VERİ YAPILARI SİPARİŞ PROJESİ

Hazırlayanlar:

Zeynep Verda ASLAN 220303010

Kader YAPRAK 220303064

Caner BULUT 220303012

Özet:

Projemiz bir “yönetim sipariş sistemi” oluşturmaktı. Ağaç veri yapısı yardımıyla sipariş verilerini tutup, yönetmeyi planladık. Yazdığımız kodlar,

sipariş ekleme, iptal etme, belirli ürün kümesi sorgulama ve yapıyı hiyerarşik olarak yazdırma gibi temel işlevleri desteklemektedir.

Projemiz, 3 sınıftan oluşuyor: TreeNode, SiparisYonetim ve Main Class.

TreeNode classı: ağacın kök de dahil olmak üzere tüm düğümlerinin veri tipidir.

SiparisYonetim: ekleme, silme ve arama işlemlerini içeren sınıf.

Main: siparis yönetim classının işlemlerini uygulama kısmı.

Projeyi yaparken Java programlama dilini nesne yönelimli yapısından yararlandık.

Projeyi nasıl yaptık?

1. İlk başta bir ağaç veri yapısı oluşturduk.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

String data: düğümün içindeki veriyi tutar.

Count: sipariş sayısını tutar.

Map<String, TreeNode> çocuk: alt düğümleri tutan bir map yapısıdır.

TreeNode ebeveyn: yukarıya doğru erişim için kullanılır. Çocuk düğümünün atasıdır.

boolean kontrol: siparişin bitip, bitmediğini anlamak için kullanıyoruz.

1. İşlemlerimizi gerçekleştireceğimiz sipariş yönetim classını açtık.

Yapıcı method oluştururken bir ağaç tanımladık. Her nesne oluşturulduğunda ağacın kökü oluşturuluyor.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. ‘addOrder’ methodu siparişleri alır, dizi haline atayarak girdileri küçük harfe dönüştüren ve kodda boşluklar varsa hata fırlatan ‘siparisler’ methoduna gönderir( ilk satır).

Bu method da en sonunda siparişleri ‘siralama’ methoduna atar.

‘siralama’ methodu siparişteki bütün ürünleri karşılaştırır ve Unicode’a göre sıralamasını yapar.

İlk ürün karşılaştırılan üründen sonra geliyorsa pozitif bir değer döner, ürünler yer değiştirilir.

İlk ürün, ikinci üründen daha önce geliyorsa negatif değer döner ve zaten alfabetik sıraya göredir, işlem yapılmaz.

Karşılaştırılan iki ürün de alfabetik olarak eşitse sıfır döner ve yine bir işlem yapılmaz.

metin, yazı tipi, beyaz, tipografi içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, yazı tipi, ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

addOrder methodunun alt satırlarında ise ilk başta ürünlerin aynı olup, olma dığını test eder. Aynıysa alttaki cümleyi yazdırır.



Bir treeNode değişkeni oluşturuyoruz ve üstte oluşturduğumuz root değişkenine atıyoruz.

Ürünlerin hepsini tek tek dönerek, düğümün alt düğümlerine eğer ekli değilse, yeni bir düğüm oluşturup ekler.

Yeni eklenen ya da var olan düğümü(part) alır, alt çocuğa atar.

Alt çocuğun ebeveynini ne üstteki düğüme atar.

Sipariş eklendiğinden dolayı kontrol: true olur.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. Silme işlemi için ‘deleteOrder’ methodunu kullandık.

Daha önce ‘addOrder’ methodunda olduğu gibi ilk başta ‘siparisler’ methoduna gitti. Orada boşluklardan arınmış olan girdi, ‘siralama’ methoduna gidip alfabetik sıraya uygun hale geldi.

Girdiler kontrol edildi. Eğer ‘null’ giriş varsa hata mesajı yazdırıldı.

Silinme yapılamadı, false değer döndü.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

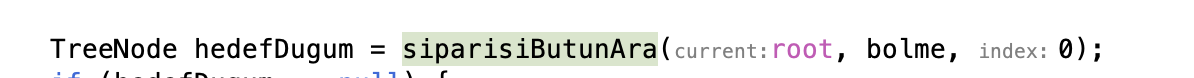
Siparişin tamamını kaldırmak amacıyla ‘siparisiButunAra’ methodunu yazdık.

