# iOS alapú szoftverfejlesztés - Labor 03

A laborsegédletet összeállította: Kelényi Imre - imre.kelenyi@aut.bme.hu

A labor témája:

* Egynézetes alkalmazások
* Alap nézetek használata
* Outlet és akció metódusok

A labor során egy egyszerű számológép alkalmazást készítünk el, melyen keresztül megismerkedünk az iOS-es felhasználói felület készítésének alapjaival.

# iCalculator

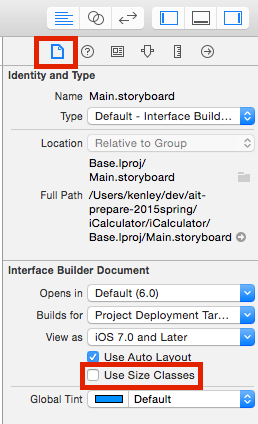
Hozzunk lére egy új "Single View" application-t, "iCalculator névvel" iPhone-ra!

Egy storyboard összefoglalja az alkalmazás felhasználói felületének több "jelenetét". Minden jelenethez tartozik egy nézet hierarchia és egy View Controller (UIViewController), ami az adott nézet hierarchiát menedzseli (fogadja a nézetek eseményeit és konfigurálja a nézeteket).

Lényegében a View Controller-hez tartozó egyedi osztály forráskódjában tudjuk "hozzátenni a logikát" az Interface Builderbne megtervezett nézet hierarchiához.

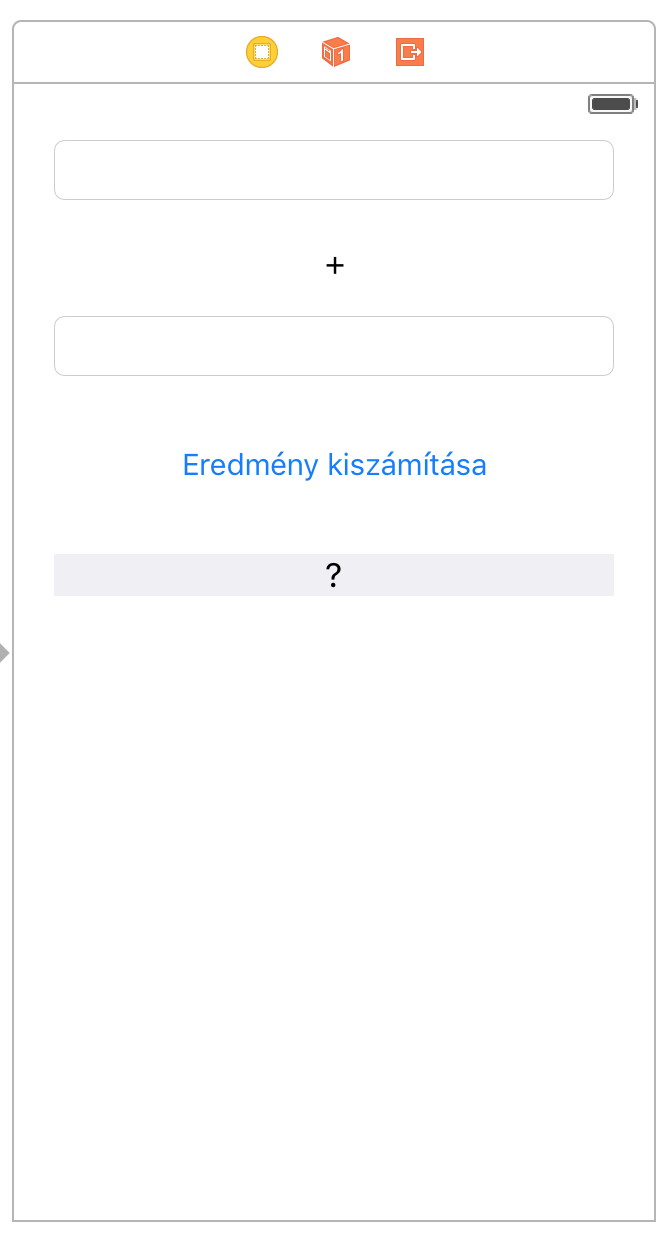
Egy alkalmazásban több storyboard is lehet, a projekt beállítások között lehet megadni melyik legyen az, melyet megjelenít a program indításkor.

