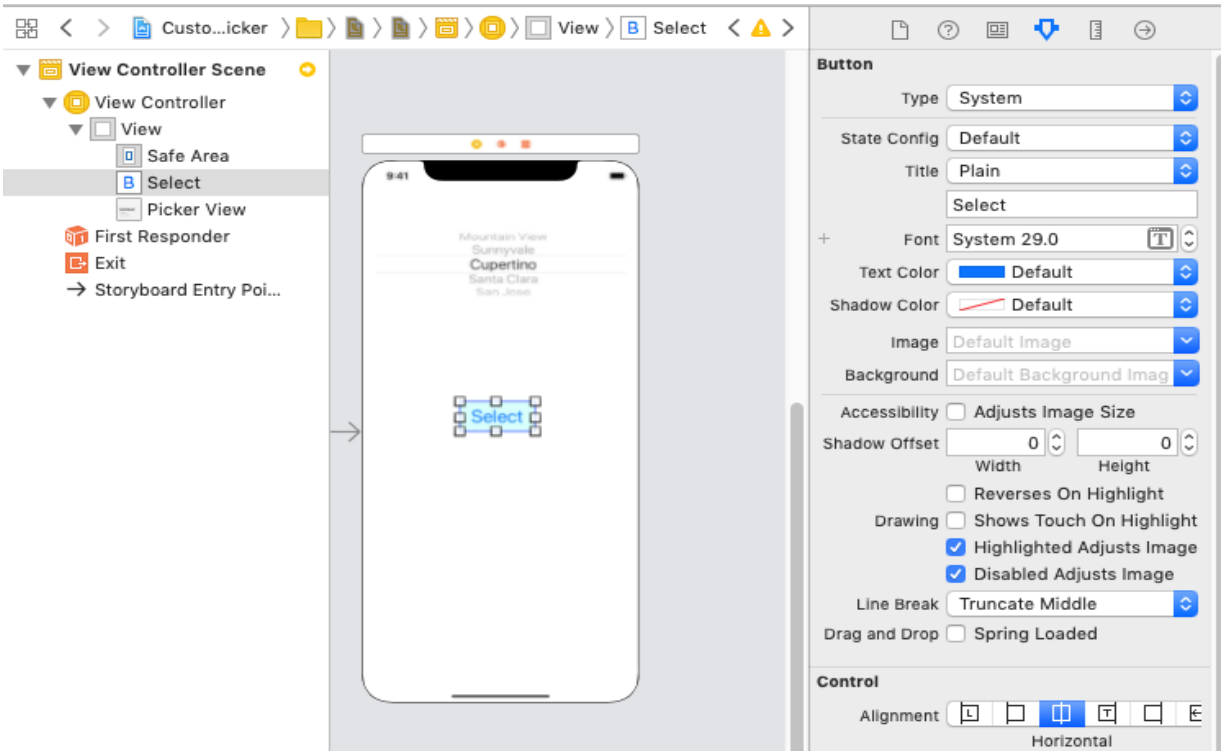
In-class activity on CustomizedPicker

1. Create a new iOS app with XCode and name it as CustomizedPicker. Place a **Button** (with a title:Select) and a **Picker View** into storyboard like this:



2. Connect the View Controller ‘s Picker View from storyboard to its associated swift file (i.e. ViewController.swift) as a new referencing outlet (name it as doublePicker), and connect the Select button with a sent event “Touch Up Inside” to an IBAction method (name it as selectPicker).

   The file should look like this now:

```
import UIKit

class ViewController: UIViewController {
    @IBOutlet weak var doublePicker: UIPickerView!

    @IBAction func selectPicker(_ sender: Any) {
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
    }

}
```

3. Change the class's heading to adopt two protocols: **UIPickerViewDelegate** and **UIPickerViewDataSource.** Then add a couple statements to make the current class (**self**)to be the delegate and dataSource of the Picker View like this inside the viewDidLoad () function::

```
doublePicker.delegate = self
doublePicker.dataSource = self
```

