NATIONAL RESEARCH UNIVERSITY
HIGHER SCHOOL OF ECONOMICS

HOMEWORK #1

dedicated to Research Seminar
«Immersion in iOS Development»

MOSCOW 2023

Theme: Creating the first iOS app.
Objective: Learn how to work with Storyboard, and gain experience with Xcode and Swift.

Task description

To complete this homework you must create an iOS app using a storyboard. The app has a single screen with multiple views and a single button. Pressing the button leads to views color and shape changes.

Task requirements

- You must use Storyboard and Swift to complete this task.
- There are no requirements for app architecture for this task.
- To use code from this tutorial or the internet you must understand it first.

Grading

Grade	Task
1	The app has a storyboard, button and views in it.
2-3	Pressing the button changes views colors.
4	Each views color is unique.
5	Pressing the button changes corner radius as well as colors.
6	Color change is animated.
7	The button is disabled during the color change animation.
8	Views are arranged by you in an intersting creative way.
9	Your code does not have magic numbers and has marks.
10	Rundom colors are generated in HEX and then converted to SRGB using an UIColor extension.

Disclaimer

Overdue submissions are **punished** by a decrease in the grade equal to the number of days past the due date. This punishment is limited to **minus 5 points**. Please keep this in mind and plan your week accordingly.

Do not hesitate to ask any questions regarding this task or iOS development in general. Our goal is to enable you to become **the best iOS developer** possible. Teachers and assistants are here to **help**!

Submissions containing extraordinary solutions, exceptional code style, pattern usage, and entertaining features can lead to more points added. This is non-negotiable. Repeating or copying solutions that earned bonus points previously does not grant bonus points.

Code style violations and poor solutions will lead to a grade decrease starting homework #3.

Students who already are able to animate and create UI can call the teacher for the **harder** home task.

Tutorial

Point 1:

Open Xcode and create a new project (Image 1)

Image 1

On the following popup choose iOS as your platform and choose the App option in the «Application» section (Image 2)

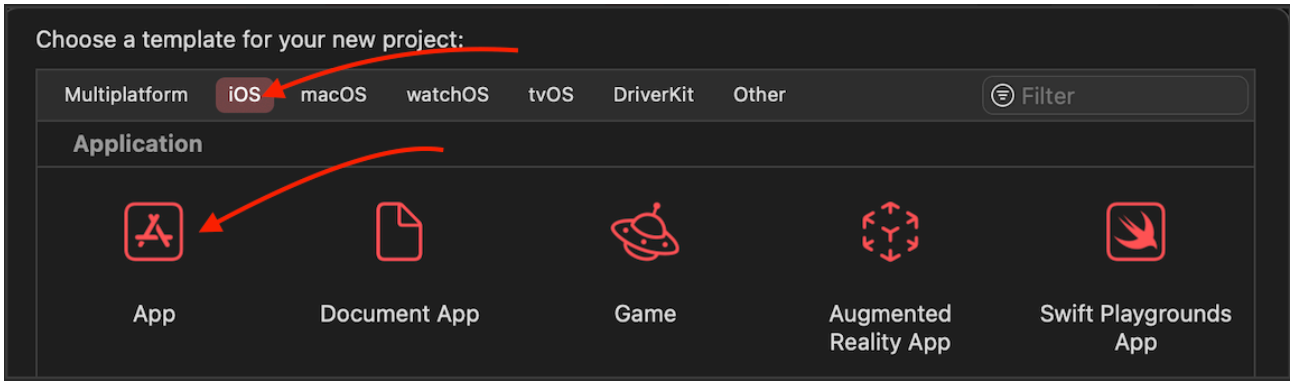

Image 2

Name your project as follows: *your study email before «@»*PW*homework number*. Example: gmsosnovskiyPW1. **Make sure** you selected the **Storyboard** as an interface for this task. Press next and create the app.

Select the Main file on the left file menu. This is our storyboard. Next, find and press a plus button on your top right. In a popup select a button that you prefer (Image 3), and grab and drop it to the iPhone screen. Feel free to set its title and location on the screen.