Az első lépésként válasszuk ki a Main.storyboard-ot, majd a File Inspectorban kapcsoljuk ki a **Use Size Classes** beállítást:



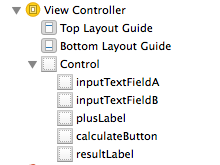
A Size Classes (vagy más néven Adaptive Layout) lehetővé teszi, hogy különféle képernyőméretekre (készülék kategóriákra) és tájolásokra (álló/fekvő) egyetlen nagy közös UI tervet készítsünk. Ezt a funkciót később fogjuk megismerni. Ha kikapcsoljuk a Size Classes-t, akkor a Storyboard azt feltételezi, hogy a felületet csak egyetlen "eszköz típusra" (pl. iPhone) definiáljuk.

Hozzunk létre Main.storyboard-ban, a "ViewController"-en belül lévő View-ban 2 UITextField-et, 2 UILabel-t és 1 UIButton-t.



A nézetek elrendezéséhez most még abszolút koordinátákat használunk, ami miatt elforgatott, vagy eltérő méretű kijelzőn a felület "rosszul" fog megjelenni (nem középen lesz amit középre rakunk). Ezt a problémát oldja meg az Auto Layout, melyről később fogunk tanulni.

Lehetőség van rá, hogy értelmes neveket adjunk az egyes felületelemeknek. Az átnevezéshez válasszunk ki egy elemet a bal szélső listából, majd az Enter megnyomása után átnevezhetjük. Teszteljük a felületet a szimulátorral!

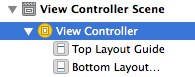
****

Hozzunk létre egy IBOutlet-et ViewController-ben az egyik Text Field-hez!

class ViewController: UIViewController {

@IBOutlet weak var inputTextFieldA: UITextField!

Kössük be a Text View-ot az outletre, a Storyboard-ból a Connections inspector-t használva:

****

****

****

Módosítsuk kódból, a viewDidLoad metódusból a Text Field értékét:

override func viewDidLoad() {

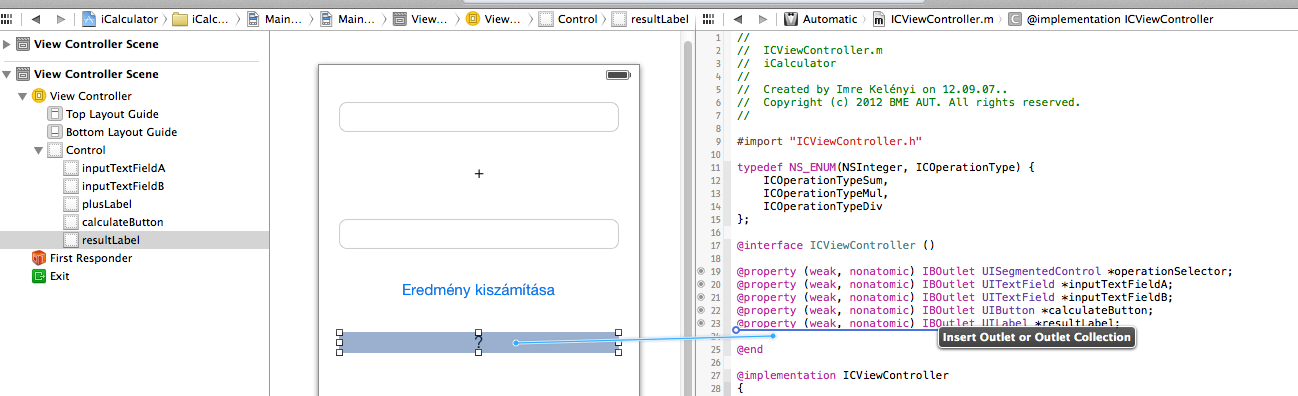
super.viewDidLoad()

self.inputTextFieldA.text = "13"

