IvanCaldwell / **ios-projects**
forked from LambdaSchool/ios-projects

Branch: **master** ▾ **ios-projects** / Sprint 7 / **Generics (7.2)** / Create new file | Upload files | Find file | History

This branch is 6 commits ahead of LambdaSchool:master. Pull request | Compare

armadsen Add README for 7.3 Concurrency II - Fundamentals Latest commit 85773da on Sep 4, 2018

..

README.md Add README for 7.3 Concurrency II - Fundamentals 5 months ago

README.md

Generics Challenge

Create a generic `CountedSet` struct that is constrained to `Hashable` elements. A counted set is an unordered collection of unique elements that may appear more than once in the collection. Use a private dictionary as your backing storage for set members and their counts.

Set up the project

Create a new playground or project to develop and test your type. It may be easier to develop a single type in a playground but Xcode is a bit unstable with playgrounds these days. Use whatever tool best works for you.

Goals

- Add insertion and removal (`insert` and `remove`) of one element at a time.
- Support subscripting to look up current values (by implementing `subscript(_ member: Element) -> Int`). Return `0` for any value that is not found.
- Add `count` , returning the number of unique elements in the counted set and `isEmpty` for when `count` is zero.

Conform to `ExpressibleByArrayLiteral`

As demonstrated in class, conform your set to `ExpressibleByArrayLiteral` so you can initialize a counted set using an array of same-type items.

Your implementation should support the following interaction style:

```
enum Arrow { case iron, wooden, elven, dwarvish, magic, silver }
var aCountedSet = CountedSet<Arrow>()
aCountedSet[.iron] // 0
var myCountedSet: CountedSet<Arrow> = [.iron, .magic, .iron, .silver, .iron, .iron]
myCountedSet[.iron] // 4
myCountedSet.remove(.iron) // 3
myCountedSet.remove(.dwarvish) // 0
myCountedSet.remove(.magic) // 0
```

Test

1. Run the project and make sure everything works. Create a good suite of tests that check for boundary conditions and many different types.
2. If anything doesn't work the way the above example shows, go back and debug your issues.
3. As always, if you need help, follow the 20-minute rule, then ask your PM.

Go Farther

Time allowing, consider adding the following enhancements to your `CountedSet` type:

- Conform `CountedSet` to `Sequence` by creating a custom iterator (`DictionaryIterator`).
- Implement `contains` to test whether your set contains at least one of a given item.
- Implement `union` with another set, returning all members and their combined sums. For extra karma, implement a mutating version (change in place) and a non-mutating version (return a copy).
- Implement `intersection` and/or `subtraction` using the same logic. Intersection returns all members and counts appearing in both sets. Subtraction removes all counts found in the second set from the first.
- Implement `isDisjoint` , testing that there's no overlap between elements.
- Conform to `Equatable` and implement `==` to test if two counted sets are the same.

References

- Swift Generics WWDC 2018: https://developer.apple.com/videos/play/wwdc2018/406/
- https://docs.swift.org/swift-book/LanguageGuide/Generics.html
- Search the web for: *swift generics*