4. Java'da heap ve stack kavramlarını örneklerle açıklayın.(5 PUAN)

Java Programlama Dilinde Değer ve Referans Tipler Hakkında

Hepinize merhaba arkadaşlar. Bu yazımızda her programlama dilinin temel konularından bir tanesi olan veri tiplerinden bahsedeceğiz. Bunun için öncelikle değişken tanımını ele almamız gerekmektedir. Programlama yapılırken verilen bir değeri bilgisayarın hafızasında tutan ve ihtiyaç duyulduğunda kullanılmasına imkan veren yapılar değişken olarak adlandırılır. Veri tipi ise tanımlanan değişkenin hangi veri türünde saklanacağını belirtmek için kullanılır. Kullanacağımız değişkenler tam sayı, kesirli sayı, metin, karakter veya doğru, yanlış gibi mantıksal ifadeler olabilirler. Değişkenler, atandıkları veri tipine bağlı olarak bellekte farklı biçimlerde ve farklı boyutlarda tutulurlar.

Veri tiplerini bellek davranışlarına göre iki farklı alanda sınıflayabiliriz.

1. Değer Tip(Value Type)
2. Referans Tip(Referance Type)

Değişkenler yani veriler RAM (Random Access Memory—Rasgele Erişimli Hafıza) üzerinde geçici olarak saklanır. RAM içerisinde stack(yığın) ve heap(öbek) isimli iki farklı davranış şekline sahip mantıksal bölüm bulunmaktadır. Değer tipindeki veri tipinde bir değişken tanımlandığında ve bu değişkene bir değer atandığında hem değişken hem de değeri belleğin stack kısmında tutulur. Referans veri tipinde bir değişkene değer atandığında ise bu değer belleğin heap bölgesinde adres değeri ise stack bölgesinde tutulmaktadır.

Değer Tipler

İlkel(primitive) veri tipleri olarak da adlandırılırlar. Bu tip değişkenler her zaman tek bir değere sahiptirler(metin, sayı vb.) Java'da değişken tanımlama formülü "<veri tipi> <değişken ismi> = veri (değer)" şeklindedir. Öncelikle tanımlanacak değişkenin veri tipi belirlenir daha sonra değişken isimlendirmesi yapılır. Eğer tanımlama sırasında ilk değer atanacaksa eşittir işareti kullanılarak veri tipine uygun bir değer yazılır. Böylece hem değişken tanımlama hem de değer ataması yapılmış olur. Değer ataması yapılmayan değişkenler bellek üzerinde yer kaplamaz.

	Veri Tipi	Anahtar Sözcük	Bellek Hacmi	Değer Aralığı
Tam Sayı Türünde Veri Tipleri	Byte	"byte"	1 byte (8 bit)	-128 ile 127
	Short	"short"	2 byte (16 bit)	-32768 ile 32767
	Integer	"int"	4 byte (32 bit)	-2^{31} ile $2^{31}-1$
	Long	"long"	8 byte (64 bit)	$-1.*10^{38}$ ile $-3.4*10^{38}$
Ondalıklı Sayı Türünde Veri Tipleri	Float	"float"	4 byte (32 bit)	1.4×10^{-45} ile 3.4×10^{38}
	Double	"double"	8 byte (64 bit)	4.9×10^{-324} ile 1.8×10^{308}
Karakter	Char	"char"	2 byte (16 bit)	0 ile 65535
Mantıksal	Boolean	"boolean"	(1 bit)	True & False

Değer Tipli Değişken Bilgileri

```
int sayi1; //Henüz değeri atanmamış integer veri tipinde sayi1 isimli değişkeni ifade eder.
int sayi2=5; //Değeri 5 olarak atanmış integer veri tipinde sayi isimli değişkeni ifade eder.
```

Java'da Değişken Tanımlama

Değer tip değişkenler program içerisinde çağırıldığında işlemler değişkenin stack üzerinde tanımlanan değeri üzerinden gerçekleşir. Ufak bir örnek ile değer tiplerin belleğin stack bölümdeki davranışına bir göz atalım.