}

Váltsunk Assistant Editor nézetbe (*hotkey: Cmd+Alt+Enter*), majd hozzunk létre Outleteket az egyes nézetekhez Jobbklikk-Drag&Drop-al (kivéve a + jeles UILabel-hez):



****

@IBOutlet weak var inputTextFieldA: UITextField!

@IBOutlet weak var inputTextFieldB: UITextField!

@IBOutlet weak var resultLabel: UILabel!

Adjunk hozzá egy akciót is a gomb (calculateButton) TouchUpInside akciójához: (calculateButtonTouchUpInside) és valósítsuk meg a gomb lenyomásakor meghívódó metódust:

@IBAction func calculateButtonTouchUpInside(sender: AnyObject) {

let numberFormatter = NSNumberFormatter()

if let

textA = inputTextFieldA.text,

textB = inputTextFieldB.text,

a = numberFormatter.numberFromString(textA)?.doubleValue,

b = numberFormatter.numberFromString(textB)?.doubleValue {

let result = a + b

resultLabel.text = "\(result)"

}

}

A többsoros if valójában négy „if let binding” egymás után végrehajtva. textA, textB, a és b is mind új változók, melyek csak az if-hez tartozó blokkon belül láthatók. Az ilyen, vesszővel elválasztott, több tagú if-eknél az egyes feltételek sorban értékelődnek ki és ha valamelyik feltétel nem sikerül, a többit már nem ellenőrzi a fordító („short-circuit kiértékelés”).

Próbáljuk ki a félkész számológépet!

Állítsuk át a storyboard-ben, a Text Field-ek "Keyboard" attribútumát "**Decimal Pad**"-ra. Ezzel elérjük, hogy egy csak számokat tartalmazó billentyűzet jelenjen meg:



Ezek után adjuk az onCalculateButtonTouchUpInside akció metódusunkhoz a következő két utasítás, melyek hatására el fog tűnni a billentyűzet a képernyőről (amennyiben éppen aktív/látható):

inputTextFieldA.resignFirstResponder()

inputTextFieldB.resignFirstResponder()

A "first responder" az az objektum (esetünkben nézet), mely éppen "fókuszban van" és először fogadja a felhasználótól érkező billentyű eseményeket. Ha egy Text Field nézet lesz a first responder, akkor automatikusan megjelenik a billentyűzet. Ha "lemondunk" a first responder státuszról a **resignFirstResponder()** metódussal, akkor eltűnik a billentyűzet is.

Ha nem tudjuk pontosan hogy épp ki a first responder, akkor elegánsabb megoldás a **view.endEditing(true)** hívás, mely végiglépked a nézet gyereknézetein és mindegyiknél lemond a first reponder státuszról.

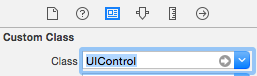
Írjuk át a kódot, hogy egyszerre mondjuk le az összes text field first reponder státuszáról:

view.endEditing(true)

A probléma most már csak az, hogy a feljövő virtuális billentyűzetet csak úgy tudjuk eltüntetni, ha megnyomjuk az számológép gombját. Az elegáns megoldás Number Pad billentyűzet eltüntetésére, hogy a gyökér nézet hátterét bárhol megérintve eltűnjön a billentyűzet.

