

**EGE UNIVERSITY**

**FACULTY OF ENGINEERING**

**COMPUTER ENGINEERING DEPARTMENT**

**204 DATA STRUCTURES (3+1)**

**2021–2022 FALL SEMESTER**

**PROJECT-3 REPORT**

**(Search Tree, Heap, Hash Table, Food Delivery Company)**

**DELIVERY DATE**

15/01/2022

**PREPARED BY**

05190000114 , Mahmut Çelik

İçindekiler

[1.a YemekSınıfı nesnesinin ve Yemek Sipariş Ağacının oluşturulması 3](#_Toc92533878)

[1.b Ağaç derinliği ve ağaçtaki bilgilerin ekrana listelenmesi 13](#_Toc92533879)

[1.b.1 Kaynak Kod 13](#_Toc92533880)

[1.b.2 Ekran görüntüleri 17](#_Toc92533881)

[1.c 150 TL üstündeki siparişlerin bilgilerini listeleme 38](#_Toc92533882)

[1.c.1 Kaynak Kod 38](#_Toc92533883)

[1.c.2 Ekran görüntüleri 41](#_Toc92533884)

[1.d Bir yiyecek/içeceğin tüm ağaçta kaç adet sipariş verildiğini döndürme ve fiyat güncelleme 45](#_Toc92533885)

[1.d.1 Kaynak Kod 45](#_Toc92533886)

[1.d.2 Ekran görüntüleri 47](#_Toc92533887)

[2.a Hash Tablosuna Ekleme 47](#_Toc92533888)

[2.b Hash Tablosu Güncelleme 47](#_Toc92533889)

[3.a Heap Veri Yapısı Tasarlama 49](#_Toc92533890)

[3.b Max Heap düğüm yerleştirme 59](#_Toc92533891)

[3.c Heap bilgi çekme 60](#_Toc92533892)

[3.c.1 Kaynak Kod 60](#_Toc92533893)

[3.c.2 Ekran görüntüleri 64](#_Toc92533894)

[4.a Simple sorting veya Advanced sorting algoritması 65](#_Toc92533895)

[4.b Sıralama algoritması ile karşılaştırılması 67](#_Toc92533896)

[4.c Görselleştirme araçları 68](#_Toc92533897)

[Özdeğerlendirme Tablosu 68](#_Toc92533898)

ARAMA AĞAÇLARI, YIĞINLAR VE HASH TABLOSU: YEMEK DAĞITIM ŞİRKETİ

NetBeans , 12.6, and Java

## 1.a YemekSınıfı nesnesinin ve Yemek Sipariş Ağacının oluşturulması

YemekSinifi CLASSI

package tree;

import java.util.Random;

public class YemekSinifi { //YEMEKSINIFI CLASSI AGACTAKI EN KUCUK BIRIMIN OLUSTURUŞACAGI CLASS

private String yemekAdi; //YEMEKADI SIPARIS ADEDI VE BIRIM FIYATI OZELLIKLERI BULUNUYOR

private int yemekSiparisAdedi;

private int yemekBirimFiyati;

public YemekSinifi(String yemekAdi , int yemekBirimFiyati){ //CONSTRUCTORDA YEMEKADI VE BIRIM FIYATI GELIYOR AYNI YEMEKTEN 1 SIPARISTE 2 DEFA OLMAMASI ICIN BOYLE YAPTIM YEMEKSIPARISADEDI DE 1LE 8 ARASI RANDOM SAYI SECILIYOR

Random r = new Random();

this.yemekAdi = yemekAdi;

this.yemekSiparisAdedi = r.nextInt(8)+1;

this.yemekBirimFiyati = yemekBirimFiyati;

}

public String getYemekAdi() { //GETTER VE SETTERLAR

return yemekAdi;

}

public void setYemekAdi(String yemekAdi) {

this.yemekAdi = yemekAdi;

}

public int getYemekSiparisAdedi() {

return yemekSiparisAdedi;

}

public void setYemekSiparisAdedi(int yemekSiparisAdedi) {

this.yemekSiparisAdedi = yemekSiparisAdedi;

}

public int getYemekBirimFiyati() {

return yemekBirimFiyati;

}

public void setYemekBirimFiyati(int yemekBirimFiyati) {

this.yemekBirimFiyati = yemekBirimFiyati;

}

public int yemekToplamFiyat(){ //1 YEMEK NESNESININ TOPLAM FIYATINI HESAPLAYAN METOD ORNEK VERECEK OLURSAK SALATA ADET 5TL VE SIPARIS ADEDI 4 TANE 20TL YI HESAPLAR

return yemekSiparisAdedi\*yemekBirimFiyati;

}

@Override

public String toString(){ //YEMEKSINIFININ TOSTRING METODU

return "Yemek ADI: "+ yemekAdi +"\n"+

"Sipariş Adedi: " + yemekSiparisAdedi + "\n"+

"Birim Fiyatı: " + yemekBirimFiyati + "TL";

}

}

SiparisBilgileri CLASSı

package tree;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Random;

public class SiparisBilgileri { //SIPARIS BILGILERI CLASSI

//AYNI YEMEKTEN 2 TANE OLMAMASI ICIN BU CLASSTAN YEMEKLER RASTGELE OLARAK SECILIR VE YEMEKSINIFINDAN NESNE OLUSTURULUR

//VERILERIN SAKLANMASI ICIN DIZILERE KENDI BELIRLEDIGIM FIYAT VE SAYIDA YEMEKLER VE FIYATLARINI EKLEDIM

//AYNI SIPARISLISTESINDE 2 DEFA OLMASINI ONLEMEK ICIN GECICI 2 TANE ARRAYLIST OLUSTURDUM.

//BU ARRAYLISTLERE DIZILER COLLECTIONS UN ADDALL METODU ILE EKLENIYOR VE 1 YEMEK SECILIP YOLLANDIGINDA ARRAYLISTTEN SILINIYOR

private final String[] yemekAdlari ={"Pizza","Burger","Döner","Pide","Waffle","Çiğ Köfte","Lahmacun","Sütlaç","Kazandibi","Kabak Tatlısı","Börek","Cola","Sprite","Espresso","Ayran","Fanta","Yoğurt Çorbası"}; //KENDI BELIRLEDIGIM YEMEKLER VE FIYATLARI

private final Integer[] yemekFiyatlari ={50,40,40,35,35,10,10,5,5,5,20,15,10,20,10,8,6};

private ArrayList<YemekSinifi> siparisBilgileriListesi = new ArrayList<>(); //YEMEKLERIN TUTULACAGI ARRAYLIST

//NOT: BURADA YANLIS ADLANDIRMA YAPMISIM KARISIKLIK OLMAMASI ICIN ACIKLAMA GEREKSINIMI DUYDUM. SIPARIS BILGILERI LISTESI PROJEDE MAHALLE NESNESININ OZELLIGI OLARAK GOZUKMEKTEDIR BURADA YAPMAM GEREKEN ADLANDIRMA siparisBilgileri olması gerekiyordu kod dogru calısıyor fakat siz incelerken yanlış anlaşılma olmasın diye bunu yazdım.

public SiparisBilgileri(){ //CONSTRUCTOR

Random r = new Random();

int sayi = r.nextInt(3)+3; //SIPARISBILGILERINDE KAC TANE YEMEK SINIFINDAN NESNE OLACAKSA RANDOM SEKILDE 3ILE 6 ARASINDA BELIRLENIR

ArrayList<String> geciciYemekAdlari = new ArrayList<>(); //GECICI YEMEK ARRAYLISTLERI BURADAN SECILEN ELEMAN SONRASINDA SILINECEK

ArrayList<Integer> geciciYemekFiyatlari = new ArrayList<>();

Collections.addAll(geciciYemekAdlari, yemekAdlari); //ARRAYLISTE DIZILERIN EKLENMESI

Collections.addAll(geciciYemekFiyatlari, yemekFiyatlari);

for(int a=0 ; a<sayi ; a++){ //FOR DONGUSUYLE BELIRLENEN SAYI KADAR YEMEKSINIFI NESNESI EKLENIR

int sayi2 = r.nextInt(geciciYemekAdlari.size());

siparisBilgileriListesi.add(new YemekSinifi(geciciYemekAdlari.get(sayi2), geciciYemekFiyatlari.get(sayi2))); //ARRAYLISTTEN ELEMAN SECILIR VE YENI NESNE OLUSTURULUP ARRAYLISTE EKLENIR

geciciYemekAdlari.remove(sayi2); //SECILEN YEMEK ARRAYLISTTEN SILINIR

geciciYemekFiyatlari.remove(sayi2);

}

}

public ArrayList<YemekSinifi> getSiparisBilgileriListesi() { //GETTER SETTER

return siparisBilgileriListesi;

}

public void setSiparisBilgileriListesi(ArrayList<YemekSinifi> siparisBilgileriListesi) {

this.siparisBilgileriListesi = siparisBilgileriListesi;

}

public void siparisBilgileriYazdir(){ //SIPARIS BILGILERININ YAZDIRILMASI

for(YemekSinifi yemekler : siparisBilgileriListesi){

System.out.println(yemekler.toString() + "\n");

}

}

public int siparisBilgileriToplamFiyat(){ //SIPARISBILGILERININ TOPLAM FIYATININ FOR EACH DONGUSUYLE ILK BASTA OLUSTURDUGUMUZ ARRAYLISTIN DOLASILIP YEMEKSINIFINDAKI TOPLAMA METODUNU KULLANARAK SIPARISBILGILERINDEKI TOPLAM FIYAT HESAPLANIR

int toplamFiyat = 0;

for(YemekSinifi yemekler : siparisBilgileriListesi){

toplamFiyat += yemekler.yemekToplamFiyat();

}

return toplamFiyat;

}

}

Mahalle CLASSı

package tree;

import java.util.ArrayList;

import java.util.Random;

public class Mahalle { //MAHALLE NESNELERININ OLUSTURULACAGI CLASS

private String mahalleAdi; //MAHALLE CLASSININ 2 OZELLIGI VAR 1 MAHALLE ADI IKINCI OLARAK SIPARIS BILGILERININ TUTULACAGI SIPARISLER LISTESI

private ArrayList<SiparisBilgileri> siparislerListesi = new ArrayList<>(); //SIPARISBILGILERINI TUTMAK ICIN ARRAYLIST KULLANDIM

public Mahalle(String mahalleAdi){ //CONSTRUCTOR

this.mahalleAdi = mahalleAdi;

Random r = new Random(); //5 ILE 11 ARASI RANDOM SAYI OLUSTURMAK ICIN RANDOM KULLANILACAK

int sayi = r.nextInt(6)+5;

for(int a=0 ; a<sayi ; a++){ //RANDOM OLUSAN SAYI KADAR DONECEK FOR DONGUSUYLE SIPARISLISTESI ARRAYLISTINE SIPARISBILGILERINDEN NESNE OLUSTURULUP ADD METODU ILE EKLENIR

siparislerListesi.add(new SiparisBilgileri());

}

}

public String getMahalleAdi() { //GETTER SETTER

return mahalleAdi;

}

public void setMahalleAdi(String mahalleAdi) {

this.mahalleAdi = mahalleAdi;

}

public ArrayList<SiparisBilgileri> getSiparislerListesi() {

return siparislerListesi;

}

public void setSiparislerListesi(ArrayList<SiparisBilgileri> siparislerListesi) {

this.siparislerListesi = siparislerListesi;

}

public void mahalleBilgiYazdir(){ //MAHALLE BILGILERININ YAZDIRILMASI MAHALLE ADI VE FOR EACH DONGUSU ILE SIPARISBILGILERILISTERI ARRAYLISTI DOLASILARAK YAZDIRMA GERCEKLESIR

System.out.println("Mahalle Adi: "+ mahalleAdi + "\n");

int siparisIndex =1;

for(SiparisBilgileri siparis : siparislerListesi){

System.out.println(siparisIndex+". sipariş: ");

siparis.siparisBilgileriYazdir();

siparisIndex++;

}

}

}

TREE CLASSINDA BU KISIM ICIN GEREKLI METODLAR

package tree;

public class TREE {

private TreeNode root;

private int derinlik = -1; //AGACIN ILK DERINLIGININ 0 OLMASI ICIN -1 ATADIM. AGAC DOLASILIRKEN HER YENI RECURSION FONK GIRILDIGINDE 1 ARTACAK VE BU DEGERDE TUTULACAK

private int maxDerinlik =0; //MAXDERINLIGI TUTMAK ICIN GEREKLI DEGISKEN

private int toplamYemekSayisi =0; //VERILEN YEMEGIN AGACTAKI TOPLAM SIPARIS SAYISINI BULMAK ICIN GEREKLI DEGISKEN

private int yemeginEskiFiyati =0; //BUNLAR GEREKLI DEGIL FAKAT FIYAT DEGISIMINI GOREBİLMEK ICIN YEMEGIN ESKI VE YENI FIYATINI TUTUYORLAR

private int yemeginYeniFiyati =0;

public TREE(){ //CONSTRUCTORDA ROOT A NULL ATAMASI YAPTIK AGACI KAYBETMEMEK ICIN ONEMLI ROOT KODUN YUKARISINDA KALMIS

this.root =null;

}

public TreeNode getRoot() { //ROOT ICIN GETTER VE SETTER

return root;

}

public void setRoot(TreeNode root) {

this.root = root;

}

public class TreeNode{ //AGAC DUGUMLERININ OLUSTURULDUGU CLASS ICINDE DATA SOL VE SAG COCUK OZELLIKLERI BULUNMAKTA

private Mahalle mahalleNesnesi; //DATA OLARAK MAHALLE CLASSINDAN NESNELER TUTAR

private TreeNode leftChild;

private TreeNode rightChild;

public TreeNode(Mahalle mahalleNesnesi){ //CONSTRUCTOR

this.mahalleNesnesi = mahalleNesnesi;

this.leftChild = null;

this.rightChild = null;

}

public void displayNode(){ //OLUSTURULAN TREENODUN GORUNTULENMESI ICIN METOD MAHALLE CLASSINDAKI BILGI YAZDIRMA METODU KULLANARAK YAZDIM

mahalleNesnesi.mahalleBilgiYazdir();

}

}

}

MAIN CLASSı

package tree;

public class MAINCLASS {

static final int istenilenFiyat = 150; //C SIKKI ICIN GEREKLI OLDUGUNDAN BURADA TANIMLADIM

static final String[] mahalleAdlari = {"Evka 3","Özkanlar","Atatürk","Erzene","Kazımdirik"}; //DOKUMANDA VERILEN ORNEK MAHALLELERI DIZIYE EKLEDIM

public static void main(String[] args) {

TREE myTree = new TREE(); //TREE CLASSINDAN NESNE OLUSTURDUM

for(String eklenecekMahalle : mahalleAdlari){ //OLUSTURULAN TREE YE ADD METODU KULLANILARAK MAHALLELER EKLENIYOR

myTree.add(eklenecekMahalle);

}

}

}

## 1.b Ağaç derinliği ve ağaçtaki bilgilerin ekrana listelenmesi

### 1.b.1 Kaynak Kod

public void agactakiTumBilgilerinYazdirilmasi(){ //1.B MADDESI ICIN CALISACAK METOD

//AGACIN DERINLIGI VE SONRASINDA TUM AGACTAKI BILGILERIN YAZDIRILMASINI YAPAR

derinligiBulma(root, derinlik); //DERINLIGI DEGISKEN UZERINE ATAR

System.out.println("AGACIN DERİNLİGİ: "+ maxDerinlik + "\n");

preOrder(root); //AGACIN DOLASILMASI HERHANGI BIR DOLASMA YONTEMI DE OLABILIRDI BUNU TERCIH ETTIM

}

private void derinligiBulma(TreeNode node , int derinlik){ //AGACIN DERINLIGINI TUM AGACI DOLASARAK VE MEVCUT DERINLIGI MAXDERINLIKLE KARSILASTIRARAK MEVCUT DAHA DERINSE DEGERLERI GUNCELLEYEN RECURSION METOD

if(node !=null){ //NODE NULL OLMADIGI SURECE CALISACAK RECURSION

derinlik ++; //HER 1 KADEME ASAGIYA INILDIGINDA DERINLIK 1 ARTTIRILIR

if(derinlik > maxDerinlik){ //DERINLIGIN KARSILASTIRILIP BLOK CALISIRSA MAXDERINLIGIN GUNCELLENMESI

maxDerinlik =derinlik;

}

derinligiBulma(node.leftChild, derinlik); //TUM AGACIN DOLASILMASINI SAGLAYAN KISIM

derinligiBulma(node.rightChild,derinlik);

}

}

public void add(String stringToAddTree){ //AGACA ELEMAN EKLEMEYE SAGLAYAN METOD ICINDE RECURSION FONKSIYONU CAGIRIYOR

root = addWithRecursion(root, stringToAddTree);

}

private TreeNode addWithRecursion(TreeNode node , String stringToAddTree){ //RECURSION BIR METOD NODE NULLSA ELEMAN ELEMAN EKLENIR KARSILASTIRMA DURUMUNA GORE AGACTA ILERLENIR

if(node ==null){ //NODE NULL A ESIT OLDUGUNDA CALISACAK BLOK EKLENECEK MAHALLENIN ADINDA MAHALLE NESNESI OLUSTURULUR VE AGACA EKLENIR

node = new TreeNode(new Mahalle(stringToAddTree));

return node;

}

// IKI STRINGIN COMPARETO METOD SONUCU

// SAYI 0 CIKIYORSA AYNI STRINGLERDIR

// SAYI 1 CIKIYORSA SOLDAKI SAGDAKINDEN ALFABETIK SIRADA DAHA BUYUKTUR F > A MESELA

// SAYI -1 CIKIYORSA SOLDAKI SAGDAKINDEN ALFABETIK SIRADA DAHA KUCUKTUR N < S MESELA

if(stringToAddTree.compareTo(node.mahalleNesnesi.getMahalleAdi()) <0){ //SAYI -1 SE SOLA GİDİLİR

node.leftChild = addWithRecursion(node.leftChild, stringToAddTree);

return node;

}

else{ //SAYI 1SE SAGA GIDILIR

node.rightChild = addWithRecursion(node.rightChild, stringToAddTree);

return node;

}

}

public void preOrder(TreeNode node){ //AGACIN ILK OLARAK SOL ELEMANI SONRA PARENT EN SON SAG ELEMANIN YAZDIRILDIGI RECURSION DOLASMA YONTEMI

if(node !=null){ //NODE NULL OLMADIGI SURECE ILERLEYECEK

node.displayNode();

preOrder(node.leftChild);

preOrder(node.rightChild);

}

}

public void inOrder (TreeNode node){ //AGACIN ILK OLARAK PARENT SONRA SOL ELEMANI EN SON SAG ELEMANIN YAZDIRILDIGI RECURSION DOLASMA YONTEMI

if(node!=null){ //NODE NULL OLMADIGI SURECE ILERLEYECEK

inOrder(node.leftChild);

node.displayNode();

inOrder(node.rightChild);

}

}

public void postOrder (TreeNode node){ //AGACIN ILK OLARAK SOL ELEMANI SONRA SAG ELEMANI EN SON PARENT ELEMANIN YAZDIRILDIGI RECURSION DOLASMA YONTEMI

if(node!=null){ //NODE NULL OLMADIGI SURECE ILERLEYECEK

postOrder(node.leftChild);

postOrder(node.rightChild);

node.displayNode();

}

}

### 1.b.2 Ekran görüntüleri

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

## 1.c 150 TL üstündeki siparişlerin bilgilerini listeleme

### 1.c.1 Kaynak Kod

MAIN CLASS I ICINDEKI KODLAR

myTree.dolasmaVeFiyatUstuYazdirma("Evka 3", istenilenFiyat); //C maddesi mahalle adı girilebilir

TREE CLASSI ICINDEKI KODLAR

public void dolasmaVeFiyatUstuYazdirma(String arananMahalleAdi , int fiyat){ //1.C KISMI ICIN CALISACAK METOD

//ADI GELEN MAHALLE BASKA BIR METODA GIDILEREKK ORADA KULLANILIR

dolasmaVeYazdirma(getRoot(), arananMahalleAdi, fiyat);

}

private void dolasmaVeYazdirma(TreeNode node , String arananMahalleAdi ,int fiyat){ //TUM AGAC DOLASILIR VE ISTENEN MAHALLEDEKI SIPARISBILGILERI GELEN FIYAT UZERIYSE YAZDIRILIR

// BURADA MAHALLENIN TREE DE BULUNMAMA DURUMUNU KONTROL ETMEDIM DOKUMANDA DA OLMAYINCA KODU YAZARKEN AKLIMA GELMEDI

// SONDAKI ELSE BLOGUNU ELSE IF YAPIP COMPARETO = 0 YAPILIP EN SONA ELSE EKLENIRSE MAHALLENIN AGACTA OLMAMA DURUMUNDA HATALI CALISMA DURUMUNUN ONUNE GECILMIS OLUNUR

if(node!= null){ //NODE NULL OLMADIGI SURECE CALISACAK RECURSION

// IKI STRINGIN COMPARETO METOD SONUCU

// SAYI 0 CIKIYORSA AYNI STRINGLERDIR

// SAYI 1 CIKIYORSA SOLDAKI SAGDAKINDEN ALFABETIK SIRADA DAHA BUYUKTUR F > A MESELA

// SAYI -1 CIKIYORSA SOLDAKI SAGDAKINDEN ALFABETIK SIRADA DAHA KUCUKTUR N < S MESELA

if(arananMahalleAdi.compareTo(node.mahalleNesnesi.getMahalleAdi())<0){ //ARANAN MAHALLE VE AGAC DOLASILIRKEN MEVCUT NODE KARSILASTIRMA SONUCU SOLA VEYA SAGA GIDILIR

dolasmaVeYazdirma(node.leftChild, arananMahalleAdi , fiyat);

}

else if(arananMahalleAdi.compareTo(node.mahalleNesnesi.getMahalleAdi())>0){

dolasmaVeYazdirma(node.rightChild, arananMahalleAdi , fiyat);

}

else{ //MAHALLEYE GELINDIGINDE MAHALLE ADI YAZDIRILIR

//SONRASINDA SIPARISLERLISTESI DOLASILIR ICINDE SIPARISBILGILERI SAKLIYORDU O CLASSIN TOPLAM FIYAT METODU KULLANILARAK KARSILASTIRMA FIYATIMIZDAN BUYUKSE BILGILERI YAZDIRILIR

System.out.println("MAHALLE ADI: "+node.mahalleNesnesi.getMahalleAdi());

for(SiparisBilgileri siparisBilgileri : node.mahalleNesnesi.getSiparislerListesi()){ // SIPARISLERLISTESININ FOREACH ILE DOLASILMASI

if(siparisBilgileri.siparisBilgileriToplamFiyat()>fiyat){ //SIPARISBILGILERI TOPLAM FIYATI BUYUKSE CALISACAK BLOK

System.out.println("SİPARİŞ TOPLAM FİYAT: "+siparisBilgileri.siparisBilgileriToplamFiyat() + "TL ");

siparisBilgileri.siparisBilgileriYazdir();

}

}

}

}

}

### 1.c.2 Ekran görüntüleri

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

## 1.d Bir yiyecek/içeceğin tüm ağaçta kaç adet sipariş verildiğini döndürme ve fiyat güncelleme

### 1.d.1 Kaynak Kod

MAIN CLASSI ICINDE CALISACAK KISIM

myTree.arananYemeginToplamSayisiveFiyatinGuncellenmesi("Burger");

TREE CLASSINDAKI METODLAR

public void arananYemeginToplamSayisiveFiyatinGuncellenmesi(String arananYemeginAdi){ //1.D ICIN CALISACK METOD BASKA BIR METODA GIDER YAZDIRMA KISIMLARINI VE MAINDE YAPILACAK GETROOT YERINE O ISLEMI BURADA YAPAR

yemekAramaVeFiyatGuncelleme(arananYemeginAdi, root); //GIDILECEK RECURSION METOD

System.out.println("AĞACTAKİ TOPLAM "+arananYemeginAdi + " SAYISI: "+ toplamYemekSayisi); //DEGISKENDE TUTULAN SAYI GIDILEN METODDA DEGISIR VE BURADA O DEGISKEN KULLANILARAK SAYI YAZDIRILIR

System.out.println("İSTENEN YEMEGİN ESKİ FİYATI: "+ yemeginEskiFiyati+"\nİSTENEN YEMEĞİN YENİ FİYATI: "+ yemeginYeniFiyati); //BURASI GEREKLI DEGIL FAKAT GORME ACISINDAN DAHA IYI OLACAGINI DUSUNDUGUM ICIN YAZDIM

}

private void yemekAramaVeFiyatGuncelleme(String arananYemeginAdi , TreeNode node ){ //AGACTAKI TUM NESNELER DOLASILIR HER NESNENIN SIPARISLISTERI SIPARISLISTELERININ SIPARISBILGILERI SIPARISBILGILERINDEKI YEMEKSINIFI NESNELERINDEKI METODLAR KULLANILARAK KARSILASTIRMA YAPILIR

if(node !=null){ //NODE NULL OLMADIGI SURECE CALISACAK RECURSION

for(SiparisBilgileri siparisBilgileri : node.mahalleNesnesi.getSiparislerListesi()){ //NODE DAKI MAHALLE NESNESININ SIPARISLISTESININ DOLASILMASI

for(YemekSinifi yemekler : siparisBilgileri.getSiparisBilgileriListesi()){ //SIPARISBILGILERININ DOLASILIP YEMEKSINIFI NESNELERININ INCELENMESI

if (yemekler.getYemekAdi().equals(arananYemeginAdi)){ //GELEN YEMEKSINIFI NESNESI ARANANYEMEKLE ESITSE CALISACAK IF BLOGU

yemeginEskiFiyati = yemekler.getYemekBirimFiyati();

yemekler.setYemekBirimFiyati(yemekler.getYemekBirimFiyati() - (yemekler.getYemekBirimFiyati()\*10/100)); //FIYATTA INDIRIM YAPILIR VE DEGISTIRILIR

yemeginYeniFiyati = yemekler.getYemekBirimFiyati();

int yemekSiparisAdedi = yemekler.getYemekSiparisAdedi();

toplamYemekSayisi += yemekSiparisAdedi; //ARANAN YEMEGIN SIPARIS ADEDI ILK BASTA OLUSTURULAN DEGISKENE EKLENIR

}

}

}

yemekAramaVeFiyatGuncelleme(arananYemeginAdi, node.leftChild); //DOLASMAYI SAGLAYAN KISIM

yemekAramaVeFiyatGuncelleme(arananYemeginAdi, node.rightChild);

}

}

### 1.d.2 Ekran görüntüleri

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

## 2.a Hash Tablosuna Ekleme

// FOR DONGUSUYLE ILK BASTA OLUSTURULAN MAHALLELERIN SAKLI OLDUGU DIZIDEKI ELEMANLARI HASHMAP E PUT METODU ILE EKLEYEN METOD

public static void hashmapElemanlariEkleme(HashMap<String,Integer> myHashMap){

for(int a = 0; a<mahalleAdlari.length ;a++){ //FOR DONGUSU

myHashMap.put(mahalleAdlari[a],mahalleNufuslari[a]);

}

}

## 2.b Hash Tablosu Güncelleme

//KONSOLDAN SCANNER YARDIMIYLA HARF ALIP BU HARFI BUYULTUP SONRASINDA FOREACH DONGUSUYLE HASHMAPTEKI KEY DEGERLERINI DOLASARAK UYGUN MAHALLELERIN NUFUSLARINI 1 ARTTIRAN METOD

public static void basHarfeGoreNufusArttirma(HashMap<String,Integer> myHashMap){

//SCANNER

Scanner scanner = new Scanner(System.in);

//HARF ALIMI

System.out.print("UYGUN MAHALLELERIN NUFUSLARINI ARTTIRMAK ICIN HARF GIRINIZ: ");

char basHarf = scanner.next().toUpperCase().charAt(0); //BAS HARFIN OLUSTURULMASI

int arttirilanSehirSayisi =0;

for(String keys : myHashMap.keySet()){ //FOR EACH DONGUSUYLE HASHMAPTEKI KEYSET DOLASILIR VE OLUSTURULAN BAS HARF KEYSETTEKI MAHALLELERIN ILK HARFLERINE ESITSE O MAHALLENIN NUFUSU 1 ARTTIRILIR

if(keys.charAt(0) ==basHarf){

myHashMap.replace(keys, myHashMap.get(keys)+nufusArttirmaMiktari);

arttirilanSehirSayisi++;

}

}

if(arttirilanSehirSayisi ==0){ //SEHIR NUFUSU ARTTIRILMAMISSA

System.out.println("HashMap'te GİRDİĞİNİZ BAS HARFTE MAHALLE BULUNMAMAKTADIR.");

}

else{ //SEHIR NUFUSU ARTTIRILMISSA SEHIRLER YAZDIRILIR

System.out.println("VERILEN BAS HARFE GORE ILGILI MAHALLELERIN NUFUSLARININ 1'ER ARTTIRILDIKTAN SONRAKI NUFUSLARI");

hashMapElemanlariYazdirma(myHashMap);

}

}

public static void hashMapElemanlariYazdirma(HashMap<String,Integer> myHashMap){ //FOR EACH DONGUSUYLE HASH MAPTEKI KEY VE VALUE DEGERLERI YAZDIRILIR

for(String keys: myHashMap.keySet()){

System.out.print("Mahalle ADI: "+ keys+"\t");

System.out.println("NUFUSU: "+ myHashMap.get(keys));

}

}

## 3.a Heap Veri Yapısı Tasarlama

MAX HEAP CLASSI

package maxheap;

import java.util.ArrayList;

//MAXHEAP CLASSI

//PROJE ICIN GEREKLI OLAN KISIM BURASI

//MIN HEAP I KONUYU ANLAMAYA CALISIRKEN OLUSTURMUSTUM

public class HEAP { //HEAP CLASSI

private ArrayList<Node> heapList; //HEAPTEKI VERILERI ARRAYLISTTE SAKLAYARAK YAPTIM

public HEAP(){ //CONSTRUCTOR

heapList = new ArrayList<>();

}

public void add(Mahalle mahalle){ //HEAP E GELEN MAHALLE CLASSINDAN NESNENIN EKLENMESI

Node newNode = new Node(mahalle);

if(heapList.isEmpty()){ //LISTE BOSSA 0.INDEXE NULL 1.INDEXE GELEN MAHALLE NESNESI ARRAYLISTTE ADD METODU ILE EKLENIR

heapList.add(null);

heapList.add(1,newNode);

}

else{ //HEAPLIST BOS DEGILSE CALISACAK KISIM

heapList.add(newNode); //ARRAYLISTE ADD METODU ILE SONA DOGRU ELEMANLAR EKLENIR VE EKLENEN ELEMANLARIN HEAP YAPISINI BOZMAMASI ICIN YAPRAKTAN ROOT A DOGRU DUZENLENMESI GEREKIR

yukariyaDogruDuzenleme(); //HEAPIN YUKARIYA DOGRU DUZENLENMESI

}

}

public Mahalle delete(){ //HEAPTEN ELEMAN SILMEYI SAGLAYAN METOD

if(heapList.size() ==2){ //HEAPTE 1 TANE ELEMAN VARSA CUNKU 0.INDEXE NULL ATAMISTIK CALISACAK METOD

Node nodeToDelete = heapList.get(1); //SILINECEK ELEMAN DEGISKENE ATANIR VE 1.INDEXTE ELEMAN SILINIP VERISI GONDERILIR

heapList.remove(1);

return nodeToDelete.getMahalle();

}

else if(heapList.size()<2){ //SIZE 2DEN KUCUKSE ELEMAN YOKTUR

System.out.println("NO ELEMENT IN THE HEAP");

return null;

}

Node nodeToDelete = heapList.get(1); //YUKARIDAKI BLOKLARDAN RETURNLE CIKIS YAPMAZSA CALISACAK KISIMLAR

int lastElementIndex = heapList.size()-1; //SON EKLENEN ELEMANIN INDEXI ALINIR

heapList.set(1,heapList.get(lastElementIndex) ); //ILK YANI ROOTTAKI ELEMAN DEGISKENE ATANMISTI BURADA ONUN YERINE SON YAPRAKTAKI ELEMAN GELIR

heapList.remove(lastElementIndex); //SON YAPRAKTAKI ELEMAN SUAN ROOTTA OLDUGU ICIN O KISIM SILINIR

asagiyaDogruDuzenleme(); //HEAP YAPISI BOZULMAMASI ICIN ROOTTAKI ELEMAN COCUKLARIYLA KARSILASTIRILIP GEREKLI DEGISIKLIKLER YAPILARAK UYGUN DUZEN SAGLANIR

return nodeToDelete.getMahalle(); //SILINEN ELEMANIN GONDERILMESI

}

private void yukariyaDogruDuzenleme(){ //ELEMAN EKLENDIGINDE MAXHEAP YAPISININ KORUNMASI ICIN GEREKLI OLAN METOD

int lastElementIndex = heapList.size()-1; //SON ELEMANIN INDEXI BULUNUR

int parentIndex =lastElementIndex/2; //PARENTININ INDEXI BULUNUR

Node lastElement = heapList.get(lastElementIndex);

while(lastElementIndex>0 && parentIndex>0 && heapList.get(parentIndex).getMahalle().getMahalleNufusu() < lastElement.getMahalle().getMahalleNufusu()){ //INDEXLER ARRAYLISTTEN TASMIYORSA VE YAPRAKTAKI ELEMANIN NUFUSU PARENTINDAN BUYUK OLDUGU SURECE CALISACAK WHILE DONGUSU

//BUYUK OLDUKCA CALISACAK OLMASININ SEBEBI MAXHEAP YAPISINDA ROOTTA EN BUYUK ELEMAN BULUNUR

heapList.set(lastElementIndex , heapList.get(parentIndex)); //PARENT ASAGI INDIRILIR VE INDEX DEGERLERI GUNCELLENIR

lastElementIndex = parentIndex;

parentIndex = parentIndex/2;

}

heapList.set(lastElementIndex, lastElement); //WHILE DONGUSUNDEN CIKILNCA BULUNAN INDEX E YAPRAKTAKI ELEMAN EKLENIR

}

private void asagiyaDogruDuzenleme(){ //ROOTTAKI YANI MAXHEAPTEKI EN BUYUK ELEMAN SILININCE YERINE YAPRAKTAKI ELEMAN GETIRILIR VE YAPI TEKRARDAN DUZENLENIR

int firstElementIndex =1;

Node firstElement = heapList.get(firstElementIndex); //ILK ELEMAN SUAN YAPRAKTAKI ELEMAN

Node biggerChild = buyukCocukBulma(firstElementIndex); //MAX HEAP YAPISINDA DAHA BUYUK OLAN YUKARIDA OLACAGI ICIN BUYUK COCUK YUKARIYA GECER ONUN ICIN BUYUK COCUK BULUNUR

int biggerChildIndex = heapList.indexOf(biggerChild); //BUYUK COCUGUN INDEXI

while(biggerChild !=null && firstElement.getMahalle().getMahalleNufusu() < biggerChild.getMahalle().getMahalleNufusu()){ //YAPRAGA GIDILMEMISSE VE BUYUK COCUGUN NUFUSU PARENTTAN BUYUK OLDUGU SURECE YER DEGISTIRILIR

heapList.set(firstElementIndex, biggerChild); //BUYUK COCUK YUKARIYA CIKTI

heapList.set(biggerChildIndex, firstElement); //PARENTI COCUGUN ESKI YERINE GECTI

firstElementIndex = heapList.indexOf(firstElement); //YENI INDEX DEGERLERINE GORE ILK BASTAKI PARENT TEKRAR COCUKLARIYLA KARSILASTIRILACAK BUNUN ICIN TEKRARDAN COCUKLARIN BUYUK OLANI BULUNUR

firstElement = heapList.get(firstElementIndex);

biggerChild = buyukCocukBulma(firstElementIndex);

biggerChildIndex = heapList.indexOf(biggerChild);

}

}

private Node buyukCocukBulma(int parentIndex){ //BUYUK COCUGUN BULUNDUGU METOD

int leftChildIndex = 2\*parentIndex; //FORMULE UYGUN SEKILDE COCUKLARIN INDEXLERININ BULUNMASI

int rightChildIndex = 2\*parentIndex +1;

Node biggerChild;

if((leftChildIndex >=heapList.size() || rightChildIndex >=heapList.size())) //ARRAYLISTTEN TASIYORSA NULL DONDURUR

return null;

if(heapList.get(leftChildIndex).getMahalle().getMahalleNufusu() > heapList.get(rightChildIndex).getMahalle().getMahalleNufusu()){ //SOLDAKI COCUGUN NUFUSU DAHA BUYUKSE BUYUK COCUK SOLDAKI OLUR

biggerChild = heapList.get(leftChildIndex);

}

else if(heapList.get(leftChildIndex).getMahalle().getMahalleNufusu() < heapList.get(rightChildIndex).getMahalle().getMahalleNufusu()){ //SAGDAKI COCUGUN NUFUSU DAHA BUYUKSE BUYUK COCUK SAGDAKI OLUR

biggerChild = heapList.get(rightChildIndex);

}

else{ //ELSE DURUMUNDE NULL DONDURUR

return null;

}

return biggerChild; //BUYUK COCUGUN GONDERILMESI

}

public void heapElemanlariYazdirma(){ //FOR DONGUSUYLE HEAPTEKI ELEMANLAR ARRAYLISTTE DOLASILARAK YAZDIRILIR

for(int a=1 ; a<heapList.size() ; a++){

System.out.println(heapList.get(a).getMahalle().toString());

}

}

public boolean isEmpty(){ //ARRAYLISTIN SIZE I 1E ESIT VE KUCUKSE TRUE DONDURUR CUNKU 0.INDEXE NULL EKLEMISTIK

return (heapList.size()<=1);

}

public Mahalle peek(){ //ROOTTAKI ELEMANIN VERILERI GONDERILIR AMA DEGISIKLIK YAPILMAZ

if(!isEmpty()) return heapList.get(1).getMahalle();

else return null;

}

public class Node{ //NODE CLASSI

private Mahalle mahalle; //MAHALLE NESNESI

public Node(Mahalle mahalle){ //CONSTRUCTOR VE GETTER SETTER

this.mahalle = mahalle;

}

public Mahalle getMahalle() {

return mahalle;

}

public void setMahalle(Mahalle mahalle) {

this.mahalle = mahalle;

}

}

}

MAHALLE CLASSI

package maxheap;

public class Mahalle { //HEAP E EKLENECEK MAHALLE NESNELERININ OLUSTURULDUGU CLASS

private String mahalleAdi; //MAHALLE ADI VE NUFUSU VERILERI BULUNUYOR

private int mahalleNufusu;

public Mahalle(String mahalleAdi , int mahalleNufusu){ //CONSTRUCTOR

this.mahalleAdi = mahalleAdi;

this.mahalleNufusu = mahalleNufusu;

}

public String getMahalleAdi() { //GETTER SETTER VE TOSTRING METODLARI

return mahalleAdi;

}

public void setMahalleAdi(String mahalleAdi) {

this.mahalleAdi = mahalleAdi;

}

public int getMahalleNufusu() {

return mahalleNufusu;

}

public void setMahalleNufusu(int mahalleNufusu) {

this.mahalleNufusu = mahalleNufusu;

}

@Override

public String toString(){

return "Mahalle ADI: " + mahalleAdi+

" Mahalle NÜFUSU: " + mahalleNufusu;

}

}

MAIN CLASSI

package maxheap;

//PROJEDE ISTENEN MAXHEAP ICIN MAIN CLASSI VE MAXHEAP CLASSI BURADADIR

public class MAINCLASS {

//SECTIGIM MAHALLELER VE NUFUSLARI

static final String[] mahalleAdlari = {"ERZENE", "KAZIMDİRİK","YEŞİLOVA","ATATÜRK","İNÖNÜ","MEVLANA","EVKA 3","ERGENE","KIZILAY","MERİÇ"};

static final int[] mahalleNufuslari ={35135,33934,31008,28912,25778,25492,20445,11245,15795,8394};

static final int cikarilacakMahalleSayisi =3;

public static void main(String[] args) {

HEAP heap = new HEAP(); //MAXHEAP CLASSINDAN NESNE OLUSTURDUM

for(int a = 0 ; a<mahalleAdlari.length ; a++){ //FOR DONGUSUYLE DIZI DOLASILARAK ELEMANLAR HEAP E YENI NESNE OLUSTURULARAK EKLENIYOR

heap.add(new Mahalle(mahalleAdlari[a], mahalleNufuslari[a]));

}

System.out.println("-----------TUM MAHALLELER EKLENDIKTEN SONRA HEAP-----------");

heap.heapElemanlariYazdirma();

System.out.println("\n----------- HEAPTEN MAX NUFUSLU 3 MAHALLE CIKARILMASI -----------");

for(int a= 1 ; a<=cikarilacakMahalleSayisi ;a++){ //MAXHEAPTEKI DELETE METODU KULLANILARAK HEAPTEN ELEMANLAR SILINIR

System.out.println("HEAPTEN CIKARILAN EN BUYUK "+a+".MAHALLE BILGILERI;");

System.out.println(heap.delete());

}

}

}

## 3.b Max Heap düğüm yerleştirme

public void add(Mahalle mahalle){ //HEAP E GELEN MAHALLE CLASSINDAN NESNENIN EKLENMESI

Node newNode = new Node(mahalle);

if(heapList.isEmpty()){ //LISTE BOSSA 0.INDEXE NULL 1.INDEXE GELEN MAHALLE NESNESI ARRAYLISTTE ADD METODU ILE EKLENIR

heapList.add(null);

heapList.add(1,newNode);

}

else{ //HEAPLIST BOS DEGILSE CALISACAK KISIM

heapList.add(newNode); //ARRAYLISTE ADD METODU ILE SONA DOGRU ELEMANLAR EKLENIR VE EKLENEN ELEMANLARIN HEAP YAPISINI BOZMAMASI ICIN YAPRAKTAN ROOT A DOGRU DUZENLENMESI GEREKIR

yukariyaDogruDuzenleme(); //HEAPIN YUKARIYA DOGRU DUZENLENMESI

}

}

private void yukariyaDogruDuzenleme(){ //ELEMAN EKLENDIGINDE MAXHEAP YAPISININ KORUNMASI ICIN GEREKLI OLAN METOD

int lastElementIndex = heapList.size()-1; //SON ELEMANIN INDEXI BULUNUR

int parentIndex =lastElementIndex/2; //PARENTININ INDEXI BULUNUR

Node lastElement = heapList.get(lastElementIndex);

while(lastElementIndex>0 && parentIndex>0 && heapList.get(parentIndex).getMahalle().getMahalleNufusu() < lastElement.getMahalle().getMahalleNufusu()){ //INDEXLER ARRAYLISTTEN TASMIYORSA VE YAPRAKTAKI ELEMANIN NUFUSU PARENTINDAN BUYUK OLDUGU SURECE CALISACAK WHILE DONGUSU

//BUYUK OLDUKCA CALISACAK OLMASININ SEBEBI MAXHEAP YAPISINDA ROOTTA EN BUYUK ELEMAN BULUNUR

heapList.set(lastElementIndex , heapList.get(parentIndex)); //PARENT ASAGI INDIRILIR VE INDEX DEGERLERI GUNCELLENIR

lastElementIndex = parentIndex;

parentIndex = parentIndex/2;

}

heapList.set(lastElementIndex, lastElement); //WHILE DONGUSUNDEN CIKILNCA BULUNAN INDEX E YAPRAKTAKI ELEMAN EKLENIR

}

## 3.c Heap bilgi çekme

### 3.c.1 Kaynak Kod

public Mahalle delete(){ //HEAPTEN ELEMAN SILMEYI SAGLAYAN METOD

if(heapList.size() ==2){ //HEAPTE 1 TANE ELEMAN VARSA CUNKU 0.INDEXE NULL ATAMISTIK CALISACAK METOD

Node nodeToDelete = heapList.get(1); //SILINECEK ELEMAN DEGISKENE ATANIR VE 1.INDEXTE ELEMAN SILINIP VERISI GONDERILIR

heapList.remove(1);

return nodeToDelete.getMahalle();

}

else if(heapList.size()<2){ //SIZE 2DEN KUCUKSE ELEMAN YOKTUR

System.out.println("NO ELEMENT IN THE HEAP");

return null;

}

Node nodeToDelete = heapList.get(1); //YUKARIDAKI BLOKLARDAN RETURNLE CIKIS YAPMAZSA CALISACAK KISIMLAR

int lastElementIndex = heapList.size()-1; //SON EKLENEN ELEMANIN INDEXI ALINIR

heapList.set(1,heapList.get(lastElementIndex) ); //ILK YANI ROOTTAKI ELEMAN DEGISKENE ATANMISTI BURADA ONUN YERINE SON YAPRAKTAKI ELEMAN GELIR

heapList.remove(lastElementIndex); //SON YAPRAKTAKI ELEMAN SUAN ROOTTA OLDUGU ICIN O KISIM SILINIR

asagiyaDogruDuzenleme(); //HEAP YAPISI BOZULMAMASI ICIN ROOTTAKI ELEMAN COCUKLARIYLA KARSILASTIRILIP GEREKLI DEGISIKLIKLER YAPILARAK UYGUN DUZEN SAGLANIR

return nodeToDelete.getMahalle(); //SILINEN ELEMANIN GONDERILMESI

}

private void asagiyaDogruDuzenleme(){ //ROOTTAKI YANI MAXHEAPTEKI EN BUYUK ELEMAN SILININCE YERINE YAPRAKTAKI ELEMAN GETIRILIR VE YAPI TEKRARDAN DUZENLENIR

int firstElementIndex =1;

Node firstElement = heapList.get(firstElementIndex); //ILK ELEMAN SUAN YAPRAKTAKI ELEMAN

Node biggerChild = buyukCocukBulma(firstElementIndex); //MAX HEAP YAPISINDA DAHA BUYUK OLAN YUKARIDA OLACAGI ICIN BUYUK COCUK YUKARIYA GECER ONUN ICIN BUYUK COCUK BULUNUR

int biggerChildIndex = heapList.indexOf(biggerChild); //BUYUK COCUGUN INDEXI

while(biggerChild !=null && firstElement.getMahalle().getMahalleNufusu() < biggerChild.getMahalle().getMahalleNufusu()){ //YAPRAGA GIDILMEMISSE VE BUYUK COCUGUN NUFUSU PARENTTAN BUYUK OLDUGU SURECE YER DEGISTIRILIR

heapList.set(firstElementIndex, biggerChild); //BUYUK COCUK YUKARIYA CIKTI

heapList.set(biggerChildIndex, firstElement); //PARENTI COCUGUN ESKI YERINE GECTI

firstElementIndex = heapList.indexOf(firstElement); //YENI INDEX DEGERLERINE GORE ILK BASTAKI PARENT TEKRAR COCUKLARIYLA KARSILASTIRILACAK BUNUN ICIN TEKRARDAN COCUKLARIN BUYUK OLANI BULUNUR

firstElement = heapList.get(firstElementIndex);

biggerChild = buyukCocukBulma(firstElementIndex);

biggerChildIndex = heapList.indexOf(biggerChild);

}

}

private Node buyukCocukBulma(int parentIndex){ //BUYUK COCUGUN BULUNDUGU METOD

int leftChildIndex = 2\*parentIndex; //FORMULE UYGUN SEKILDE COCUKLARIN INDEXLERININ BULUNMASI

int rightChildIndex = 2\*parentIndex +1;

Node biggerChild;

if((leftChildIndex >=heapList.size() || rightChildIndex >=heapList.size())) //ARRAYLISTTEN TASIYORSA NULL DONDURUR

return null;

if(heapList.get(leftChildIndex).getMahalle().getMahalleNufusu() > heapList.get(rightChildIndex).getMahalle().getMahalleNufusu()){ //SOLDAKI COCUGUN NUFUSU DAHA BUYUKSE BUYUK COCUK SOLDAKI OLUR

biggerChild = heapList.get(leftChildIndex);

}

else if(heapList.get(leftChildIndex).getMahalle().getMahalleNufusu() < heapList.get(rightChildIndex).getMahalle().getMahalleNufusu()){ //SAGDAKI COCUGUN NUFUSU DAHA BUYUKSE BUYUK COCUK SAGDAKI OLUR

biggerChild = heapList.get(rightChildIndex);

}

else{ //ELSE DURUMUNDE NULL DONDURUR

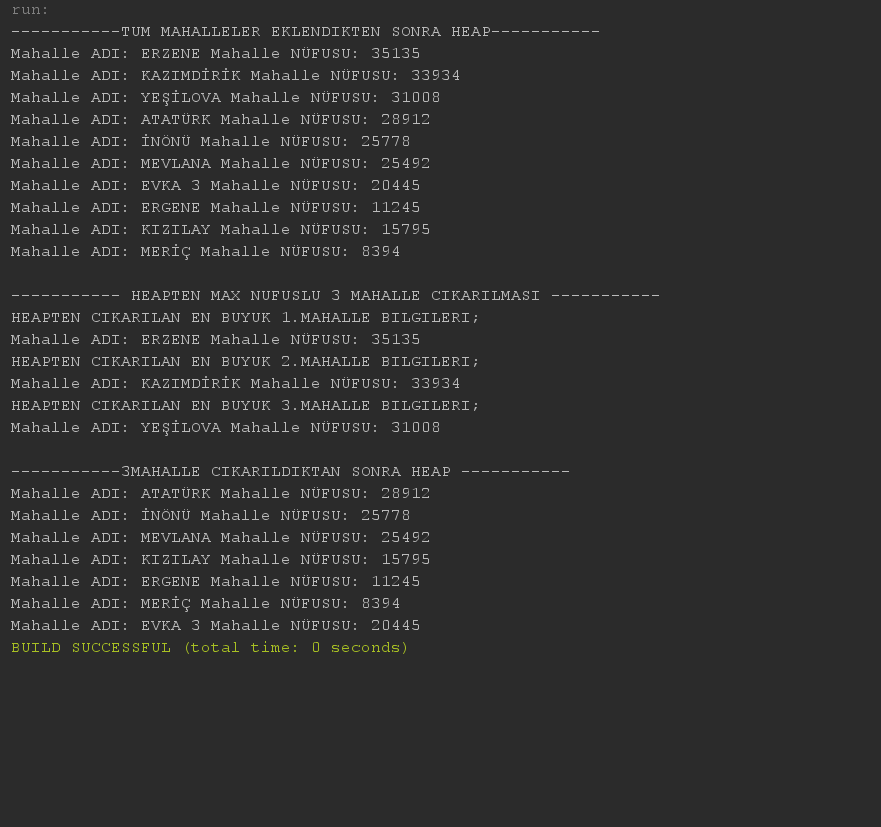
return null;

}

return biggerChild; //BUYUK COCUGUN GONDERILMESI

}

### 3.c.2 Ekran görüntüleri



## 4.a Simple sorting veya Advanced sorting algoritması

package bubblesort;

import java.util.Random;

//SIMPLE SORTING DEN BUBBLE SORT UN KODUNU YAZDIM

public class MAINCLASS {

static final int diziBoyutu =10; //ISTEDIGIMIZ DEGERLER GIRILEBILIR BEN BU DEGERLERI GIRDIM

static final int sayiAraligi = 100;

public static void main(String[] args) {

int[] array = new int[diziBoyutu]; //DIZI BOYUTUNDA YANI KAC TANE SAYI OLACAKSA ONLARIN BOYUTUNDA DIZI OLUSTURULDU

Random r = new Random();

for(int a=0 ; a<array.length ;a++){ //RANDOM SAYILAR OLUSTURULARAK FOR DONGUSUYLE DIZIYE EKLENDI

array[a] = r.nextInt(sayiAraligi);

}

System.out.println("-----DIZI SIRALANMADAN ONCEKI GOSTERIMI-----");

printTheArray(array);

bubbleSort(array);

System.out.println("-----DIZI SIRALANDIKTAN SONRAKI GOSTERIMI-----");

printTheArray(array);

}

public static void printTheArray(int[] array){ //DIZININ FOR DONGUSU KULLANILARAK ELEMANLARININ YAZDIRILMASI

for(int a=0; a<array.length ;a++){

System.out.println("DİZİNİN "+(a+1)+". ELEMANI: " + array[a]);

}

}

public static void bubbleSort(int[] array){ //BUBBLE SORTTA ELEMANLAR IKILI OLARAK KARSILASTIRILIR VE 1 TUR SONUCUNDA 1 ELEMAN GITMESI GEREKEN YERE YANI SONA GIDER BU DURUMDA CHANGED TRUE OLUR

//CHANGED YANI LISTE SIRALI OLDUGUNDA FALSE VERECEGI ICIN WHILE DONGUSUYLE ELEMANLAR SIRALANISA KADAR CALISIR VE SONUCUNDA SIRALANMIS OLURLAR

boolean isChanged =true;

while(isChanged){ //TRUE OLDUGU SURECE CALISACAK WHILE DONGUSU

isChanged =false; //FOR DONGUSUNE GIRMEDEN ONCE FALSE ATANIR DONGUDEKI IF BLOGUNA GIRMEZSE SIRALANMIS OLUR FALSE AYNEN KALIR VE WHILE DONGUSU DE BITMIS OLUR

for(int a=0 ; a<array.length-1 ;a++){ //ELEMANLARIN DOLASILMASI FAKAT UZUNLUGUN 1 KUCUGU KADAR CALISIR CUNKU SON KARSILASTIRMADA SONDAN BIR ONCEKI VE SONUNCU ELEMAN KARSILASTIRILIR TEKRAR KARSILASTIRILMASI YAPILIRSA SONUNCU ELEMANLA HERHANGI BIRISI KARSILASTIRILAMAZ BUNDAN DOLAYI 1 EKSIGI

if(array[a] > array[a+1]){ //EN KUCUK ELEMANIN SOLDA OLMASINI ISTEDIGIM ICIN SOLDAKI ELEMAN BUYUKSE SAGDAKINDEN BLOK CALISIR

int temp = array[a+1];

array[a+1] = array[a]; //YER DEGISTIRME ICIN SAGDAKI ELEMANIN KAYBOLMAMASI ICIN SAGDAKI ELEMAN DEGISKENE ATANDI VE SOLDAKI SAGA GECIRILDI SOLDAKININ YERINE DE DEGISKENDEKI VERI KULLANILARAK GECIRILIR

array[a] = temp;

isChanged =true; //YER DEGISTIRME GERCEKLESTIGI ICIN TRUE YA DONER

printTheArray(array); //DEBUG ICIN KULLANIYORUM

}

}

}

}

}

## 4.b Sıralama algoritması ile karşılaştırılması

bubble sort ->N2

quick sort -> log2N en kötü durumda -> N2

quicksort hızlı bir algoritmadır ancak uygulanması karmaşıktır, küçük veri grupları için mantıklı değildir ama büyük veriler için oldukça mantıklıdır. Bubble sort ise yavaş bir algoritmadır fakat uygulanması kolaydır ve küçük veri grupları için quicksort a göre daha mantıklıdır.

## 4.c Görselleştirme araçları

Etkileşimli görselleştirme araçları bence kod yazmaya başlamadan önce kullanılması gereklidir çünkü verilerin bağlantısını ve yapılacak işlemin mantığını görsel olarak görebiliyoruz ve farklı durumlarda inceleyebiliyoruz. Video ise kesinlikle gece izlenmemeli sağlığa zararlı. Video ise yazılan kodun hangi mantıkta düzenleme yapıldığını görmek açısından faydalı olabileceğini düşünüyorum.

# Özdeğerlendirme Tablosu

**Özdeğerlendirme Tablosu**

|  |  |  |  |
| --- | --- | --- | --- |
| **Proje 3 Maddeleri** | **Not** | **Tahmini Not** | **Açıklama** |
| **1 a) Ağaç (Yemek Sipariş Ağacı Oluşturma)** | **10** | **10** | **YAPILDI** |
| **1 b) Derinlik Bulma, Ağacı Listeleme** | **10** | **10** | **YAPILDI** |
| **1 c) Arama ve Listeleme** | **10** | **10** | **YAPILDI** |
| **1 d) Ağaçta bir yemek türüne indirim yapma** | **10** | **10** | **YAPILDI** |
| **2) Hash Tablosu** | **15** | **15** | **YAPILDI** |
| **3) Yığın Ağacı (Heap)** | **15** | **15** | **YAPILDI** |
| **4) Sıralama Algoritmaları** | **20** | **20** | **YAPILDI** |
| **5) Özdeğerlendirme Tablosu** | **10** | **10** | **YAPILDI** |
| **Toplam** | **100** | **100** | **YAPILDI** |

**Açıklama kısmında yapıldı, yapılmadı bilgisi ve hangi maddelerin nasıl yapıldığı (ve nelerin yapılmadığı / yapılamadığı) yazılmalıdır. Tahmini not kısmına da ilgili maddeden kaç almayı beklediğinizi yazmalısınız.**