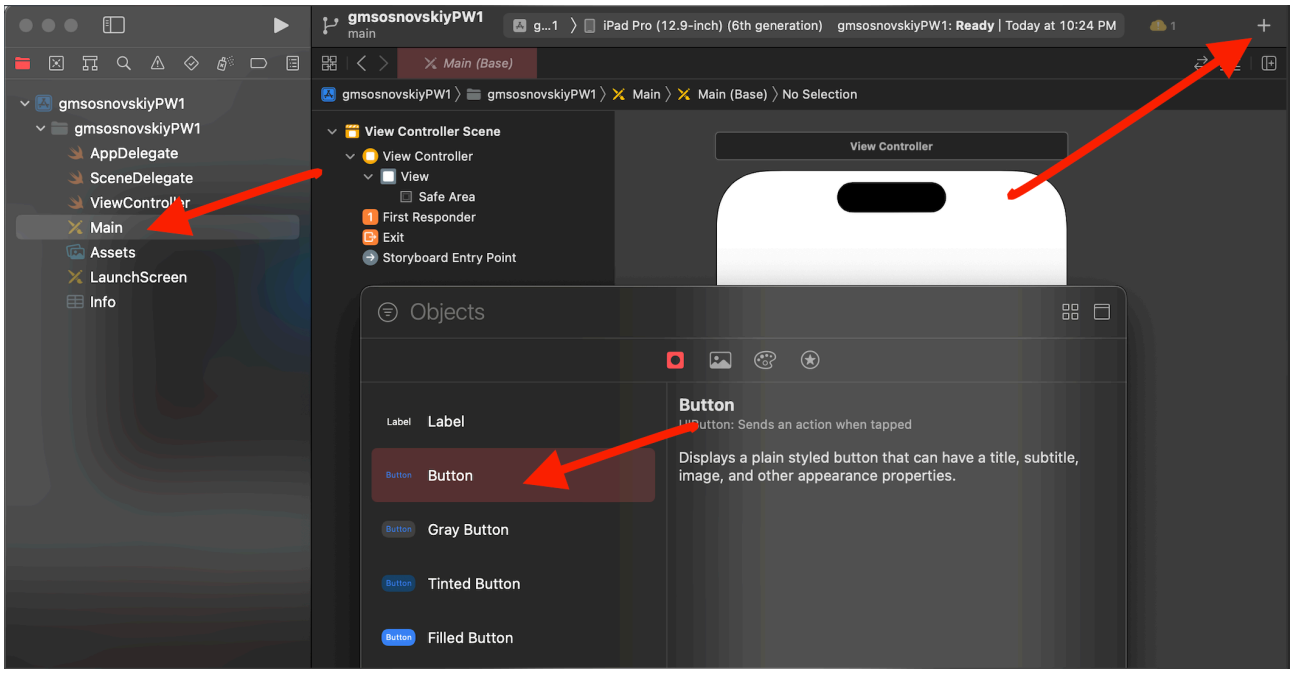

Image 3

Find a view in the same popup and drop several of them (minimum 3, but feel free to go ballistic!) to the View Controller. Choose a background color for your view to make it visible (Image 3).