Ahhoz, hogy a gyökér nézet megérintését le tudjuk kezelni, le kell cserélnünk az osztályát UIView-ról, UIControl-ra (hiszen csak UIControl és belőle származó osztályok tudnak eseményeket generálni). A storyboard szerkesztőben kiválasztva a gyökér nézetet, a **Identity Inspector**-ban választhatjuk ki hozzá a konkrét osztályt, itt váltsunk UIControl-ra:





Ezek után a "Connections inspector"-ban a "Touch Up Inside" eseményhez rendeljünk hozzá egy "onBackgroundTouchUpInside" nevű metódust:

@IBAction func backgroundTouchUpInside (sender: AnyObject) {

view.endEditing(true)

}

He szeretnénk, hogy eltűnjön a már beírt szöveg a Text Editor kiválasztásakor, akkor kapcsoljuk be a "Clear when editing begins" opciót az Attributum inspectorban. Itt adhatjuk meg azt is, hogy a törlés (Clear) gomb mikor jelenjen meg:



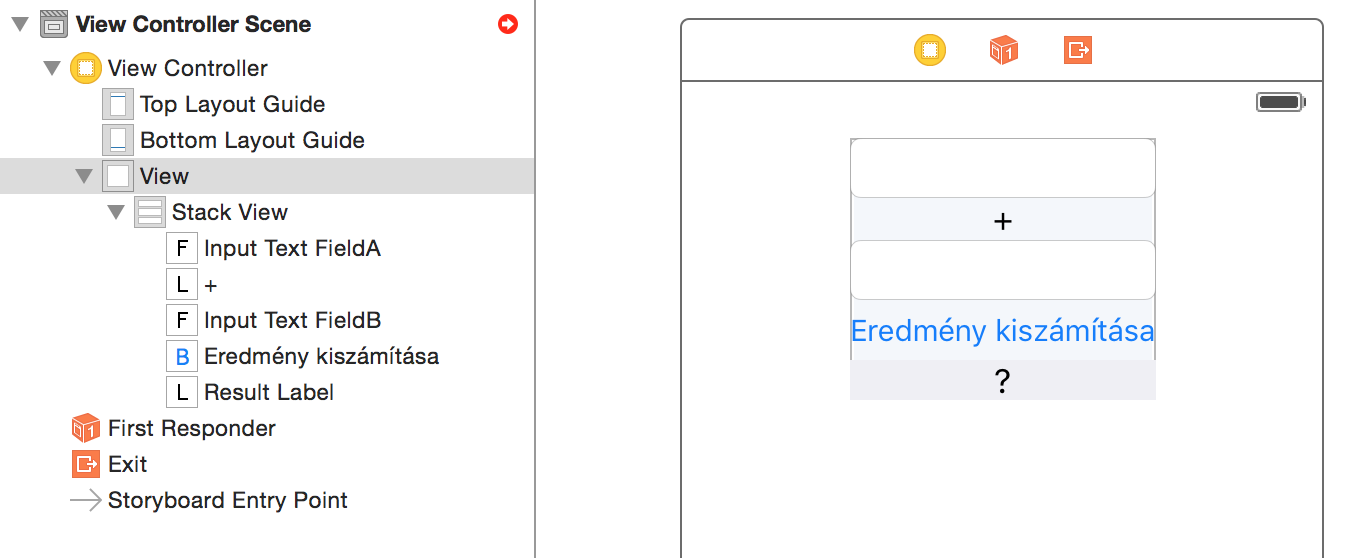
# Stack View

Ha most kipróbáljuk az alkalmazást egy, a grafikus tervezőfelülettel nem megegyező méretű szimulátoron, azt fogjuk tapasztalni, hogy a nézetek nem középre rendezve jelennek meg. Ennek oka, hogy jelenleg abszolút koordinátákkal adtuk meg a nézetek méretét és elhelyezkedését, ami nem változik ha eltérő méretű kijelzőt használunk.

