**PROGRAMLAMA LABARATUVARI 4.PROJE**

**170201043-İLKER TINKIR, 170201075-ALİ EKEN**

**ÖZET:**

Bu projede bizimle paylaşılan isterler doğrultusunda kök düğümünün istenilen sayıda çocuğa sahip olduğu bir ağaç, o ağacın her bir çocuk düğümüne bağlı ikili yapıda ve çocuk düğümleri “Max-Heap” algoritmasına göre sıralanmış bir ikili ağaç ve bu ağacın da her bir çocuk düğümüne bağlı linked-list yapılarının beraberce ifade edildiği ve tüm yapıdaki her bir parçanın mantıklı bilgiler tuttuğu bir yazılım geliştirdik.

**GİRİŞ:**

Geliştirdiğimiz projede işlem ve okuma kolaylığı olması bakımından önemli işlemleri farklı sınıflarda düzenli bir biçimde yapmaya özen gösterdik. Main methodunda temel olarak sadece ana işlemleri ve kullanıcıya sunulacak arayüz işlemlerini gerçekleştirdik. Bunun dışında bu arayüz içerisinde pek çok kez kullanılması gereken bazı methodları “Static” bir biçimde main class içerisinde tanımlayarak kod uzunluğu bakımından tasarruf sağladık.

Projenin ilk aşaması olan “Kategori Ağacı” yapısını geliştirmek için “KategoriAgaci” ve bu ağacın düğümlerini tutmak amacıyla da “dugum” adlarında iki sınıf oluşturduk. “dugum” sınıfı içerisinde her bir düğümün dinamik biçimde çocuk düğümlerinin tutulacağı dugum tipinde bir ArrayList kullandık. Bu şekilde her bir düğüm bir ArrayList yardımıyla alt kategorilerine ulaşabilecek ve alt kategoriler de aynı şekilde varsa daha alt kategorileriyle bağlantı kurabililecek şekilde düzenlendi.

Bunun dışında bu “dugum” sınıfı, içerisine alt kategori sayisi, rezervasyon sayısı, kategori yolu, derinlik gibi temel özellikler ile kendisine yeni bir düğüm gönderildiğinde bu düğümü kendi içerisindeki ArrayList’e ekleyebilecek ve alt kategorilerinin sayısını bir arttıracak bir “ekle()” ve daha sonradan kullanılmak üzere bir Max-Heap ağacını oluşturacak biçimde de bir “AgacOlustur()” methoduna sahiptir. KategoriAgaci sınıfı ise içerisinde tüm kategorilerin en üst düğümlerinin sonunda bağlanacağı, “dugum” tipindeki bir kök düğümünü içerir.

Projenin ikinci aşaması olan her kategori altındaki müşterilerin rezervasyon sayılarına göre bir “Max-Heap” ağacına yerleştirilmesi kısmını gerçekleştirmek için “MaxHeap” ve “dugum2” adlı 2 adet sınıf kullandık. MaxHeap sınıfında MaxHeap algoritmasını temel olarak, rekürsif bir yapıyla oluşturan ekle() ve elemanları ekli olan tamamlanmamış ağaç yapısının düğümlerini rezervasyon sayılarına göre yeniden düzenleyen bir degistir() methodu kullandık. “dugum2” sınıfı ise kendi içerisinde yine dugum2 tipinde right, left, previous ve String ile integer tiplerinde ad, kategori adı, rezervasyon sayısı gibi temel bilgielri tutan değişkenler ile daha sonradan projenin üçüncü aşamasında kullanılacak olan “LinkedList()” adlı bir method içerecek biçimde tasarlandı. Dugum2 tipindeki “previous” düğümleri Max-Heap ağacını düzenlerken çocuk düğümün ata düğümle gerektiği taktirde değiş edilebilmesi için tutulmuştur.

Projenin üçüncü aşamasında her bir kategori dügümü altında bulunan Max-Heap ağacı elemanlarının içermesi gereken linked list yapıları tanımlanmıştır. Linked-List yapıları oluşturulurken dosyadan çekilen veriler ile Max-Heap ağacındaki düğümlerin kendi içerisinde tuttuğu veriler birbiriyle karşılaştırırlmış ve dugumler art arda birbirine eklenmiştir.

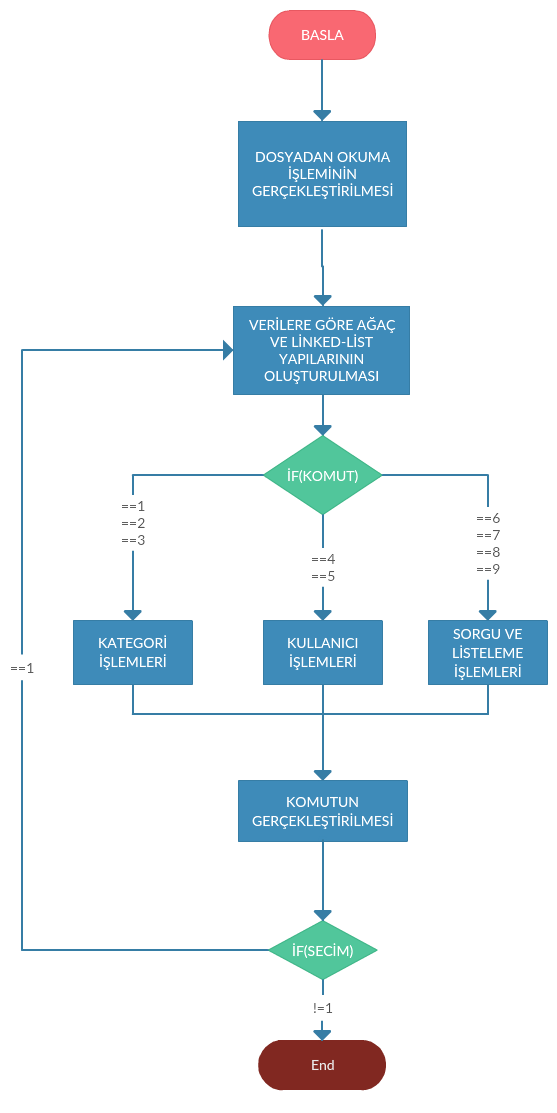
**YÖNTEM:**

Oluşturduğumuz sınıflarda özellikle düğüm sınıflarının içerisinde diğer düğümlerinin nesnelerini oluşturarak Ağaçların yapraklarının taşıdıkları diğer ağaç ve linked listlere doğrudan bir ulaşma yöntemi seçtik. Kategori ağacını oluştururken dinamik bir yapı oluşturmak amacıyla ArrayList yapısını kullandık. Bu sayede her bir kategori düğümüne istenildiği kadar alt kategori eklenebilir hale geldi. Max-Heap ağacını oluşururken ekleme işleminde Rekürsif bir yapı kullanarak işlemlerimizi tamamladık. Main methoduna arayüz oluştururken birden fazla defa kullanmamız gereken ve uzun kodları main class’ta Static methodlar olarak tanımladık. Bu static methodları oluşturduğumuz kullanıcı arayüzünde kullanarak daha az kod yazmış olduk.

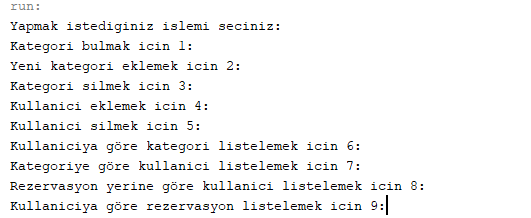
**YAPILAN ARAŞTIRMALAR:**

Max-Heap algoritmasını kurabilmek için nasıl çalıştığına dair araştırmalar yaptık.Bu araştırma sonucunda dizi şeklinde ve düğüm şeklinde yapılabileceğini öğrendik. Ancak dizi şeklindeki yöntemin kabul edilmemesi sebebiyle düğüm şeklinde yazdık. Linked-list yapısını daha önceden C dilinde biliyorduk ancak Java’da bu yapının farklı olması sebebiyle bu yapı üzerine araştırmalar yaptık ve projemizi buna göre düzenledik.

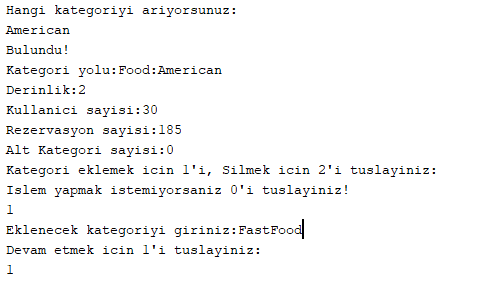
**AKIŞ ŞEMASI:**



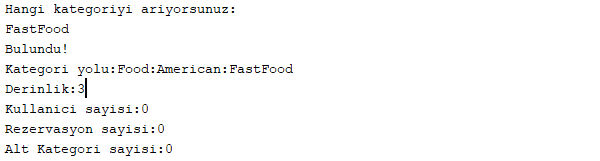
**ÖRNEK SONUÇLAR:**

****

1-) Kullanıcıya yapmak istediği işlemin sorulduğu arayüz.

****

2-)Kullanıcının seçtiği duruma göre “American” kategorisinin bulunması ve bu kategorinin altına “FastFood” kategorisinin eklenmesi.

****

3-) Eklenen kategorinin tekrar aranması ve bilgilerinin ekrana bastırılması.

**KAYNAKÇA:**

1) <https://www.sanfoundry.com/java-program-implement-max-heap/>

2)<https://codereview.stackexchange.com/questions/33365/heap-implementation-using-pointer>

3)<https://www.youtube.com/watch?v=ixdWTKWSz7s>

4) <https://www.geeksforgeeks.org/max-heap-in-java/>

5) <https://www.geeksforgeeks.org/linked-list-in-java/>

6) <http://blog.emrahkahraman.com.tr/java-dosya-okuma/>