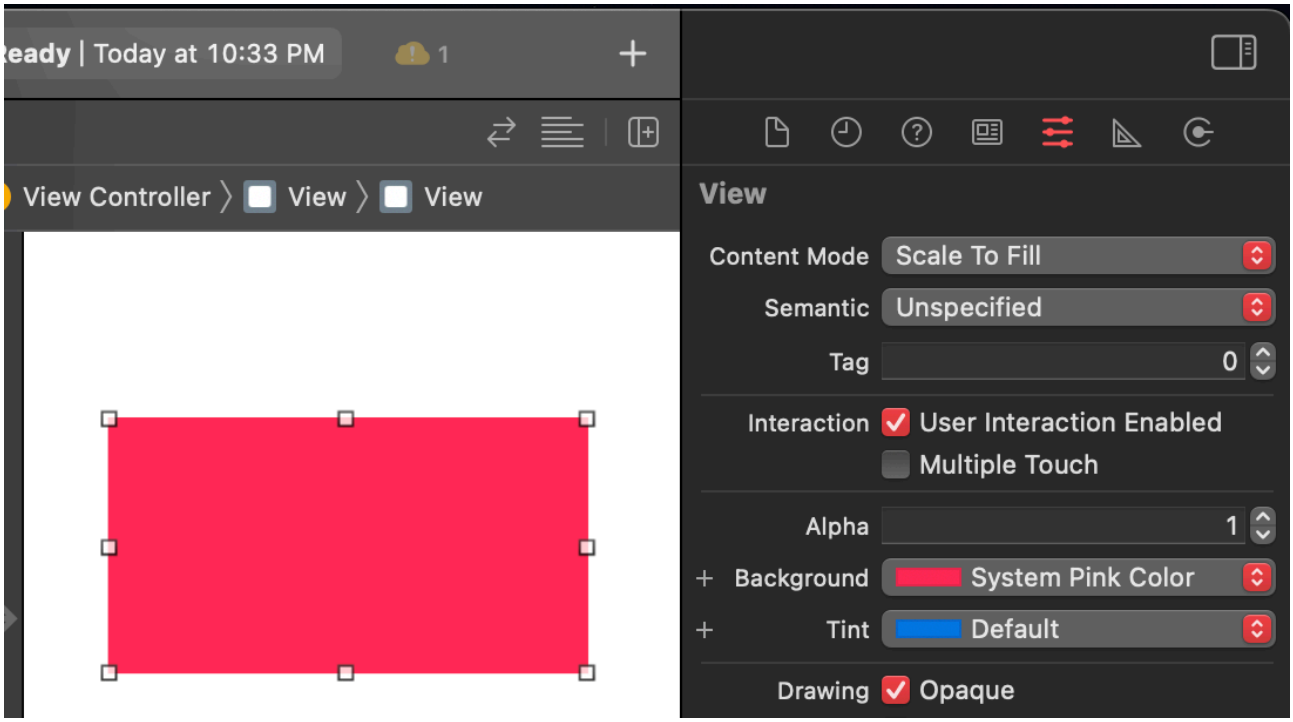

Image 4

Point 2-3:

We need to connect the button and all the views on our view controller with someplace in the code. To do so press down option (alt) key and open the ViewController file (above Main on Image 4). This will open it in side-by-side mode with the storyboard. Now we can press down the control key (do not mistake command for control key) and drag our views right under the ViewController class definition (Image 5).

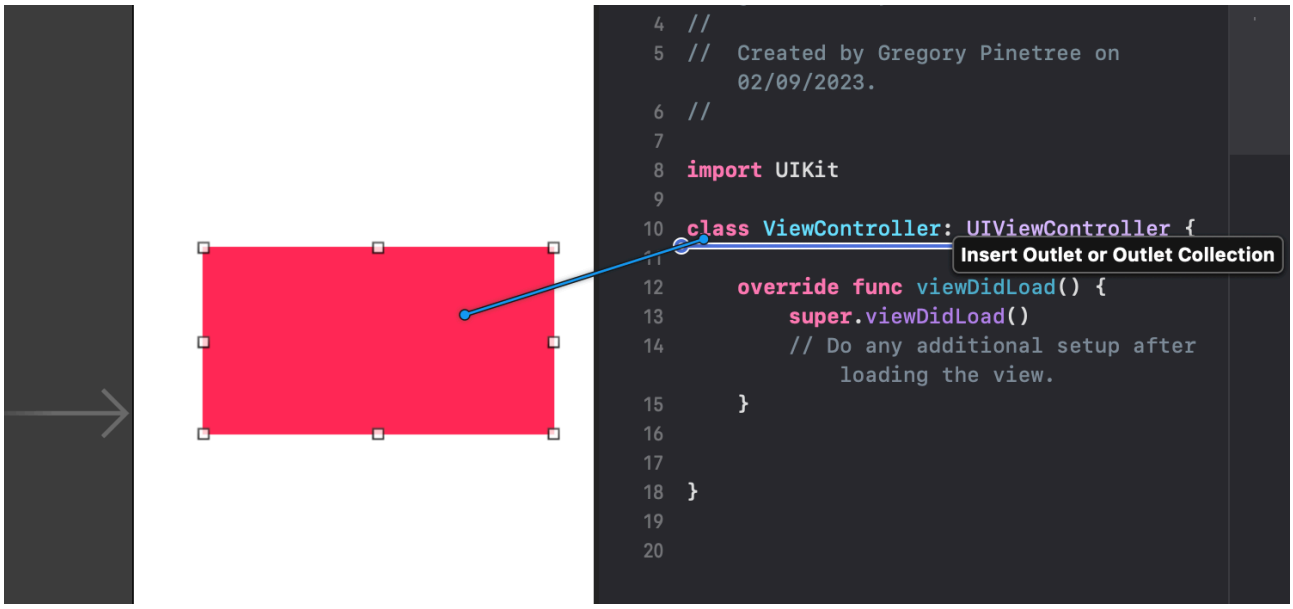

Image 5

You can name them separately, or create an array of views. In the first case, you need to be careful not to drop a view to an outlet of a previously defined view but to create an individual outlet for each one. In the second case, you need to drop all views into a single array.