A legegyszerűbb megoldás az elemek dinamikus elrendezéséhez az iOS 9-ben debütáló **Stack View**. Válasszuk ki az összes nézetet, majd nyomjuk meg a „Stack” gombot a szerkesztő nézet jobb alsó sarkában:



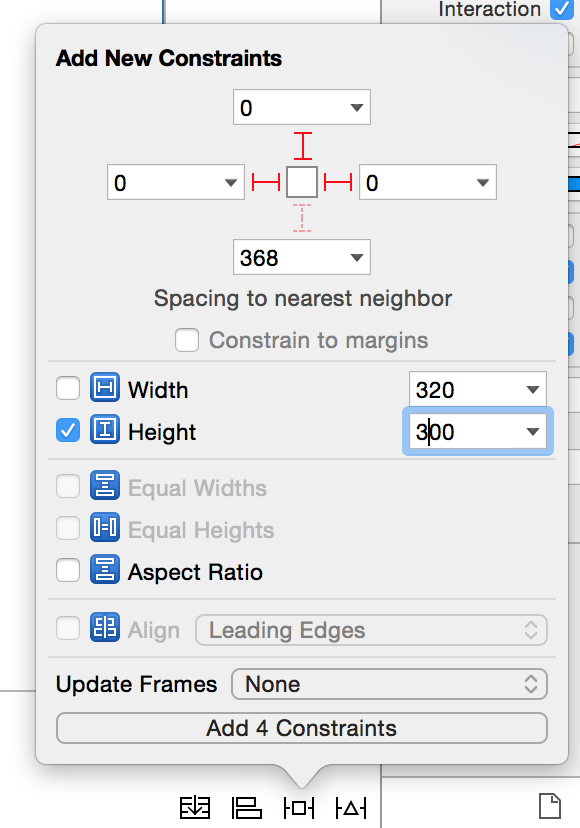
Húzzunk be egy Stack View-t a nézet hierarchiába és méretezzük úgy hogy nagyjából a képernyő felső részét foglalja el. Ezután drag-and-drop mozgassuk át bele a számológép nézeteit. Először valami hasonló, nem túl jól kinéző felületet fogunk látni:

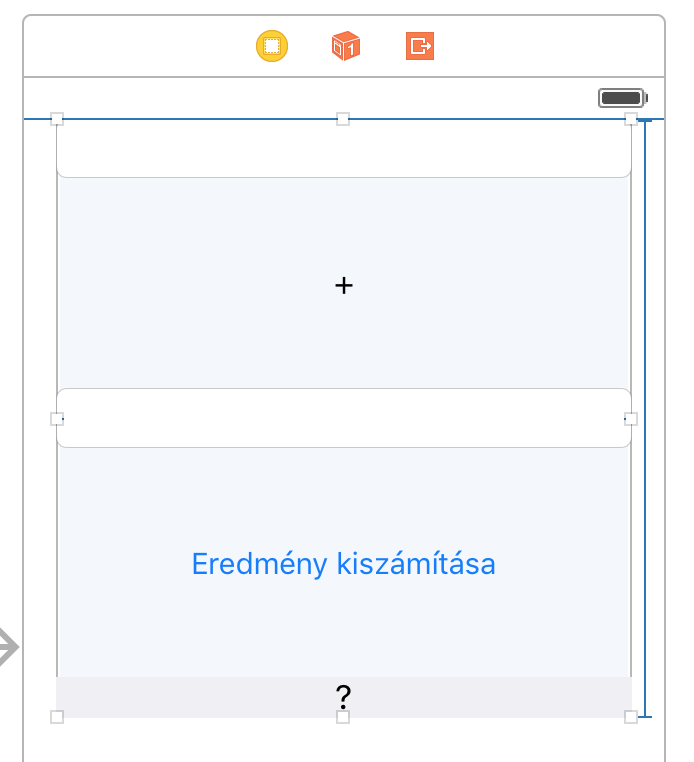


Ahhoz, hogy egyenletesen rendezzük el a nézeteket, válasszuk ki a Stack View-t és állítsuk be a Distribution paraméterét „Equal Spacing”-re:



