**Java ile veri yapıları**

Veri yapıları 3 gruba ayrılır:

|  |  |  |
| --- | --- | --- |
| Temel (İlkel) | Basit | Bileşik |
| İnt  Float  Char  Double  … | Diziler  Yapılar  Enum  …. | Bağlı listeler (Linked List)  Kuyruk (queue)  Yığınlar (Stack)  Ağaçlar (Treas)  …. |

***Bağlı listeler***

Düğüm oluşturma nedir?

(Dizilerle kıyaslanarak anlatılmaktadır)

İnt sayi;

* bu tek değişkendir ve bellekte 1 bayt’lık yer kaplar.

İnt dizi[10];

* bu bellekte 10 adet elemana yer ayıracak şekilde alan kaplar.
* Biz bu elemanları kullansak da kullanmasak da bellekte yer kaplar

Bağlı listelerde düğümler var!

Biz bu düğümleri birbirine bağlıyoruz, oluşturuyoruz ya da siliyoruz

Düğüm denildiği zaman bunu düşüneceğiz.

İhtiyacımız oldukça işletim sistemi bize bellekte düğümler oluşturup, bu oluşturduğu kısmı başlangıç adresi verecek.

İhtiyacımız bitince düğümü kapatıp bellekteki ayırdığı kısmı belleğe iade edecek.

İnt

Float

Double

String

Char

…

Düğümün Büyüklüğü içerisinde barındırdığı değişkene göre değişmektedir

Bu düğümün içerisinde int, float, double, string, char… gibi değişkenler olabilir

Düğüm içerisinden birbirinden farklı ve birden fazla değişkeni barındırabilir.

Aslında düğüm = obje (Nesne) Java’da Nesne dediğimiz şeyi Class’lardan oluşturuyoruz.

Yani Class’lardan oluşturduğumuz objelere düğüm diyoruz.

Örnek:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturulduBiz bir class’tan bir nesne oluşturup nesneye değer vermediğimizde düğüm oluşmuyor bellekte null değeri tutuluyor fakat bir class’tan bir nesneyi new’leyip türetirsek düğüm oluşturmuş oluyoruz.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Class’tan new ile kaç tane nesne türetirsek o sayıda düğümümüz oluyor.

Aşağıdaki fotoğraf **main** Class’ında yazılmıştır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Türettiğimiz nesneye değer vererek düğümlere değer vermiş oluyoruz.

Türettiğimiz düğümleri birbirine bağlamak için “Pointer” adı verilen yapılar kullanılır. Grafiksel karşılığı aşağıdaki gibidir.

Pointer

Pointer

Pointer

Null

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Burada next adındaki nesne göstericidir. Yani bir sonraki düğümün adresini tutacak olan pointer’dır. Burada önemli olan nokta şu; örneğin biz burada ClassOne sınıfından türettiğimiz düğümleri birbirlerine bağlayacağız, yani türetilen nesnelerin tipleri ClassOne tipindedir. ClassOne tipinde olan düğümleri yine ClassOne tipinde olan bir pointer bağlayabilir. Bu pointer’ı da ClassOne sınıfı içinde tanımlıyoruz.

Düğümlerin içinde gezmek

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Linked List (Bağlı Listeler)**

Bağlı listeleri bir zincir gibi düşüneceğiz ve zincirin halkalarını düğüm olarak göreceğiz.

Bağlı listeler 4’e ayrılır:

1. Tek yönlü doğrusal bağlı liste
2. Çift yönlü doğrusal bağlı liste
3. Tek yönlü dairesel bağlı liste
4. Çift yönlü dairesel bağlı liste

diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

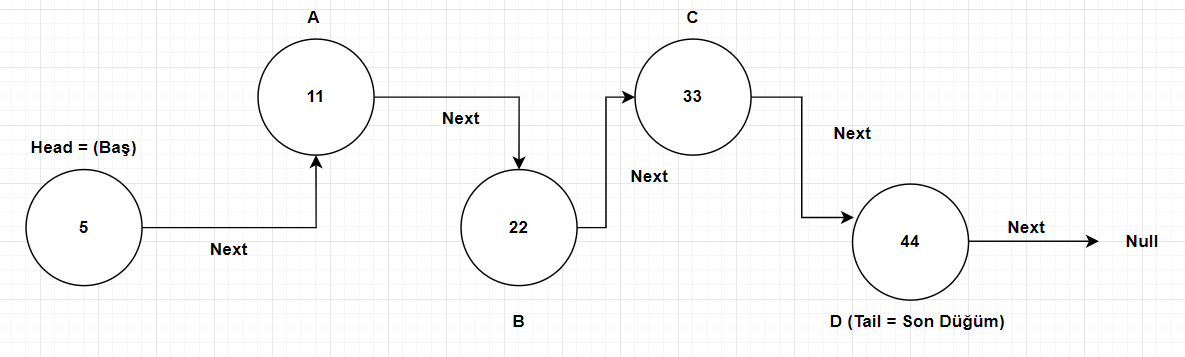
Tek yönlü doğrusal bağlı listenin yapısı yukarıdaki gibidir.

* En önemli düğüm ilke düğümdür.
  + İlk düğümün adı HEAD’dir
* İkinci bir düğüm oluşturursak HEAD’e bağlı olacak
* Eğer bir düğüm “Null” değerini gösteriyorsa o düğüm son düğümdür

Tek yönlü doğrusal bağlı listenin özellikleri

* Head düğümünden başlayıp sırayla Tail düğümüne kadar tüm düğümleri belli bir sıra ile geçiş yapılması
* Tail’in null değerini gösteriyor olması

Eğer biz farklı bir düğümü başa getirip, başa getirdiğimiz düğümü Head düğüm yaparsak yapısı aşağıdaki gibi olacaktır.



Eğer Tail düğümünden sonra bir düğüm eklersek yapısı aşağıdaki gibi olacaktır

metin, iç mekan içeren bir resim

Açıklama otomatik olarak oluşturuldu