Next insert their associated methods to be implemented. Now the file should look like this:

```
import UIKit

class ViewController: UIViewController, UIPickerViewDelegate, UIPickerViewDataSource  {
    @IBOutlet weak var doublePicker: UIPickerView!

    @IBAction func selectDoublePicker(_ sender: Any) {
    }

    /*
    two methods below need to be implemented for
     UIPickerViewDataSource protocol
     */
    func numberOfComponents(in pickerView: UIPickerView) -> Int {
        // the exact number needs to be determined later
        return 0
    }

    func pickerView(_ pickerView: UIPickerView, numberOfRowsInComponent component:
Int) -> Int {
        // the exact number needs to be determined later
        return 0
    }

/* one method below need to be implemented for UIPickerViewDelegate
    func pickerView(_ pickerView: UIPickerView, titleForRow row: Int, forComponent
component: Int) -> String? {
        //component string needs to be determined to be returned
        return ""
    }


    override func viewDidLoad() {
        super.viewDidLoad()
        doublePicker.delegate = self
        doublePicker.dataSource = self

        // Do any additional setup after loading the view.
    }

}
```

4. Right below the doublePicker IBOutlet, insert some private constants to be used for setting up the **Picker View** with two set of components:

```
private let fillingComponent = 0
private let breadComponent = 1
private let fillingTypes = ["Ham", "Turkey", "Peanut Butter", "Tuna Salad", "Chicken Salad",
                            "Roast Beef", "Vegemite"]
private let breadTypes = [ "White", "Whole Wheat", "Rye", "Sourdough","Seven Grain"]
```

5. Next let's work on the @IBAction method selectPicker with the following code:

```
let fillingRow = doublePicker.selectedRow(inComponent: fillingComponent)
let breadRow = doublePicker.selectedRow(inComponent: breadComponent)

let filling = fillingTypes[fillingRow]
let bread = breadTypes[breadRow]
let message = "Your \(filling) on \(bread) bread will be right up."

// create an Alert message to display the selected items
let alert = UIAlertController(
    title: "Thank you for your order",
    message: message,
    preferredStyle: .alert)
let action = UIAlertAction(
    title: "Great",
    style: .default,
    handler: nil)
  alert.addAction(action)
  present(alert, animated: true, completion: nil)
```

6. Now determine the number of components (fillingComponent and breadComponent) to be used for setting up the Picker View -> 2. So change the number zero to 2 inside the function numberOfComponents():

```
func numberOfComponents(in pickerView: UIPickerView) -> Int {
    // the exact number needs to be determined later
    return 2
}
```

7. Now implement the code to determine the number of rows for a given component being set up in the Picker View. Replace the statement 'return 0' with the following if-else statement inside the first pickerView function from UIPickerViewDataSource:

```
if component == breadComponent {
    return breadTypes.count
} else {
    return fillingTypes.count
}
```

8. Now implement the code to return the item string with given a **row** number for a given component for the pickerView function from UIPickerViewDelegate. This is used to display the text on the Picker view when you select a row from a component. Now replace the statement 'return ""' with the following if-else statement :

```
if component == breadComponent {
    return breadTypes[row]
} else {
    return fillingTypes[row]
}
```

9. Run the app. Select Chicken Salad and Whole Wheat from the picker, and click on the Select button to show the alert box. Take a screen shot for submission.

10. Now try to add one more component to the doublePicker to make it a triple picker:

* Change the variable name from doublePicker to triplePicker. Cancel the previous connection and then reconnect again to the new variable name. Change references inside the selectPicker() and viewDidLoad() methods from doublePicker to triplePicker.
* add a new component named quantityComponent with index value of 2

```
private let quantityComponent = 2
```

* add a new type named quanityTypes:

```
private let quantityTypes = ["1", "2", "3", "4", "5"]
```

* change the number being returned in numberOfComponents() method from 2 to 3

```
return 3
```

* add two statements inside selectPicker() method:

```
let quantityRow = triplePicker.selectedRow(inComponent: quantityComponent)
let quantity = quantityTypes[quantityRow]
```

* modify the message to include the quantity like this:

```
let message = "Your \(quantity) orders  of \(filling) on \(bread) bread will be right up."
```

* modify the first pickerView() method to include quantityComponent like this:

```
if component == quantityComponent {
    return quantityTypes.count
}
else if component == breadComponent {
    return breadTypes.count
}
else {
    return fillingTypes.count
}
```

* modify the second pickerView() method to include quantityComponent like this:

```
if component == quantityComponent {
    return quantityTypes[row]
}
else if component == breadComponent {
    return breadTypes[row]
}
else {
    return fillingTypes[row]
}
```