Utolsó lépésként még be kell állítanunk, hogy maga a Stack View dinamikusan legyen elrendezve a képernyőn. Ehhez az Auto Layout-ot fogjuk használni, amiről a következő laboron még bőven lesz szó. Most egyelőre csak válasszuk ki a Stack View-t és a Pin opciót, majd csatoljuk hozzá a szülő nézetéhez és magasságát állítsuk fixen 300-ra:





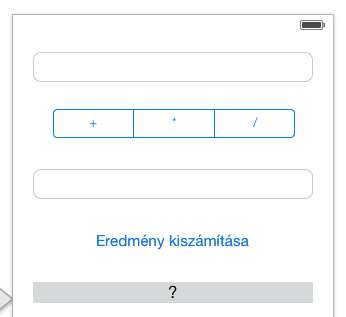
# Önálló feladatok

## Több számológép művelet támogatása

Töröljük ki a "+" jelet megjelenítő Label-t (UILabel) és a helyére húzzunk be egy Segmented Control-t (UISegmentedControl):



Módosítsuk a Segmented Control-t, hogy 3 szegmensből álljon (Attribute inspector-ban a "Segments" attribútum), majd írjuk át ezek szövegét (dupla klikk a szerkesztőben) a szorzás, osztás és összeadás müveletknek megfelelő jelekre:



Az Assistant Editor nézetre váltva, kössük be a Segmented Control "Value Changed" eseményét egy új "operationSelectorValueChanged:" nevű akció metódusra a ViewController osztályban:

@IBAction func operationSelectorValueChanged(sender: AnyObject)

Vegyünk fel egy új enumerációt a ViewController osztályon belül, mely az éppen kiválasztott számológép műveletet jelöli:

enum OperationType {

case Add

case Mul

case Div

}

Továbbá vegyünk fel egy privát tagváltozót az osztály implementációs blokkjába:

var operationType = OperationType.Add

Érdemes a Segmented Control kezdeti értékét is megadni (bár esetünkben ez pont helyesen 0-ra van inicializálva, de nem árt rászokni, hogy mindig inicializáljunk). Ehhez fel kell vennünk egy Outlet-et a Segmented Control-hoz (pl. operationSelector), majd a View Controller viewDidLoad metódusában:

operationSelector.selectedSegmentIndex = 0

Majd implementáljuk az onOperationSelectorValueChanged metódust:

@IBAction func operationSelectorValueChanged(sender: AnyObject) {

switch operationSelector.selectedSegmentIndex {

case 0:

operationType = .Add

case 1:

operationType = .Mul

case 2:

operationType = .Div

default:

operationType = .Add

}

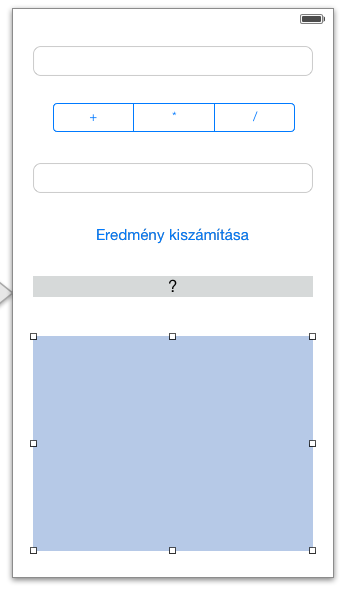
}

Ezek után már csak annyi dolgunk van, hogy a kiszámítást elindító gomb megnyomásakor meghívódó akció metódusban operationType tartalmának megfelelő műveletet végezzünk (itt is érdemes egy switch/case-t használni).