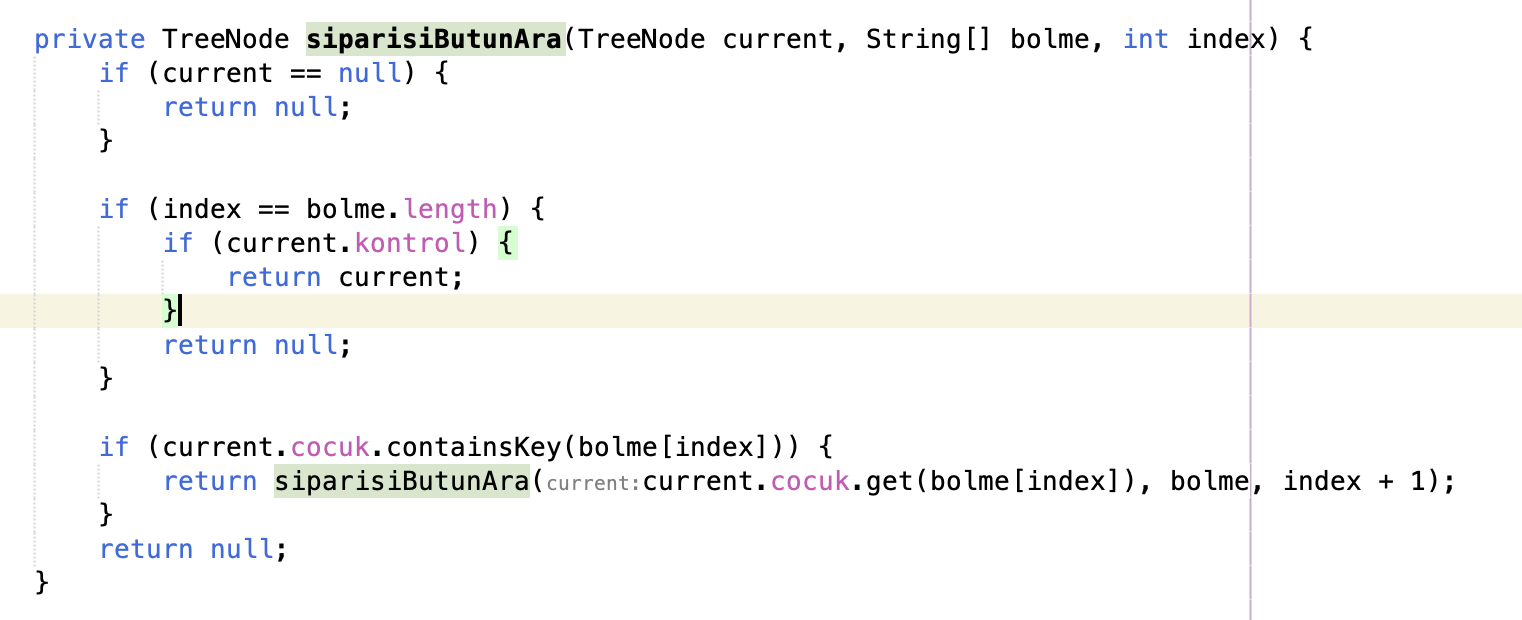
Bu method da aradığımız yol ağaçta değilse arama işlemi sonlandırılıyor.

Kontrol true çıkarsa, bu düğüm son siparişin düğümü olduğu için current döndürülüyor. Var olmayan bir sipariş yolu ise null

döndürüyor.

Sonrasında ‘bolme[index]’e karşılık gelen bir alt düğüm olup olmadığı kontrol ediliyor. Eğer alt düğüm varsa özyinelemeli olarak bir sonarki eleman çağırılır. Alt düğüm yoksa işlem durur, null döner.





Sipariş yolu varsa yeni bir treeNode’a atanıyor ardından kontrollere geçiliyor. Eğer sipariş bulunamadıysa ‘sipariş bulunamadı’ yazdırıyor.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Eğer sipariş varsa ‘deleteOrderRecursive’ methoduna girdi olarak giriyor.

metin, yazı tipi, ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

‘deleteOrderRecursive’ methodu ise düğümün alt düğümü olup olmadığını kontol eder, alt düğümü yoksa ebeveyn düğümü ‘parent’ a atıyoruz. Hedef düğüm silindikten sonra ebeveynin alt düğümleri güncellenmiş oluyor.

Eğer hedef düğümün ebeveyni varsa, çocuk hedef düğüm siliniyor. Artık parent ile hedef düğümün ilişkisi kalmıyor.

Parent sayısı azaltılıyor.

Parent düğümün alt düğümü kalmadıysa, kontrol false olur.

Eğer kontrol false olursa, özyinelemeli olarak silinir.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Son olarak hiçbir ‘if’ e girmezse “sipariş silinemedi” yazdırılır, ve false döndürülür.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. Sipariş aramak amacıyla ‘searchOrder’ methodunu yazdık.

Girdiyi diziye ayırmak amaçlı ‘siparisler’ methoduna gönderildi.

Girdideki boşluk silindi, küçük harfe dönüştürüldü. ‘siralama’ methoduna gönderildi. Burada da alfabetik sıraya göre siparişler dizildi.

metin, yazı tipi, ekran görüntüsü, beyaz içeren bir resim

Açıklama otomatik olarak oluşturuldu

Ardından kökten başlayarak arama işlemi yapılır.

Her bir düğümün çocukları ve çocuğun alt düğümleri alınır. Alt düğümden herhangi bir part null dönerse method false döndürür.

Eğer tüm parçalar ağacın içinde bulunursa, sipariş mevcut demektir, true döner.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. ‘ürünKümesiSorgulama ‘ kullanıcıdan alınan ürünlerin yollarını bulur.

Eğer boş bir veri seti girildiyse ‘Siparis kumesi bos olamaz’ yazdırılır.

metin, yazı tipi, ekran görüntüsü, çizgi içeren bir resim

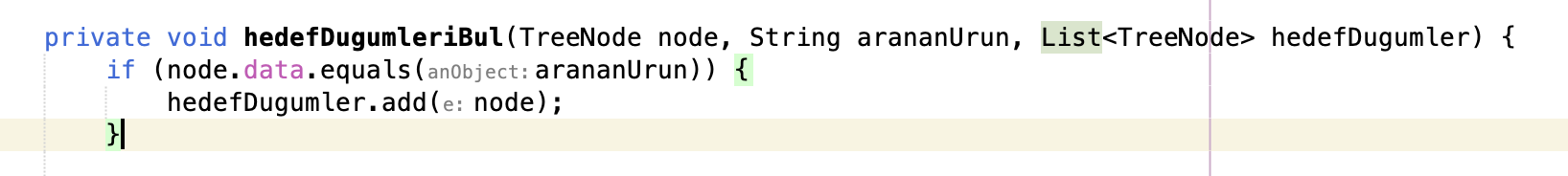
Açıklama otomatik olarak oluşturuldu

Liste sonuna eklenen ürün alınır, ağaç içinde listeSonuÜrün ile eşleşen tüm düğümleri(hedef düğümler) toplar ve ‘hedefDugumleriBul’ methoduna gönderilir.

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

‘hedefDugumleriBul’ methodu mevcut düğüm(node), verisi (data ), aranan ürün ile eşleşiyor mu ona bakılıyor. Eşleşme varsa node, hedef düğümlere atanıyor.



Mevcut düğümün, tüm alt düğümleri arasında gezilir. Her bir düğüme uygulamak amaçlı özyinelemeli olarak aynı işlem uygulanır. İşlem sonunda tüm dallar taranmış olur.

metin, yazı tipi, ekran görüntüsü, beyaz içeren bir resim

Açıklama otomatik olarak oluşturuldu

Tüm dallar tarandıktan sonra ‘ürünKümesiSorgulama ‘methoduna geçer.

Uygun yolları tutmak için bir liste yapısı kurulur.metin, yazı tipi, ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Düğümler(hedefDugumler) döngüye girer. Ve ‘siparisYoluUygunMu’ methoduna gider.

metin, yazı tipi, çizgi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

‘siparisYoluUygunMu’ methodunda ise sipariş methodunun son elamanının indeksini alır. Kök düğüme ulaşana kadar düğümleri kontrol eder.

Mevcut düğümün datası, siparişdeki eleman ile eşleşiyor mu? Eşleşme varsa index bir azaltılır.

Mevcut düğümden ebeveyn düğüme geçiş yapılır. İndeks 0’ın altına düştüyse tüm elemanlar doğrulanmış demektir.metin, yazı tipi, ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Verilen sipariş kümesiyle, yol birbirine eşleştiyse ‘toplamSiparisSayisi’ artırılır. metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Ardından ‘yoluOlustur’ methoduna geçeriz.

Düğüm yolunu listede tutarız. Düğüm verisi null olana kadar ‘yol’ a node.data eklenir. Sonra node un ebeveyni node a atanır. Böyle sondan başa doğru çıkılır.

Sonrasında yol string e dönüştürülür. ‘--> ‘ şekli konsolda gözükür.metin, yazı tipi, çizgi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Daha sonrasında’ ürünKümesiSorgulama’ methoduna geri dönerek, bulunan kodlar ve toplam sipariş sayısı yazdırılır.metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. ‘agaciYazdir’ methodu da ‘printTreeRecursive’ methodunu çağırır.metin, yazı tipi, ekran görüntüsü, çizgi içeren bir resim

   Açıklama otomatik olarak oluşturuldu

‘printTreeRecursive’ methodu da ağaç veri yapısını ekrana yazdırır.

Node null ise işlem yapılamaz.

