Codility_

Krunal Chauhan

Candidate

E-mail:
chauhankrunal3909@gmail.com

Session

ID: K8JDWV-MWH
Time limit: 50 min.
Report recipients: No one
Accessed from: 124.123.160.187, 124.123.160.187
Invited by: arun.k@uplers.in

Status: completed

Created on: 2023-08-22 04:35 UTC
Started on: 2023-08-22 04:36 UTC
Finished on: 2023-08-22 05:26 UTC

Notes:
N/A

Similarity Check

Status: not found
No similar solutions have been detected.

Test score

100%

Tasks in test	Score
1 SessionWatcher Submitted in: Swift	100%

Canvas Details

The canvas board was not used during that session.

Tasks Details

Medium

1. SessionWatcher

Implement a session watcher in your Swift code.

Task Score	Correctness	Performance
100	100	Not assessed

Task description

Finish the implementation of `SessionWatcher` and `DefaultWorkItemProvider`. `SessionWatcher` will be responsible for tracking user inactivity in the app for a given period of time. If there was no user action received during the given session time then `onTimeExceeded` completion should be called from `SessionWatcher` instance. Creation of a `DispatchWorkItem` will take place in `DefaultWorkItemProvider`. The created work item will be used in the `SessionWatcher` class.

Requirements

Provide an implementation for the following methods and classes:

`DefaultWorkItemProvider`

- `func workItem(actionBlock: @escaping () -> ()) -> DispatchWorkItem?`
  - This method should return `DispatchWorkItem` with the given `actionBlock` to execute.

`SessionWatcher`

- `start()`
  - The `start` method will be called when the session time starts. As an effect, `DispatchWorkItem` created from `DefaultWorkItemProvider` should be started to track the user's inactivity.
  - A new `DispatchWorkItem` needs to be created when `start` is called.
  - The created `DispatchWorkItem` should be executed asynchronously on a queue given in the `SessionWatcher` initializer.
  - The actionBlock of `DispatchWorkItem` should be called after the given `sessionTime`.
  - The previously created `DispatchWorkItem` needs to be cancelled.
- `receivedUserAction()`
  - This method will be called every time the user triggers some action in the app. So `SessionWatcher` should take care of any `DispatchWorkItem` that has already been created, and start counting again with the new one.
- `stop()`
  - The `stop` method will be called when the app stops watching the session time. The current `DispatchWorkItem` should be cancelled.

Protocols and structures

Below, you can find the protocol that is used in `SessionWatcher`. It will be useful if you want to work in a separate Xcode project or playground in order to provide a solution.

Please do not change the following protocol. Do not copy it to the editor - it will be added automatically during compilation process

```
protocol WorkItemProvider {
    func workItem(actionBlock: @escaping () -> ()) -> DispatchWorkItem?
}
```

Environment

- Swift 5.3
- Foundation library

Hints

- Do not create a retain cycle if you work with completions.
- Use the `Foundation` framework; documentation can be found here.

- You don't need to use sleep function to fulfil delay requirement. Using `asyncAfter` on queue might be helpful.
- Documentation for `DispatchWorkItem` might be useful too.

Solution

See Live Version

Programming language used:	Swift
Total time used:	50 minutes
Effective time used:	50 minutes
Notes:	*not defined yet*

Source code

Code: 05:26:01 UTC, swift, final, score: **100**

```swift
import Foundation

class DefaultWorkItemProvider: WorkItemProvider {
    func workItem(actionBlock: @escaping () -> ()) -> DispatchWorkItem? {
        let newWorkItem = DispatchWorkItem { [weak self] in
            print ("DefaultWorkItemProvider")
                actionBlock()
        }
        return newWorkItem
    }
}

class SessionWatcher {
    private var workItemProvider: WorkItemProvider
    private var workItem: DispatchWorkItem?
    private let sessionTime: TimeInterval
    private let queue: DispatchQueue

    var onTimeExceeded: (() -> Void)?

    init(sessionTime: TimeInterval = 5, workItemProvider: WorkItemProvider, queue: DispatchQueue) {
        self.workItemProvider = workItemProvider
        self.sessionTime = sessionTime
        self.queue = queue
        let newWorkItem = DefaultWorkItemProvider().workItem { [weak self] in
            print ("init")
           self?.onTimeExceeded?()
        }
        workItem = newWorkItem
        if let workItem = workItem {
            queue.asyncAfter(deadline: .now() + sessionTime, execute: workItem)
        }
    }

    func cancelWorkTime() {
    }
    func start() {
        workItem?.cancel()
        let newWorkItem = workItemProvider.workItem { [weak self] in
            print ("start")
            self?.onTimeExceeded?()
        }
        workItem = newWorkItem
        if let workItem = workItem {
            queue.asyncAfter(deadline: .now() + sessionTime, execute: workItem)
        }
    }

    func receivedUserAction() {
        workItem?.cancel()
        start()
    }

```

```
54    func stop() {
55            workItem?.cancel()
56    }
57}
```

Analysis summary

The solution obtained perfect score.

Analysis

Correctness tests	
TaskTests.TaskTests - test_startAction_onTimeExceededHasNotBeenCalledAfterStart	✔ OK
TaskTests.TaskTests - test_startAction_workItemCancelledOnStart	✔ OK
TaskTests.TaskTests - test_receivedUserAction_secondWorkItemCreatedAfterReceivedAction	✔ OK
TaskTests.TaskTests - test_stopAction_workItemCancelledAfterStopCalled	✔ OK
TaskTests.TaskTests - test_receivedUserAction_onTimeExceededCountCorrect	✔ OK
TaskTests.TaskTests - test_workItem_hasBeenCreated	✔ OK
TaskTests.TaskTests - test_startAction_actionBlockCalledAfterGivenSessionTime	✔ OK
TaskTests.TaskTests - test_weakSelfInActionBlock	✔ OK
TaskTests.TaskTests - test_startAction_workItemCreatedAfterStart	✔ OK
TaskTests.TaskTests - test_workItem_actionBlockHasBeenPassed	✔ OK
TaskTests.TaskTests - test_stopAction_workItemCancelledAfterStopCalledCase2	✔ OK
TaskTests.TaskTests - test_startAction_actionBlockHasBeenPassed	✔ OK
TaskTests.TaskTests - test_stopAction_onTimeExceededNotCalledAfterStop	✔ OK
TaskTests.TaskTests - test_correctlyExecutedWorkItemWhenStartCalledTwoTimes	✔ OK
TaskTests.TaskTests - test_startAction_workItemExecutedOnCorrectQueue	✔ OK
TaskTests.TaskTests - test_startAction_onTimeExceededCalledAfterInitialSessionTime	✔ OK
TaskTests.TaskTests - test_startAction_actionBlockCalledAfterGivenSessionTimeCase2	✔ OK
TaskTests.TaskTests - test_initialState_onTimeExceededNotCalledRightAfterStartAndBefore	✔ OK
TaskTests.TaskTests - test_receivedUserAction_firstWorkItemShouldBeCancelled	✔ OK
TaskTests.TaskTests - test_initialState_didNotCreateAWorkItemBeforeStart	✔ OK