## Korábbi számítások ("history nézet")

Vegyünk fel egy Text View-t a Stack View aljára, majd hozzunk létre hozzá egy IBOutlet-et a View Controller interfészében (pl. textView property névvel):

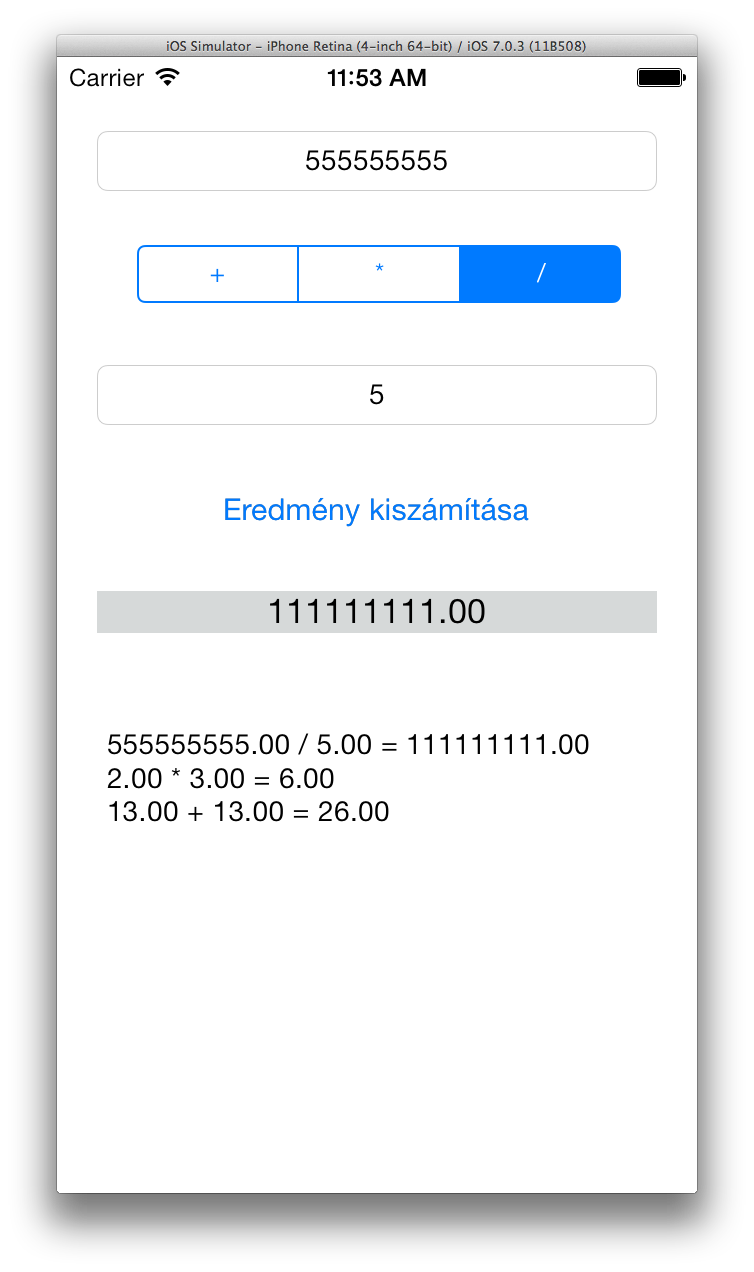




A Text View legfőbb különbségei UILabel-hez képest, hogy szerkeszthető és a kilogó szöveg görgethető.

Állítsuk a Text View-t csak olvasható üzemmódba (self.textView.editable = false)

Módosítsuk az eredményeket kiszámító kódot oly módon, hogy az aktuális számításról bekerüljön egy sor a Text View-ba, pl. "13.00 + 13.00 = 26.00":



Tippek:

* Állítsunk össze egy String-et az új bejegyzéshez
* Állítsuk be a Text View text propertyjének értékét oly módon, hogy az új bejegyzéshez hozzáillesztjük a Text View text property-jének korábbi értékét (textView.text = újSzöveg + textView.text ).

## Extra: alkalmazás ikon :)

Töltsük le, majd állítsuk be az ikont:

<https://www.dropbox.com/s/gmo2l1zi3rg55uu/iCalculator-AppIcon%402x.png?dl=0>