Eğer node null değilse ve root ise konsola yazdırılır. Görselleştirme için semboller eklenir.metin, yazı tipi, çizgi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Node root değilse node’un verisi yazdırılır. Count da parantez içinde gösterilir.

metin, yazı tipi, ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Alt düğüm sayısı hesaplanır(size).

Mevcut düğümün tüm alt düğümleri üstünde gezinilir. Her bir alt düğüm child değişkeniyle işlenir.

Görsellik oluşturmak için uygun bir girinti hazırlanır.

Prefix: önceki düğümlerden miras alınan girinti.

İndeks her iterasyonda artırılır. İndeks size’a eşit olduğunda bu düğüm son çocuk demektir.

Eğer düğüm, kendi ebeveyninin son çouğuysa kuyrukmu true döner girinti olarak da boşluk bırakır ama eğer düğüm son çocuk değilse “|” kullanılır.

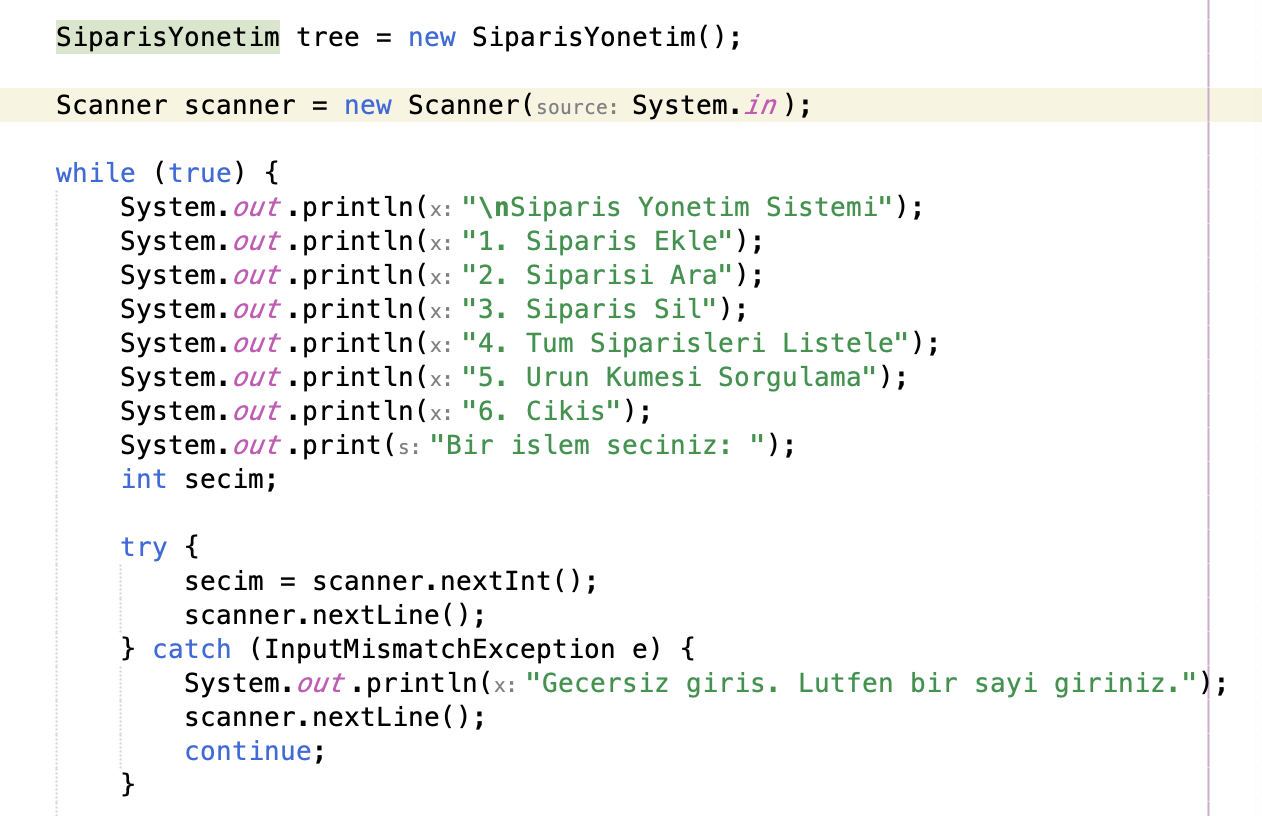
Alt düğümleri yazdırmak amacıyla aynı methodu tekrar çağırırız.

Kökten başlayarak tüm dallları yazdırmış oluruz.

metin, yazı tipi, çizgi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. Son classımız olan main class’da ise ilk başta ‘sipariş yönetim’ classımızın nesnesini oluşturduk.

Ardından seçim ekranı yazdık, uyuşmayan girişler exception fırlattık.

Seçime uygun olan methodları çağırıp, yazdırdık. metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu