

Swift Programlama Dili ile iOS Mobil Uygulama Geliřtirme

Closure Yapısı ile Koleksiyon İşlemleri

Closure Yapısı ile Koleksiyon İşlemleri

Egzersizler

1. **String** olarak bazı isimler içeren **names** adlı sabit bir dizi oluşturun. Diziyi üç adetten fazla isim içerecek şekilde ilklendirin. Şimdi dizideki tüm adların birleşimi olan bir **String** oluşturmak için **reduce** fonksiyonunu kullanın.
2. Aynı **names** dizisini kullanarak, önce diziyi yalnızca dört karakterden uzun isimler içerecek şekilde filtreleyin ve ardından yukarıdaki egzersizde olduğu gibi aynı isim birleşimini oluşturun. (İpucu: Bu işlemleri birbirine zincirleyebilirsiniz.)

Closure Yapısı ile Koleksiyon İşlemleri

Egzersizler

3. **String** olarak adların **Int** olarak yaşlara eşlenen **namesAndAges** adlı sabit bir sözlük oluşturun. Şimdi sadece 18 yaşın altındaki kişileri içeren bir sözlük oluşturmak için filtrelemeyi kullanın.
4. Aynı **namesAndAges** sözlüğünü kullanarak, yetişkinleri (18 yaş ve üstü) filtreleyin ve ardından sadece isimleri içeren bir diziye dönüştürmek için **map** fonksiyonunu kullanın.

Closure Yapısı ile Koleksiyon İşlemleri

Egzersizler

5. Bir closure'ı belirli sayıda çalıştıracak bir fonksiyon yazın. Fonksiyonun imzası şu şekildedir:

```
func repeatTask(times: Int, task: () -> Void)
```

Closure Yapısı ile Koleksiyon İşlemleri

Egzersizler

6. Farklı matematiksel toplamlar oluşturmak için yeniden kullanabileceğiniz bir fonksiyon yazın. Fonksiyonun imzası şu şekildedir:

```
func mathSum(length: Int, series: (Int) -> Int) -> Int
```

İlk parametre olan **length**, toplanacak değerlerin sayısını tanımlar. İkinci parametre olan **series**, bir dizi değer oluşturmak için kullanılabilen bir closure'dır. **series** closure'unun, serideki değerlerin konumu olan bir parametresi olmalı ve bu konumdaki değeri döndürmelidir.

mathSum, 1. konumdan başlayarak **length**'e kadar olan değerleri **series** closure'u ile hesaplamalı ve toplam değeri dönmelidir.

Fonksiyonu, ilk 10 sayının karelerinin toplamı olan 385'i bulmak için kullanın.

Ardından, ilk 10 Fibonacci sayısının toplamı olan 143'ü bulmak için bu fonksiyonu kullanın. Burada fibonacci sayıları için, önceden yazdığımız fonksiyonu kullanabilirsiniz.

Closure Yapısı ile Koleksiyon İşlemleri

Egzersizler

7. Elinizde aşağıdaki gibi uygulamalar ve bu uygulamalara verilmiş derecelendirmelerin olduğu bir sözlüğünüz var.

```
let appRatings = [  
  "Calendar Pro": [1, 5, 5, 4, 2, 1, 5, 4],  
  "The Messenger": [5, 4, 2, 5, 4, 1, 1, 2],  
  "Socialise": [2, 1, 2, 2, 1, 2, 4, 2]  
]
```

İlk olarak, uygulama adlarının ortalama derecelendirmeler ile bir eşlemesini içerecek **averageRatings** adlı bir sözlük oluşturun. **appRatings** sözlüğünü yinelemek için **forEach**'i kullanın, ardından ortalama puanı hesaplamak için **reduce** fonksiyonunu kullanın. Bu derecelendirmeyi **averageRatings** sözlüğünde saklayın. Son olarak, ortalama puanı 3'ten büyük olan uygulama adlarının bir listesini almak için **filter** ve **map** fonksiyonlarını birlikte kullanın.

Closure Yapısı ile Koleksiyon İşlemleri

Özet

1. Closure'lar, isimsiz fonksiyonlardır. Değişkenlere atanabilir ve fonksiyonlara parametre olarak aktarılabilirler.
2. Closure'lar, kullanımını diğer fonksiyonlara göre çok daha kolay hale getiren kısa söz dizimine sahiptir.
3. Bir closure, değişkenleri ve sabitleri çevreleyen kapsamdan yakalayabilir.
4. Bir koleksiyonun nasıl sıralanacağını yönlendirmek için bir closure kullanılabilir.

Closure Yapısı ile Koleksiyon İşlemleri

Özet

5. Koleksiyonlarda, bir koleksiyonu yinelemek ve dönüştürmek için kullanabileceğiniz kullanışlı bir dizi fonksiyon vardır. Dönüşümler, her bir öğeyi yeni bir değere eşlemeyi (**map**), belirli değerleri filtrelemeyi (**filter**) ve koleksiyondaki değerleri tek bir değere indirgemeyi (**reduce**) içerir.
6. Lazy koleksiyonlar, bir koleksiyonu yalnızca kesinlikle ihtiyaç duyulduğunda elde etmek için kullanılabilir; bu, büyük, pahalı veya potansiyel olarak sonsuz koleksiyonlarla kolaylıkla çalışabileceğiniz anlamına gelir.

String'ler

String'ler

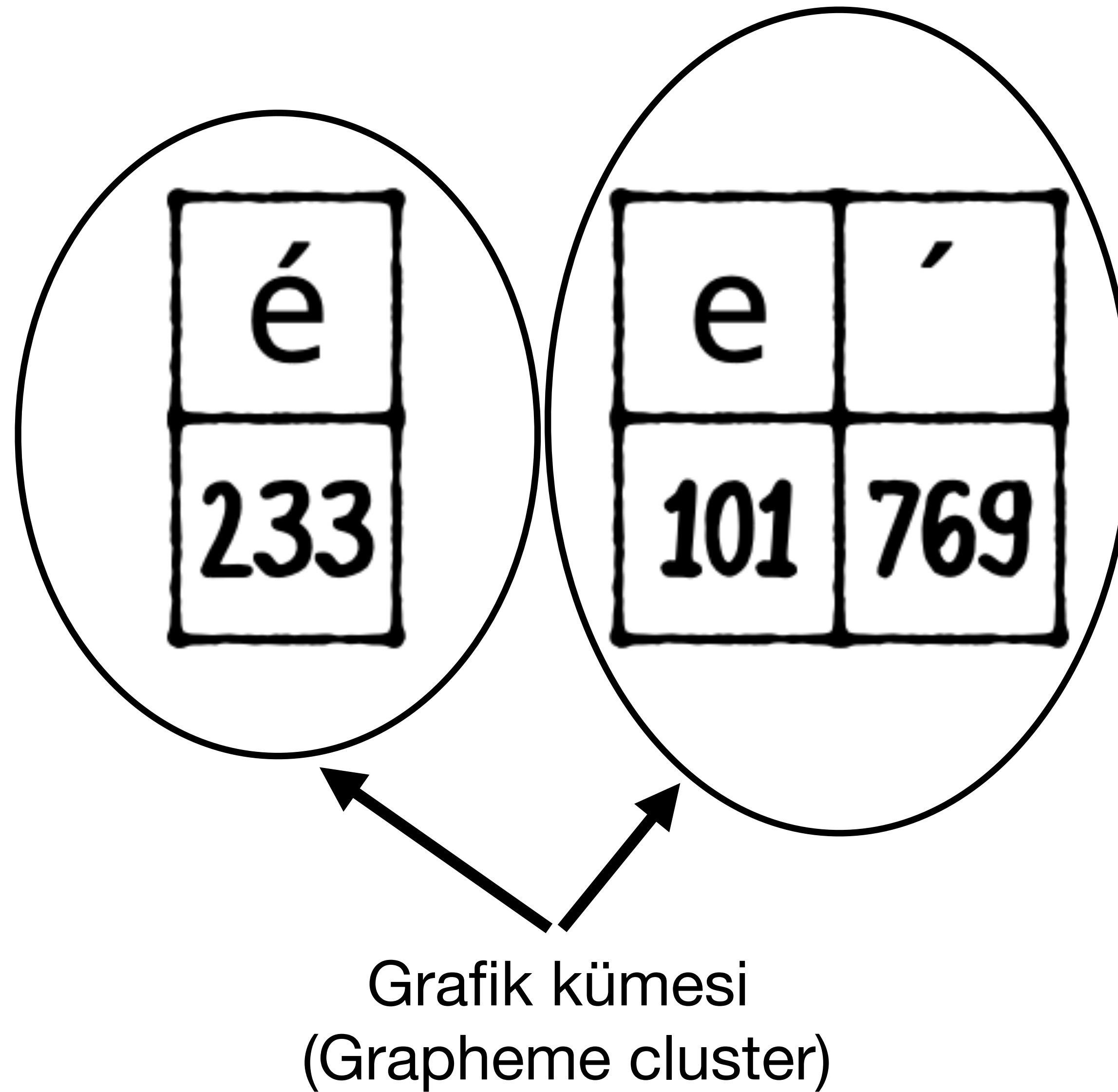
café

String'ler

é
233

e	'
101	769

String'ler





String'ler

Grafik kümesi
(Grapheme cluster)



Character

String'ler

	
128077	127997



String'ler

c	a	f	é
99	97	102	233

c	a	f	e	'
99	97	102	101	769

String'ler

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]