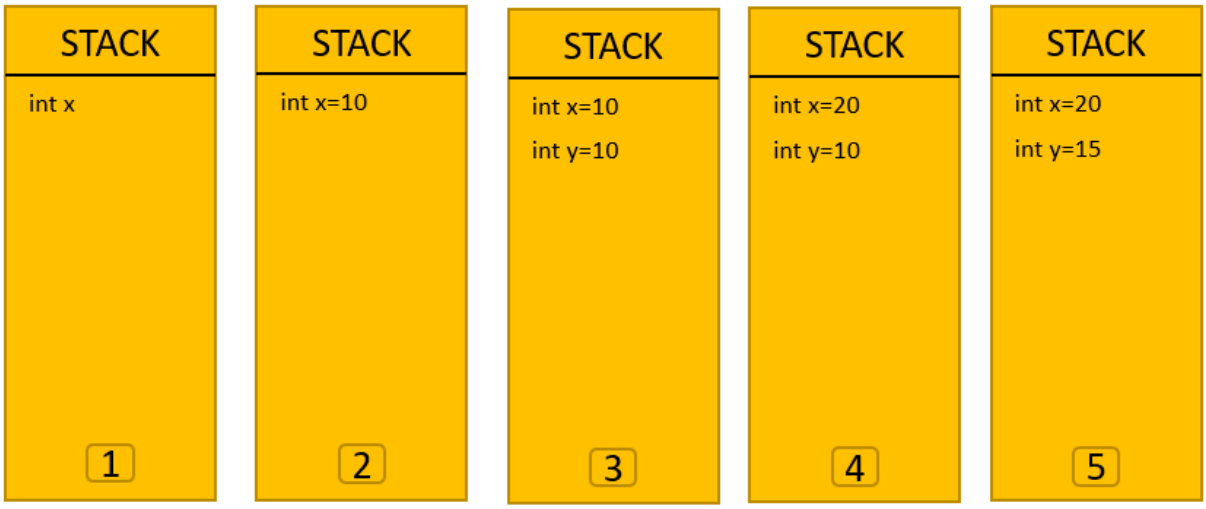
```
int x; /* Integer veri tipinde x isimli bir değişken tanımlandı.
1 numaralı görsel stack üzerinde bu tanımlamayı göstermektedir.*/

x=10; /* Tanımlanan x değişkenine 10 değeri atanır. Değişkenin veri tipi sadece
ilk tanımlama sırasında belirtilir. Aksi durumda program yeni bir değişken tanımlamayı
istediğimizi düşünerek hata verecektir. Çünkü bir değişken ismi sadece bir değişken
tanımlamasında kullanılabilir. 2 numaralı stack görseli belleğin işlem sonrası
durumunu temsil eder. */

int y=x; /* İlk değeri x'in mevcut değeri (10) ile aynı olan integer tipinde y isimli
bir değişken tanımlandı. Yapılan bu tanımlamanın "int y=10" tanımlamasından hiç bir
farkı bulunmamaktadır. y değişkenine x'in o anki değeri atanır ve x değişkeni ile
y değişkeni arasında bunun dışında herhangi bir bağlantı oluşmaz. 3 numaralı stack
görseli belleğin işlem sonrası durumunu temsil eder. */

x=20; /* x'in yeni değeri 20 olarak atanır. Aralarında herhangi bir ilişki bulunmadığından
y değişkeni bu değişiklikten etkilenmez. 4 numaralı stack görseli bellğin işlem sonrası
durumunu temsil eder. */

y=15; /* y'nin yeni değeri 15 olarak atanır. x değişkeninin değeri bu değişiklikten
etkilenmez. 5 numaralı stack görseli belleğin işlem sonrası durumunu temsil eder. */
```



Değer Tipli Değişkenlerin Bellek (Stack) Davranışı

Görüldüğü üzere bir değer tip olan integer ile yapılan bir değişken tanımlaması ve bu değişkene yapılan bütün değer atamaları stack bellek üzerinde gerçekleşir ve tutulur.

Referans Tipler

Temelde 3 adet referans tip mevcuttur. Bunlar Sınıf(class), Dizi(array), ve Arayüz(interface)'lerdir. Referans tipler değer tiplerden farklı olarak birden fazla değeri tutabilirler. Ayrıca değeri atanmış bir referans tip belleğin hem heap hem de stack bölümünde yer kaplar.

Sınıf(class) Yapısı:

Kendisinden üretilecek nesneler için şablon görevi gören kısmen soyut yapılardır. Nesnelerin sahip olacağı ortak özellikleri ve davranışları(method) barındırırlar. Birbirleri ile ilişkisi olan

verileri bir arada tutarak tek bir referans ile bu verilere erişim sağlanmasına yararlar. “Class” anahtar sözcüğü kullanılarak oluşturulurlar.

Arayüz(interface) Yapısı:

Arayüz farklı sınıflardan nesnelerin birbirleri arasında kategorize edilmesini sağlayan bir soyut tür çeşididir. Daha çok birer sözleşme gibi düşünülebilirler. Method imzası belirtilmiş methodlar barındırırlar. Bu arayüzleri kullanan sınıflar arayüzde tanımlanan methodları kendi bünyelerinde barındırmak zorundadırlar.

Dizi(array) Yapısı:

Aynı veri tipinden birden fazla veriyi bir arada tutmaya yarayan nesne yapılarını ifade eder. Örneğin bir sınıftaki öğrencilerin bir sınavdan almış oldukları notları tek bir değişkende tutmak istersek dizilerden yararlanabiliriz.

“veriTipi[] <diziAdi> = new veriTipi[elemanSayisi]” kod bloğu ile yeni bir dizi tanımlayabiliriz.

Referans tipli değişken tanımlama ve bu değişkene değer atanmasını aşağıdaki örnek ile inceleyelim.

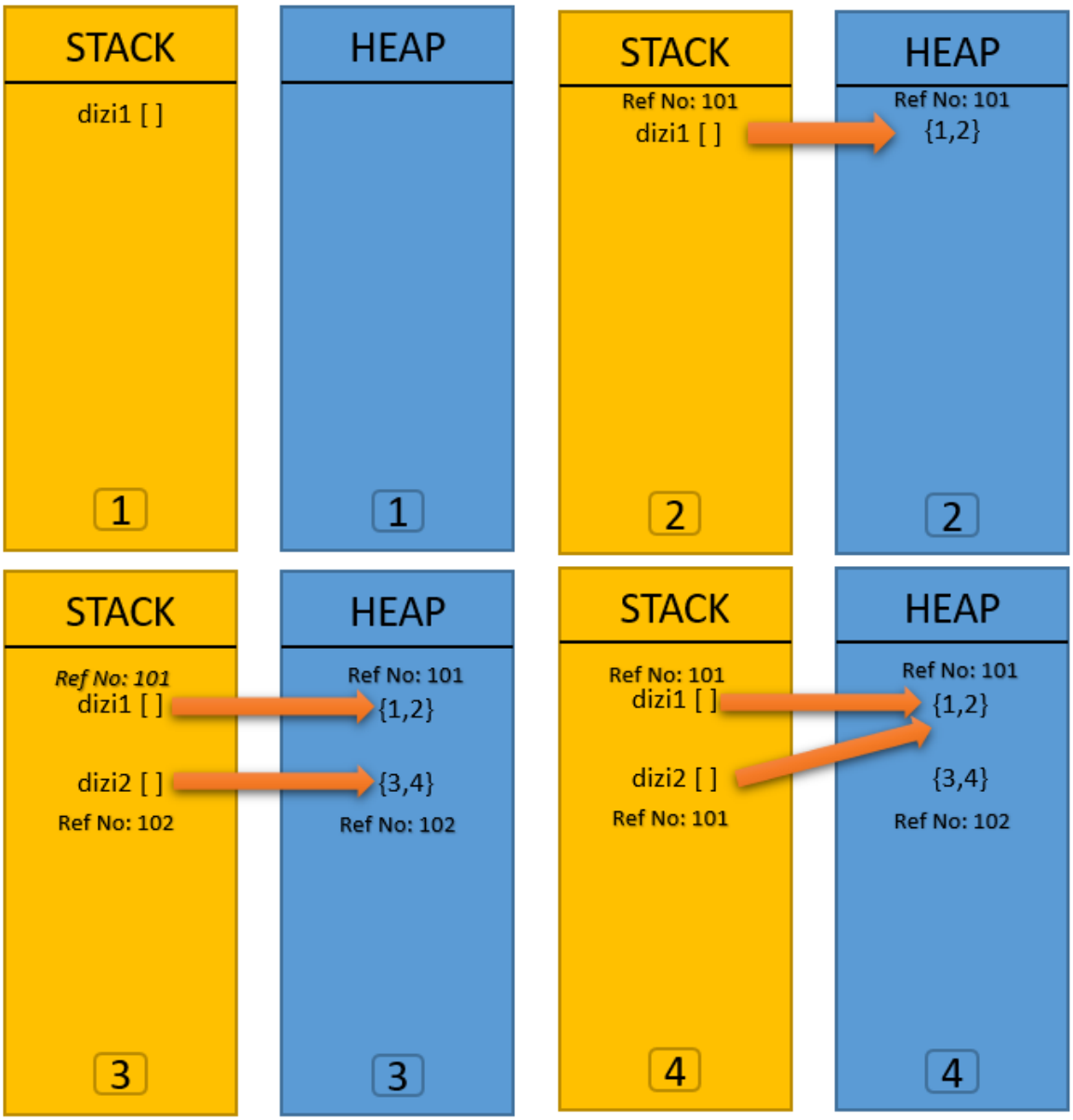
```
int[] dizi1; //integer tipli verileri tutabilecek dizi1 isimli dizi referansı stack bellekte tanımlandı.
//"1 numaralı görsel"

dizi1=new int[] {1,2}; /* 1 ve 2 değerlerini içeren dizi1 referansı heapte tanımlandı. Stack ve Heap'de
aynı referans numarası ile tutulurlar. (Örnek: 101) "2 numaralı görsel"*/

int[] dizi2=new int[] {3,4}; /* dizi2 isimli integer dizisi stack bellekte tanımlandı. 3 ve 4 değerlerini
içeren dizi2 referansı heapte tanımland. Stack ve Heap'de aynı referans numarası ile tutulurlar. (Örnek 102)
"3 numaralı görsel"*/

dizi2=dizi1; /* dizi2'nin referansı eşittir dizi1'in referansı şeklinde okunur. Artık hem dizi1 değişkeni
hem de dizi 2 değişkeni 101 numaralı referansı ({1,2}) tutmaktadır. "4 numaralı görsel */

/* Program içerisinde 101 numaralı referansa ulaşmak istediğimizde hem dizi1 hem de dizi2 değişkeni kullanabilir.
 her iki değişkeni kullanarak yapacağımız değişiklikler 101 numaralı referans üzerinde gerçekleşecektir.*/
```



Referans Tipli Değişkenlerin Bellek (Stack-Heap) Davranışı

Şafak Akın