```
// checklist app
//  ViewController.swift
//  Checklist
//
//  Created by Brian on 6/18/18.
//  Copyright © 2018 Razeware. All rights
reserved.
//

import UIKit

class ChecklistViewController:
UITableViewController {

  var todoList: TodoList

  private func priorityForSectionIndex(_ index:
Int) -> TodoList.Priority? {
    return TodoList.Priority(rawValue: index)
  }

  @IBAction func addItem(_ sender: Any) {

    let newRowIndex =
todoList.todoList(for: .medium).count
    _ = todoList.newTodo()  func

    let indexPath = IndexPath(row: newRowIndex, ?
section: 0)
    let indexPaths = [indexPath]
    tableView.insertRows(at: indexPaths,
with: .automatic)

  }

  @IBAction func deleteItems(_ sender: Any) {
    if let selectedRows =
tableView.indexPathsForSelectedRows {
      for indexPath in selectedRows {
        if let priority =
priorityForSectionIndex(indexPath.section) {
```

```
            let todos = todoList.todoList(for:
priority)

            let rowToDelete = indexPath.row >
todos.count - 1 ? todos.count - 1 : indexPath.row
            let item = todos[rowToDelete]

            todoList.remove(item, from: priority,
at: rowToDelete)
          }
        }
        tableView.beginUpdates()
        tableView.deleteRows(at: selectedRows,
with: .automatic)
        tableView.endUpdates()
      }
    }
```

??

reload()?

class init men hvorfor

```
  required init?(coder aDecoder: NSCoder) {
    todoList = TodoList()
    super.init(coder: aDecoder)
  }

  override func viewDidLoad() {
    super.viewDidLoad()


navigationController?.navigationBar.prefersLargeTi
tles = true
    navigationItem.leftBarButtonItem =
editButtonItem
    tableView.allowsMultipleSelectionDuringEditing
= true
  }

  override func setEditing(_ editing: Bool,
animated: Bool) {
    super.setEditing(editing, animated: true)
    tableView.setEditing(tableView.isEditing,
animated: true)
```

?

```
  }

  override func tableView(_ tableView:
UITableView, numberOfRowsInSection section: Int)
-> Int {
    if let priority =
priorityForSectionIndex(section) {
      return todoList.todoList(for:
priority).count
    }
    return 0
  }

  override func tableView(_ tableView:
UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {
    let cell =
tableView.dequeueReusableCell(withIdentifier:
"ChecklistItem", for: indexPath)
    //let item = todoList.todos[indexPath.row]
    if let priority =
priorityForSectionIndex(indexPath.section) {
      let items = todoList.todoList(for: priority)
      let item = items[indexPath.row]
      configureText(for: cell, with: item)
      configureCheckmark(for: cell, with: item)
    }
    return cell
  }

  override func tableView(_ tableView:
UITableView, didSelectRowAt indexPath: IndexPath)
{
    if tableView.isEditing {
      return
    }
    if let cell = tableView.cellForRow(at:
indexPath) {
      if let priority =
priorityForSectionIndex(indexPath.section) {
        let items = todoList.todoList(for:
```

```
priority)
        let item = items[indexPath.row]
        item.toggleChecked()
        configureCheckmark(for: cell, with: item)
        tableView.deselectRow(at: indexPath,
animated: true)
      }

    }
  }

  override func tableView(_ tableView:
UITableView, commit editingStyle:
UITableViewCell.EditingStyle, forRowAt indexPath:
IndexPath) {
    if let priority =
priorityForSectionIndex(indexPath.section) {
      let item = todoList.todoList(for: priority)
[indexPath.row]
      todoList.remove(item, from: priority, at:
indexPath.row)
      let indexPaths = [indexPath]
      tableView.deleteRows(at: indexPaths,
with: .automatic)
    }

  }

  override func tableView(_ tableView:
UITableView, moveRowAt sourceIndexPath: IndexPath,
to destinationIndexPath: IndexPath) {

    if let srcPriority =
priorityForSectionIndex(sourceIndexPath.section),
       let destPriority =
priorityForSectionIndex(destinationIndexPath.secti
on) {

      let item = todoList.todoList(for:
srcPriority)[sourceIndexPath.row]
      todoList.move(item: item, from: srcPriority,
```

```
at: sourceIndexPath.row, to: destPriority, at:
destinationIndexPath.row)
    }

    tableView.reloadData()
  }

  func configureText(for cell: UITableViewCell,
with item: ChecklistItem) {
    if let checkmarkCell = cell as?
ChecklistTableViewCell {
      checkmarkCell.todoTextLabel.text = item.text
    }
  }

  func configureCheckmark(for cell:
UITableViewCell, with item: ChecklistItem) {
    guard let checkmarkCell = cell as?
ChecklistTableViewCell else {
      return
    }
    if item.checked {
      checkmarkCell.checkmarkLabel.text = "√"
    } else {
      checkmarkCell.checkmarkLabel.text = ""
    }
  }

  override func prepare(for segue:
UIStoryboardSegue, sender: Any?) {
    if segue.identifier == "AddItemSegue" {
      if let itemDetailViewController =
segue.destination as? ItemDetailViewController {
        itemDetailViewController.delegate = self
        itemDetailViewController.todoList =
todoList
      }
    } else if segue.identifier ==
"EditItemSegue" {
      if let itemDetailViewController =
segue.destination as? ItemDetailViewController {
```

```
        if let cell = sender as? UITableViewCell,
           let indexPath =
tableView.indexPath(for: cell),
           let priority =
priorityForSectionIndex(indexPath.section)
          {

          let item = todoList.todoList(for:
priority)[indexPath.row]
          itemDetailViewController.itemToEdit =
item
          itemDetailViewController.delegate = self
        }
      }
    }
  }

  override func numberOfSections(in tableView:
UITableView) -> Int {
    return TodoList.Priority.allCases.count
  }

  override func tableView(_ tableView:
UITableView, titleForHeaderInSection section: Int)
-> String? {
    var title: String? = nil
    if let priority =
priorityForSectionIndex(section) {
      switch priority {
      case .high:
        title = "High Priority Todos"
      case .medium:
        title = "Medium Priority Todos"
      case .low:
        title = "Low Priority Todos"
      case .no:
        title = "Someday Todo Items"
      }
    }
    return title
  }
```

```
}

extension ChecklistViewController:
ItemDetailViewControllerDelegate {
  func itemDetailViewControllerDidCancel(_
controller: ItemDetailViewController) {

navigationController?.popViewController(animated:
true)
  }

  func itemDetailViewController(_ controller:
ItemDetailViewController, didFinishAdding item:
ChecklistItem) {

navigationController?.popViewController(animated:
true)
    let rowIndex =
todoList.todoList(for: .medium).count - 1
    let indexPath = IndexPath(row: rowIndex,
section: TodoList.Priority.medium.rawValue)
    let indexPaths = [indexPath]
    tableView.insertRows(at: indexPaths,
with: .automatic)
  }

  func itemDetailViewController(_ controller:
ItemDetailViewController, didFinishEditing item:
ChecklistItem) {

    for priority in TodoList.Priority.allCases {
      let currentList = todoList.todoList(for:
priority)
      if let index = currentList.index(of: item) {
        let indexPath = IndexPath(row: index,
section: priority.rawValue)
        if let cell = tableView.cellForRow(at:
indexPath) {
          configureText(for: cell, with: item)
```

```
        }
      }
    }

navigationController?.popViewController(animated:
true)
  }



}

//
//  ChecklistTableViewCell.swift
//  Checklist
//
//  Created by Brian on 6/20/18.
//  Copyright © 2018 Razeware. All rights
reserved.
//

import UIKit

class ChecklistTableViewCell: UITableViewCell {

  @IBOutlet weak var checkmarkLabel: UILabel!
  @IBOutlet weak var todoTextLabel: UILabel!


    override func awakeFromNib() {
        super.awakeFromNib()
        // Initialization code
    }

    override func setSelected(_ selected: Bool,
animated: Bool) {
        super.setSelected(selected, animated:
animated)

        // Configure the view for the selected
state
    }
```

```
}
//
//  ChecklistItem.swift
//  Checklist
//
//  Created by Brian on 6/19/18.
//  Copyright © 2018 Razeware. All rights
reserved.
//

import Foundation

class ChecklistItem: NSObject {

  @objc var text = ""
  var checked = false

  func toggleChecked() {
    checked = !checked
  }

}



//
//  TodoList.swift
//  Checklist
//
//  Created by Brian on 6/19/18.
//  Copyright © 2018 Razeware. All rights
reserved.
//

import Foundation

class TodoList {

  enum Priority: Int, CaseIterable {
    case high, medium, low, no
  }
```

?

```
  private var highPriorityTodos: [ChecklistItem] =
[]
  private var mediumPriorityTodos: [ChecklistItem]
= []
  private var lowPriorityTodos: [ChecklistItem] =
[]
  private var noPriorityTodos: [ChecklistItem] =
[]

  init() {

    let row0Item = ChecklistItem()
    let row1Item = ChecklistItem()
    let row2Item = ChecklistItem()
    let row3Item = ChecklistItem()
    let row4Item = ChecklistItem()
    let row5Item = ChecklistItem()
    let row6Item = ChecklistItem()
    let row7Item = ChecklistItem()
    let row8Item = ChecklistItem()
    let row9Item = ChecklistItem()

    row0Item.text = "Take a jog"
    row1Item.text = "Watch a movie"
    row2Item.text = "Code an app"
    row3Item.text = "Walk the dog"
    row4Item.text = "Study design patterns"
    row5Item.text = "Go camping"
    row6Item.text = "Pay bills"
    row7Item.text = "Plan vacation"
    row8Item.text = "Walk the cat"
    row9Item.text = "Play games"

    addTodo(row0Item, for: .medium)
    addTodo(row1Item, for: .low)
    addTodo(row2Item, for: .high)
    addTodo(row3Item, for: .no)
    addTodo(row4Item, for: .high)
    addTodo(row5Item, for: .medium)
    addTodo(row6Item, for: .low)
```

func

```
    addTodo(row7Item, for: .high)
    addTodo(row8Item, for: .no)
    addTodo(row9Item, for: .high)

  }

  func addTodo(_ item: ChecklistItem, for
priority: Priority, at index: Int = -1) {
    switch priority {
    case .high:
      if index < 0 {
        highPriorityTodos.append(item)
      } else {
        highPriorityTodos.insert(item, at: index)
      }
    case .medium:
      if index < 0 {
        mediumPriorityTodos.append(item)
      } else {
        mediumPriorityTodos.insert(item, at:
index)
      }
    case .low:
      if index < 0 {
        lowPriorityTodos.append(item)
      } else {
        lowPriorityTodos.insert(item, at: index)
      }
    case .no:
      if index < 0 {
        noPriorityTodos.append(item)
      } else {
        noPriorityTodos.insert(item, at: index)
      }
    }
  }

  func todoList(for priority: Priority) ->
[ChecklistItem] {
    switch priority {
    case .high:
```

```
        return highPriorityTodos
      case .medium:
        return mediumPriorityTodos
      case .low:
        return lowPriorityTodos
      case .no:
        return noPriorityTodos
      }
    }

    func newTodo() -> ChecklistItem {
      let item = ChecklistItem()
      item.text = randomTitle()
      item.checked  = true
      mediumPriorityTodos.append(item)
      return item
    }

    func move(item: ChecklistItem, from
sourcePriority: Priority, at sourceIndex: Int, to
destinationPriority: Priority, at
destinationIndex: Int) {
      remove(item, from: sourcePriority, at:
sourceIndex)
      addTodo(item, for: destinationPriority, at:
destinationIndex)

    }

    func remove(_ item: ChecklistItem, from
priority: Priority, at index: Int) {
      switch priority {
      case .high:
        highPriorityTodos.remove(at: index)
      case .medium:
        mediumPriorityTodos.remove(at: index)
      case .low:
        lowPriorityTodos.remove(at: index)
      case .no:
        noPriorityTodos.remove(at: index)
      }
```

default new entry level

```
  }

  private func randomTitle() -> String {
    var titles = ["New todo item", "Generic todo",
"Fill me out", "I need something to do", "Much
todo about nothing"]
    let randomNumber = Int.random(in: 0 ...
titles.count - 1)
    return titles[randomNumber]
  }

}

//
//  AddItemTableViewController.swift
//  Checklist
//
//  Created by Brian on 6/19/18.
//  Copyright © 2018 Razeware. All rights
reserved.
//

import UIKit

protocol ItemDetailViewControllerDelegate: class {
  func itemDetailViewControllerDidCancel(_
controller: ItemDetailViewController)
  func itemDetailViewController(_ controller:
ItemDetailViewController, didFinishAdding item:
ChecklistItem)
  func itemDetailViewController(_ controller:
ItemDetailViewController, didFinishEditing item:
ChecklistItem)
}

class ItemDetailViewController:
UITableViewController {

  weak var delegate:
ItemDetailViewControllerDelegate?
```

```
  weak var todoList: TodoList?
  weak var itemToEdit: ChecklistItem?

  @IBOutlet weak var cancelBarButton:
UIBarButtonItem!
  @IBOutlet weak var addBarButton:
UIBarButtonItem!
  @IBOutlet weak var textfield: UITextField!


  @IBAction func cancel(_ sender: Any) {

delegate?.itemDetailViewControllerDidCancel(self)
  }

  @IBAction func done(_ sender: Any) {
    if let item = itemToEdit, let text =
textfield.text {
      item.text = text
      delegate?.itemDetailViewController(self,
didFinishEditing: item)
    } else {
      if let item = todoList?.newTodo() {
        if let textFieldText = textfield.text {
          item.text = textFieldText
        }
        item.checked = false
        delegate?.itemDetailViewController(self,
didFinishAdding: item)
      }
    }



  }

  override func viewDidLoad() {
    super.viewDidLoad()
    if let item = itemToEdit {
      title = "Edit Item"
      textfield.text = item.text
      addBarButton.isEnabled = true
```

```
    }
    navigationItem.largeTitleDisplayMode = .never
  }

  override func viewWillAppear(_ animated: Bool) {
    textfield.becomeFirstResponder()
  }
  override func tableView(_ tableView:
UITableView, willSelectRowAt indexPath: IndexPath)
-> IndexPath? {
    return nil
  }
}

extension ItemDetailViewController:
UITextFieldDelegate {

  func textFieldShouldReturn(_ textField:
UITextField) -> Bool {
    textfield.resignFirstResponder()
    return false
  }

  func textField(_ textField: UITextField,
shouldChangeCharactersIn range: NSRange,
replacementString string: String) -> Bool {

    guard let oldText = textfield.text,
          let stringRange = Range(range, in:
oldText) else {
        return false
    }

    let newText = oldText.replacingCharacters(in:
stringRange, with: string)
    if newText.isEmpty {
      addBarButton.isEnabled = false
    } else {
      addBarButton.isEnabled = true
    }
    return true
```

VC açılınca aktive olsun

aynısı tekrar yazılmasın diye!

1 harf bile değişse newText

```
  }


}
```

PLAYGROUND

```
import UIKit

protocol Persist {
  func save()
}

class Monster: Persist {
  func save() {
    print("Monster save")
  }
}

class Sword: Persist {
  func save() {
    print("Sword save")
  }
}

class Player {

}

let monster = Monster()
let sword = Sword()
let player = Player()

let items: [Persist] = [monster, sword]

class GameManager {

  func saveLevel(_ items: [Persist]) {
    for item in items {
      item.save()
    }
  }
```

```
}

let gameManager = GameManager()
gameManager.saveLevel(items)


//
//  AppDelegate.swift
//  Checklist
//
//  Created by Brian on 6/18/18.
//  Copyright © 2018 Razeware. All rights
reserved.
//

import UIKit

@UIApplicationMain
class AppDelegate: UIResponder,
UIApplicationDelegate {

  var window: UIWindow?


  func application(_ application: UIApplication,
didFinishLaunchingWithOptions launchOptions:
[UIApplication.LaunchOptionsKey: Any]?) -> Bool {
    // Override point for customization after
application launch.
    return true
  }

  func applicationWillResignActive(_ application:
UIApplication) {
    // Sent when the application is about to move
from active to inactive state. This can occur for
certain types of temporary interruptions (such as
an incoming phone call or SMS message) or when the
user quits the application and it begins the
transition to the background state.
    // Use this method to pause ongoing tasks,
```

```
disable timers, and invalidate graphics rendering
callbacks. Games should use this method to pause
the game.
  }

  func applicationDidEnterBackground(_
application: UIApplication) {
    // Use this method to release shared
resources, save user data, invalidate timers, and
store enough application state information to
restore your application to its current state in
case it is terminated later.
    // If your application supports background
execution, this method is called instead of
applicationWillTerminate: when the user quits.
  }

  func applicationWillEnterForeground(_
application: UIApplication) {
    // Called as part of the transition from the
background to the active state; here you can undo
many of the changes made on entering the
background.
  }

  func applicationDidBecomeActive(_ application:
UIApplication) {
    // Restart any tasks that were paused (or not
yet started) while the application was inactive.
If the application was previously in the
background, optionally refresh the user interface.
  }

  func applicationWillTerminate(_ application:
UIApplication) {
    // Called when the application is about to
terminate. Save data if appropriate. See also
applicationDidEnterBackground:.
  }
```

}