Case 1:

```
    @IBOutlet weak var view1: UIView!
    @IBOutlet weak var view2: UIView!
    @IBOutlet weak var view3: UIView!
```

Case 2:

```
    @IBOutlet var views: [UIView]!
```

This allows us to address views as fields in the ViewController class. We do not need our button as a field for now, but we need to take a certain action when the user taps on it. So we press down control key and drag and drop our button one line below the end of the **viewDidLoad** function. Make sure the connection type is action! Name this function **buttonWasPressed**, since it will be called on the user pressing the button.

Through this method, we want to change the colors of our views. This can be done by setting a random UIColor to view's backgroundColor property. Sadly, UIColor does not have a .random() method (Maybe some hardworking student can change that?), so we will have to write our own.

Those who use an array to store views will have to figure out how to use "**for**" on their own.

```
view1.backgroundColor = UIColor(
    displayP3Red: .random(in: 0...1),
    green: .random(in: 0...1),
    blue: .random(in: 0...1),
    alpha: 1
)
```

This will change the view's color once the button is pressed. But there is still a long way to go.
Question: Why this won't guarantee us unique colors for our views?

Point 4:

There are many ways to make sure the colors are unique. The best students will create a separate method getUniqueColors() that produces an array of random unique colors, but in this tutorial we will go with the simplest solution possible:

```
        var set = Set<UIColor>()
        while set.count < views.count {
            set.insert(
                UIColor(
                    red: .random(in: 0...1),
                    green: .random(in: 0...1),
                    blue: .random(in: 0...1),
                    alpha: 1
                )
            )
        }
```

```
        for view in views {
            view.backgroundColor = set.popFirst()
        }
```

This time around people who are stuck with separate views will have to figure out how to adjust the code for themselves, what a plot twist.

Please do not just copy this code to your project! Try to come up with your own approach. You are a strong and independent student and you don't need this solution to achieve greatness, right?

Point 5:

To change the view's corner radius you can do the following:

```
    view.layer.cornerRadius = .random(in: 0…25)
```

How you do this to all your views is up to you. Some might believe it is better to extract such logic in a new function (They are correct!).

Point 6:

Animation is a method that belongs to the UIView class. It is important to know, that any change to the app's UI must happen on the main thread (Threads will be studied later). This is so important that this will be on the test.

UIView has a method UIView.animate(withDuration: TimeInterval, animations: {}) that allows us to animate UI changes.

The first parameter is time to animate in seconds (type TimeInterval is a typealias to double. What is typealias by the way?).

The second parameter is the animation itself. This is achieved by the function accepting an anonymous function (You've studied anonymous functions, right? If not try to research it on your own or reach out for help!).

Our color change animation would look something like this:

```
UIView.animate(withDuration: 3.49, animations: {
    view1.backgroundColor = set.popFirst()
    view1.layer.cornerRadius = .random(in: 0...25)
})
```

Point 7:

If we investigate the .animate function we might find a hidden parameter in its semantics called completion. It is an optional anonymous function (anonymous functions in Swift are called callbacks) that will be executed once the animation is finished. So all we need to do is disable our button at the beginning of our buttonWasPressed method and enable it to throw a completion:

```
UIView.animate(
    withDuration: 0.21,
    animations: {
        view1.backgroundColor = set.popFirst()
        view2.layer.cornerRadius = .random(in: 0...25)
    },
    completion: { [weak self] _ in
        self?.button.isEnabled = true
    }
)
```

Here is where you should start to feel lost. You do not know what «[weak self]» is, why «_ in» is there, or what it means? «self?.» triggers an existential crisis. iOS development is scary, but please don't cry underneath the table. All of this will be covered later in the course. All you need to know for now is that is part of memory management in Swift.

There are two ways to get your hands on the button. First is to do the same thing we did to views and create an outlet for our button to make it ViewController's field. Or you can do this:

```
let button = sender as? UIButton
```

If you go with the second option you won't have «[weak self]» or «self?.», but you will have to question your button every time you use it. Example: button**?**.isEnabled = false.

Conclusion:

When run, our views will start with plain colors and not rounded corners. To avoid that we can call methods for random colors and radius change in the viewDidLoad method after super.viewDidLoad().

If you successfully finish this tutorial you can get up to **7 points**. This is a good grade, to earn more you need to do the remaining points on your own. You would have to learn about auto-resizing and auto-layout. You would have to try extensions, show some creativity, and show off your programming skills. **Astonish** homework inspectors and gain bonus points and respect!

Lots of steps of this tutorial are purely educational and can be skipped, but it won't be possible without reading the task first, before doing it head-on. Next time you might want to **read the task before doing it**.

Example of the finished app: