Veri Yapıları Dersi Proje Raporu

Ahmet Ercan Ünal:05-13-212

Erkin Alp Güney:05-12-233

Proje 3(A)

İnsan Kaynakları Yönetimi

Eleman Bakma Başvurma ve Arama Sistemi

Gerçekleştirilen Platfrom ve Dil

**Visual Studio 2010 – Framework: 4.0  
MonoDevelop/Mono 4.0**

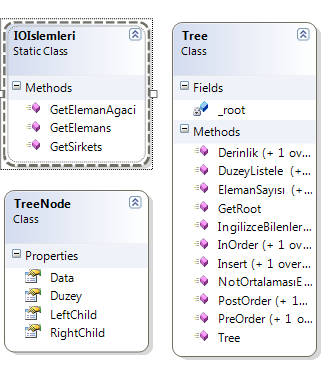
**Dil: C#.Net**

Problemin Kısa Tanımı

İnsan kaynakları sistem, İş ilanı yayınlama, bu ilanları güncelleme veya silme, Yine bu iş ilanlarına elemanların başvuru yapması.

Kullanılan Sınıfların ve Metotların Tanımı

Program başlangıçta FrmMain.cs formunu çağırıyor, Ağaç sınıfım ve Hashtable değişkenlerim public erişim belirleyicilerimle burada tutuluyor. Veri yapılarını anlatırken tekrar değineceğim buraya. Metotları açıklarken daha çok açıklama satırlarından yararlanacağım. Çok fazla metotlarım olduğu için önemli gördüğüm ağaç sınıfını ve IOIslemleri sınıfını anlatacağım.



IOIslemleri.cs

using System;

using System.Collections;

using System.Collections.Generic;

using System.IO;

using System.Text;

using System.Windows.Forms;

namespace Proje3A

{

public static class IOIslemleri

{

//Eleman Agaci Üretme

public static Tree GetElemanAgaci()

{

Tree agac = new Tree();

List<Eleman> lst = GetElemans();

foreach (Eleman item in lst)

{

agac.Insert(item);

}

return agac;

}

//WinForm kullandığımdan dolayı, DataGriddin DataSource Özelliğini Kullanarak

//Elemanları Listeledim, Tabi Burası Soruya Dahil Olmayan Kısmı, Ağaç İşlemlerinde

//Ayrıca Yetten Elemanları Eklettirmeyi Yaptım, Ağaç Sınında Konuya Değineceğim

public static List<Eleman> GetElemans()

{

//eleman.txt dosyasında ki her satırı string[] dizisine attım, ismi lines

var lines = File.ReadAllLines("eleman.txt");

//Datasource için Listtem oluşturuldu ve ilgili string parçalalama

//Düzenlemeler yapılarak Listem doldurulacaktır.

List<Eleman> lstElemanlar = new List<Eleman>();

int kisi = -1;

foreach (string line in lines)

{

if (String.IsNullOrEmpty(line)) continue;

string[] parcala = line.Split('|');

if (parcala[0].Split(':')[0] == "KİŞİ")

{

Eleman eleman = new Eleman();

eleman.Ad = parcala[0].Replace("KİŞİ:", "");

eleman.Adres = parcala[1].Replace("Eleman Adresi ", "");

eleman.Telefon = parcala[2].Replace("Telefon ", "");

eleman.Uyruk = parcala[3];

eleman.EPosta = parcala[4].Replace("E-Posta ", "");

eleman.Sehir = parcala[5];

int tmp = 0;

int.TryParse(parcala[6], out tmp);

eleman.DogumYili = tmp;

eleman.Medenidurumu = parcala[7];

eleman.YabanciDil = parcala[8];

eleman.Hobi = parcala[9];

eleman.Referanslar = parcala[10];

lstElemanlar.Add(eleman);

kisi++;

}

else if (parcala[0].Split(':')[0] == "OKUL")

{

Okul okul = new Okul();

okul.OkulAdi = parcala[0].Replace("OKUL:", "");

okul.Bolum = parcala[1];

int tmp = 0;

int.TryParse(parcala[2], out tmp);

okul.BaslangicYili = tmp;

tmp = 0;

int.TryParse(parcala[3], out tmp);

okul.BitisYili = tmp;

tmp = 0;

int.TryParse(parcala[4], out tmp);

okul.NotOrtalamasi = tmp;

lstElemanlar[kisi].Okullar.Add(okul);

}

else if (parcala[0].Split(':')[0] == "DENEYİM")

{

Deneyim deneyim = new Deneyim();

deneyim.SirketAdi = parcala[0].Replace("DENEYİM:", "");

deneyim.Adres = parcala[1];

deneyim.Pozisyon = parcala[2];

lstElemanlar[kisi].Deneyim.Add(deneyim);

}

}

return lstElemanlar;

}

//Her Bir Satır Tek Tek Okunarak, Hastable'lın Key'ine Eklenicektir.

public static Hashtable GetSirkets()

{

var lines = File.ReadAllLines("sirket.txt");

Hashtable hashtable = new Hashtable();

Sirket sirket;

foreach (string line in lines)

{

string[] parcala = line.Split('|');

sirket = new Sirket()

{

SirketAdi = parcala[0],

Adres = parcala[1],

Telefon = parcala[2],

Faks = parcala[3],

EPosta = parcala[4]

};

//Hashtable'lın value kısmı Şirketlerin İş İlanlarını Tutacaktır.

hashtable.Add(sirket,new ArrayList());

}

return hashtable;

}

}

}

Tree.cs ve TreeNode.cs

Kodları paylaşmadan önce genel olarak Recursive fonksiyonlar kullanmaya çalıştım. Genelde işlerimi kolaylaştırmıştır.

using System;

using System.Collections.Generic;

using System.Text;

namespace Proje3A

{

public class TreeNode

{

public Eleman Data { get; set; }

public TreeNode LeftChild { get; set; }

public TreeNode RightChild { get; set; }

public int Duzey { get; set; }

}

public class Tree

{

private TreeNode \_root;

public Tree() { \_root = null; }

public TreeNode GetRoot()

{ return \_root; }

#region Ekleme

//Node Eklemek İçin Kullanılan Method

public void Insert(Eleman data)

{

TreeNode node = new TreeNode() { Data = data, Duzey = 0 };

Insert(node);

}

//Asıl Ekleme İşlemlerinin Yapıldığı Method

private void Insert(TreeNode localNode)

{

if (\_root == null) { \_root = localNode; return; }

TreeNode current = \_root;

TreeNode parent;

while (true)

{

localNode.Duzey++;

parent = current;

if (localNode.Data.Ad.CompareTo(current.Data.Ad) ==-1)

{

current = current.LeftChild;

if (current == null)

{

parent.LeftChild = localNode;

return;

}

}

else

{

current = current.RightChild;

if (current == null)

{

parent.RightChild = localNode;

return;

}

}

}

}

#endregion

#region Elemanların Listelenmesi

//Kullanıcının Görebileceği Metot

public void PreOrder()

{

FrmElemaninListelenmesi frm = new FrmElemaninListelenmesi();

PreOrder(\_root,frm);

frm.ShowDialog();

}

//Okunan Düğümler Gönderdiğim Forma Eklenir

private void PreOrder(TreeNode node, FrmElemaninListelenmesi frm)

{

if(node==null) return;

frm.Source.Add(node.Data);

PreOrder(node.LeftChild,frm);

PreOrder(node.RightChild, frm);

}

public void InOrder()

{

FrmElemaninListelenmesi frm = new FrmElemaninListelenmesi();

InOrder(\_root,frm);

frm.ShowDialog();

}

private void InOrder(TreeNode node, FrmElemaninListelenmesi frm)

{

if (node == null) return;

InOrder(node.LeftChild, frm);

frm.Source.Add(node.Data);

InOrder(node.RightChild, frm);

}

public void PostOrder()

{

FrmElemaninListelenmesi frm = new FrmElemaninListelenmesi();

PostOrder(\_root,frm);

frm.ShowDialog();

}

private void PostOrder(TreeNode node, FrmElemaninListelenmesi frm)

{

if (node == null) return;

PostOrder(node.LeftChild, frm);

PostOrder(node.RightChild, frm);

frm.Source.Add(node.Data);

}

#endregion

#region Not Ortalaması En Az 90 Olanlar

public void NotOrtalamasıEnAz90()

{

FrmElemaninListelenmesi frm = new FrmElemaninListelenmesi();

NotOrtalamasıEnAz90(\_root,frm);

frm.ShowDialog();

}

private void NotOrtalamasıEnAz90(TreeNode node, FrmElemaninListelenmesi frm)

{

if (node == null) return;

NotOrtalamasıEnAz90(node.LeftChild, frm);

bool flat = false;

foreach (Okul okul in node.Data.Okullar)

{

if (okul.NotOrtalamasi >= 90)

{

flat = true;

break;

}

}

if(flat) frm.Source.Add(node.Data);

NotOrtalamasıEnAz90(node.RightChild, frm);

}

#endregion

#region İngilizce Bilenler

public void IngilizceBilenler()

{

FrmList frm = new FrmList();

frm.listBox1.Items.Add("İngilizce Bilenler");

IngilizceBilenler(\_root, frm);

frm.ShowDialog();

}

private void IngilizceBilenler(TreeNode node, FrmList frm)

{

if (node == null) return;

IngilizceBilenler(node.LeftChild, frm);

if (node.Data.YabanciDil.ToLower().Contains("ingilizce"))

frm.listBox1.Items.Add(string.Format("Adı :{0}\tYabancı Diller:{1}", node.Data.Ad, node.Data.YabanciDil));

IngilizceBilenler(node.RightChild, frm);

}

#endregion

#region İsimlerin Düzeyler İle Birlikte Listelenmesi

public void DuzeyListele()

{

FrmList frm = new FrmList();

frm.listBox1.Items.Add("İsimlerin Düzeylerle Birlikte Listelenmesi");

DuzeyListele(\_root,frm);

frm.ShowDialog();

}

private void DuzeyListele(TreeNode node, FrmList frm)

{

if (node == null) return;

DuzeyListele(node.LeftChild, frm);

frm.listBox1.Items.Add(string.Format("Düzey:{0} Adı :{1}",node.Duzey, node.Data.Ad));

DuzeyListele(node.RightChild, frm);

}

#endregion

#region Derinlik ve Eleman Sayısı

public int Derinlik()

{

return Derinlik(\_root);

}

private int Derinlik(TreeNode node)

{

if (node == null) return -1;

int tmp1 = 0, tmp2 = 0;

if (node.LeftChild != null) tmp1 = Derinlik(node.LeftChild);

if (node.RightChild != null) tmp2 = Derinlik(node.RightChild);

tmp1 = (node.Duzey > tmp1) ? node.Duzey : tmp1;

tmp2=(node.Duzey > tmp2) ? node.Duzey : tmp2;

return (tmp1 > tmp2) ? tmp1 : tmp2;

}

public int ElemanSayısı()

{

return ElemanSayısı(\_root);

}

private int ElemanSayısı(TreeNode node)

{

if (node == null) return 0;

int tmp = 1;

if (node.LeftChild != null) tmp += ElemanSayısı(node.LeftChild);

if (node.RightChild != null) tmp += ElemanSayısı(node.RightChild);

return tmp;

}

#endregion

}

}

**Diğer Metotlarım İş İlanı oluşturmak, İlanı Silmek, İlanları Silmek, İşe Almak, İlanları Güncellemek…**

Genel açıklama olarak karşılaştırma yaptığım bütün sınıflarlar da IComparable sınıfından türeterek karşılaştırabilir yaptım. Böyleyce hem metotlarım da hem de Hashtable da key değeri üzerinden value’ya ulaşabildim. Sadece Örnek Bir Kod paylaşıcam..

**İş İlanına Başvurma Kodu**

private void btnBasvur\_Click(object sender, EventArgs e)

{

if (dgIsIlanlari.Rows.Count == 0)

{

MessageBox.Show("Şirketin İş İlanı Bulunmamaktadır");

return;

}

Sirket etkinSirket = (Sirket) dgSirketler.CurrentRow.DataBoundItem;

IsIlani etkinIlan = (IsIlani) dgIsIlanlari.CurrentRow.DataBoundItem;

if (etkinIlan.BasvuranKisi == null)

{

for (int i = 0; i < ((ArrayList) HTable[etkinSirket]).Count; i++)

{

if (etkinIlan.ElemaninOzellikleri ==

((IsIlani) ((ArrayList) HTable[etkinSirket])[i]).ElemaninOzellikleri)

{

((IsIlani) ((ArrayList) HTable[etkinSirket])[i]).BasvuranKisi = BasvuranKisi;

List<Eleman> lst = new List<Eleman>();

lst.Add(BasvuranKisi);

dgIsIlanıEleman.DataSource = lst;

break;

}

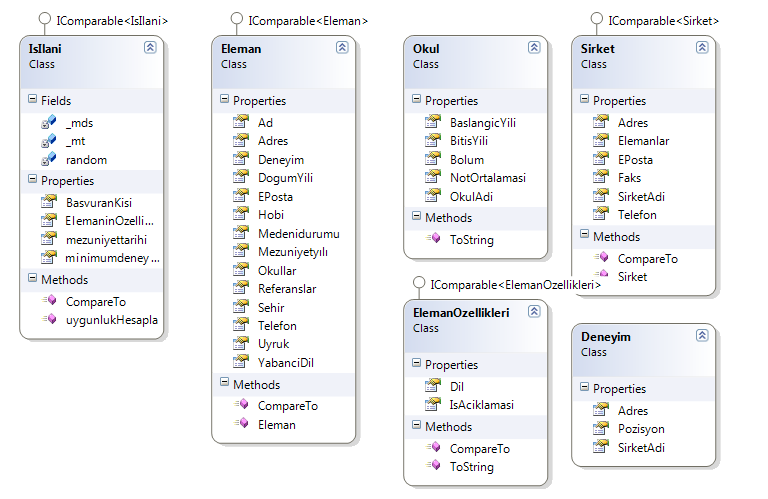
}

}

else MessageBox.Show("Başvurular dolu");

}

Kullandığım Sınıflar

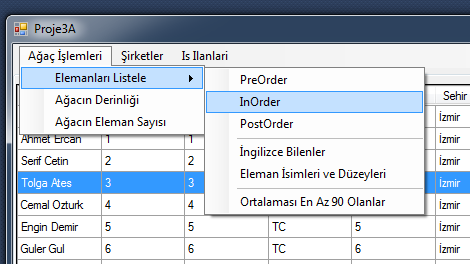


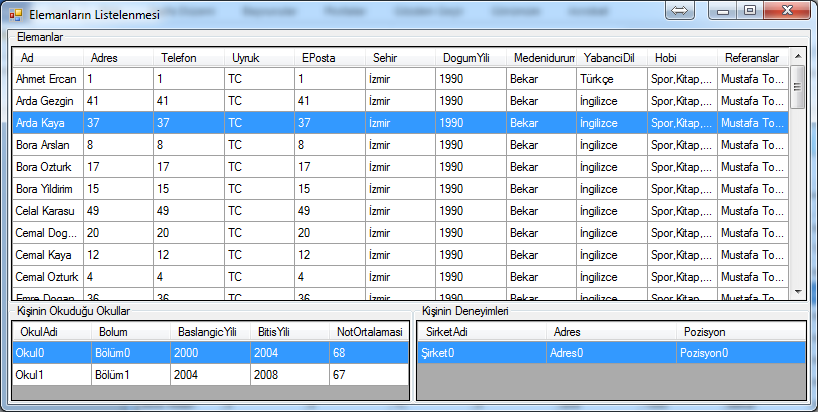
Yazılım Geliştirmek İçin Harcanan Süreler

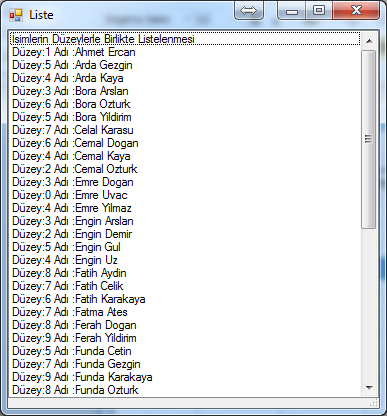
Ahmet Ercan Ünal Yaklaşık 10 Saat  
Erkin Alp Güney Yaklaşık 10 Saat

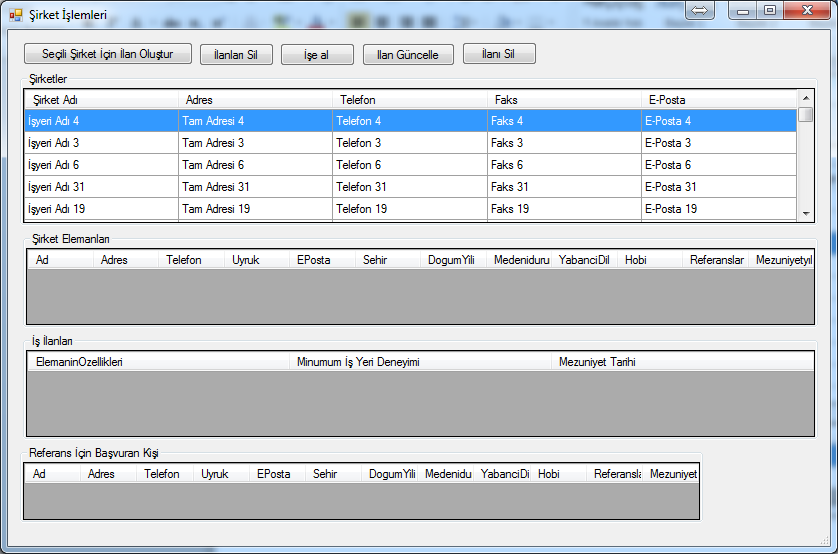
Ek Olarak Ortak Çalışma 4-5 Saat

Kullanıcı Kataloğu









Kullanıcı Kılavuzu

Şirket ve Eleman bilgileri text dosyasından otomatik çekiliyor.

**İş İlanı Ekleme**

Ana pencereden Şirketler listeli görüntülenir. Açılan pencereden İş ilanı oluştur