. Они являются вторичными и поддерживающими. Вы можете определить вспомогательный инициализатор для проведения настроек и обязательного вызова одного из назначенных инициализаторов. Вспомогательные инициализаторы не являются обязательными для их реализации в типе.

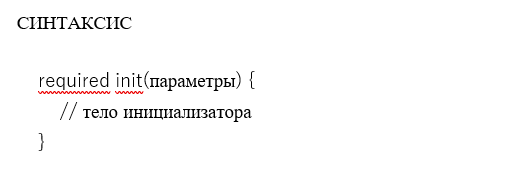
Failable init

Для этой цели служат проваливающиеся (failable) инициализаторы. Они способны возвращать nil при попытке создания экземпляра.



ВНИМАНИЕ. В классах проваливающийся инициализатор может вернуть nil только после *установки значений всех хранимых свойств*. В случае структур данное ограничение отсутствует.

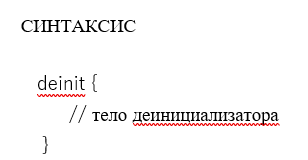
Обязательный (required) инициализатор — это инициализатор, который обязательно должен быть определен во всех подклассах данного класса.



Кроме того, модификатор required необходимо указывать перед каждой реализацией данного инициализатора в подклассах, чтобы последующие подклассы также реализовывали этот инициализатор.

**Деинициализаторы**

Деинициализаторы являются отличительной особенностью *классов*. Деинициализатор автоматически вызывается во время уничтожения экземпляра класса. Вы не можете вызвать деинициализатор самостоятельно. Один класс может иметь максимум один деинициализатор. С помощью деинициализатора вы можете, например, освободить используемые экземпляром ресурсы, вывести на консоль журнал или выполнить любые другие действия.



Деинициализатор суперкласса *наследуется подклассом* и вызывается автоматически в конце работы деинициализаторов подклассов. Деинициализатор суперкласса вызывается всегда, даже если деинициализатор подкласса отсутствует. Кроме того, экземпляр класса не удаляется, пока не закончит работу деинициализатор, поэтому все значения свойств экземпляра остаются доступными в теле деинициализатора.

ПРИМЕЧАНИЕ. Экземпляр существует до тех пор, пока на него указывает хотя бы одна ссылка.

Cоздано три экземпляра, указаны перекрестные ссылки друг на друга, но эти экземпляры вовремя не удаляются. Экземпляры остаются неудаленными,

Все создаваемые ссылки на экземпляр по умолчанию являются сильными. И когда два объекта указывают друг на друга сильными ссылками, то Swift не может принять решение о том, какую из ссылок можно удалить первой. Для решения проблемы некоторые ссылки можно преобразовать в слабые.

Слабые ссылки определяются с помощью ключевых слов weak и unowned. Модификатор weak указывает на то, что хранящаяся в параметре ссылка может быть в автоматическом режиме заменена на nil. Поэтому модификатор weak доступен только для опционалов. Но помимо опционалов бывают типы данных, которые обязывают переменную хранить значение (все неопциональные типы данных). Для создания слабых ссылок на неопционалы служит модификатор unowned.

ARC (Automatic Reference Counting — автоматический подсчет ссылок)

Сильные ссылки могут также стать источником проблем при их передаче в качестве входных параметров в замыкани. Захватываемые замыканиями экземпляры классов передаются по сильной ссылке и не освобождаются, когда замыкание уже не используется.