Program.cs

using BirKelimeBirIslemYeni;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace BirKelimeBirIslem.UI.WinForm

{

static class Program

{

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new MyMDIForm());

}

}

}

KelimeModel.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BirKelimeBirIslem.Model

{

public class KelimeModel

{

public int KelimeID { get; set; }

public string Kelime { get; set; }

public string KelimeAnlam { get; set; }

}

}

KelimeMapping.cs

using BirKelimeBirIslem.Model;

using System;

using System.Collections.Generic;

using System.Data.Entity.ModelConfiguration;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BirKelimeBirIslem.DAL

{

public class KelimeMapping : EntityTypeConfiguration<KelimeModel>

{

public KelimeMapping()

{

HasKey(a => a.KelimeID);

Property(a => a.Kelime)

.HasMaxLength(20)

.IsRequired();

Property(a => a.KelimeAnlam)

.HasMaxLength(500);

}

}

}

KelimeDAL.cs

using BirKelimeBirIslem.Model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Linq.Expressions;

using System.Text;

using System.Threading.Tasks;

namespace BirKelimeBirIslem.DAL

{

public class KelimeDAL

{

BirKelimeBirIslemDbContext db;

public KelimeDAL()

{

db = new BirKelimeBirIslemDbContext();

}

public bool Insert(KelimeModel kelime)

{

db.Kelimeler.Add(kelime);

return db.SaveChanges() > 0;

}

public bool Delete(KelimeModel kelime)

{

db.Kelimeler.Remove(kelime);

return db.SaveChanges() > 0;

}

public KelimeModel GetByID(int kelimeID)

{

KelimeModel kelime = db.Kelimeler.AsNoTracking().SingleOrDefault(a => a.KelimeID == kelimeID);

return kelime;

}

public List<KelimeModel> GetAll(Expression<Func<KelimeModel, bool>> filter = null)

{

if (filter == null)

{

return db.Kelimeler.AsNoTracking().ToList();

}

else

{

return db.Kelimeler.AsNoTracking().Where(filter).ToList();

}

}

}

}

BirKelimeBirIslemDbContext.cs

using BirKelimeBirIslem.Model;

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Data.Entity.ModelConfiguration.Conventions;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BirKelimeBirIslem.DAL

{

class BirKelimeBirIslemDbContext:DbContext

{

public BirKelimeBirIslemDbContext() : base("Server=(localdb)\\YZM3101; Database=YZM2122; uid=kelime; pwd=Remember1")

{

// System.Data.SqlClient.SqlConnection baglan = new System.Data.SqlClient.SqlConnection(@"Data Source=(localdb)\YZM3101;Initial Catalog=YZM2122;Integrated Security=True");

}

public DbSet<KelimeModel> Kelimeler { get; set; }

protected override void OnModelCreating(DbModelBuilder modelBuilder)

{

modelBuilder.Configurations.Add(new KelimeMapping());

modelBuilder.Conventions.Remove<PluralizingTableNameConvention>();

}

}

}

Oyun.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace BirKelimeBirIslem.BLL

{

public class Oyun

{

Random rnd { get; set; }

public int CiftBasamakli { get; set; }

public List<int> TekBasamakli { get; private set; }

public int HedefSayi { get; set; }

CozumBul Cozum { get; set; }

public Oyun()

{

rnd = new Random();

if (TekBasamakli != null) TekBasamakli.Clear();

CiftBasamakli = IkiBasamakliOlustur();

TekBasamakli = TekBasamakliOlustur();

HedefSayi = HedefSayiOlustur();

}

public CozumBul Basla()

{

this.Cozum = new CozumBul(HedefSayi, TekBasamakli, CiftBasamakli);

return Cozum;

}

private int IkiBasamakliOlustur()

{

List<int> sayilar = new List<int>() { 10, 20, 30, 40, 50, 60, 70, 80, 90 };

return sayilar[rnd.Next(0, sayilar.Count)];

}

private List<int> TekBasamakliOlustur()

{

List<int> sayilar = new List<int>();

for (int i = 0; i < 5; i++)

{

sayilar.Add(rnd.Next(1, 9));

}

return sayilar;

}

private int HedefSayiOlustur()

{

return rnd.Next(100, 1000);

}

public override string ToString()

{

StringBuilder sb = new StringBuilder();

sb.Append(Cozum);

return sb.ToString();

}

}

}

Operator.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace BirKelimeBirIslem.BLL

{

public class Operator

{

public enum Operator\_List

{

Topla,

Cikar,

Carp,

Bol

}

}

}

KelimeController.cs

using BirKelimeBirIslem.DAL;

using BirKelimeBirIslem.Model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassLibrary1

{

public class KelimeController

{

KelimeDAL kelimeDAL;

public KelimeController()

{

kelimeDAL = new KelimeDAL();

}

public bool Add(KelimeModel kelime)

{

try

{

return kelimeDAL.Insert(kelime);

}

catch (Exception ex)

{

throw ex;

}

}

public bool Delete(KelimeModel kelime)

{

return kelimeDAL.Delete(kelime);

}

public bool DeleteByID(int kelimeID)

{

KelimeModel kelime = kelimeDAL.GetByID(kelimeID);

return kelimeDAL.Delete(kelime);

}

public KelimeModel GetKelimeByID(int kelimeID)

{

return kelimeDAL.GetByID(kelimeID);

}

public List<KelimeModel> GetKelimeList()

{

return kelimeDAL.GetAll();

}

public List<KelimeModel> GetFilteredList(string param)

{

return kelimeDAL.GetAll(a => a.Kelime.StartsWith(param));

}

}

}

KelimeAra.cs

using BirKelimeBirIslem.Model;

using ClassLibrary1;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BirKelimeBirIslem.BLL

{

public class KelimeAra

{

private List<int> harfmiktar { get; set; }

public int puan;

private int max { get; set; }

internal List<KelimeModel> bulunankelimeler { get; private set; }

public KelimeAra(List<KelimeModel> \_liste, List<string> \_harfler)

{

int adet = 0;

int sira = -1;

max = 0;

harfmiktar = new List<int>();

bulunankelimeler = new List<KelimeModel>();

List<KelimeModel> aramasonuc = new List<KelimeModel>();

foreach (var item in \_liste)

{

string kelime = item.Kelime.ToUpper().Trim().ToString();

adet = 0;

foreach (var item2 in \_harfler)

{

if (kelime.Count() == 0) break;

sira = kelime.IndexOf(item2);

if (sira != -1)

{

adet++;

kelime = kelime.Remove(sira, 1);

sira = -1;

}

}

if (kelime.Count() <= 1)

{

if (kelime.Count() == 1) adet++;

if (adet > max) max = adet;

harfmiktar.Add(adet);

aramasonuc.Add(new KelimeModel { KelimeID = item.KelimeID, Kelime = item.Kelime, KelimeAnlam = item.KelimeAnlam });

}

}

VerileriIsle(aramasonuc);

}

private void VerileriIsle(List<KelimeModel> \_liste)

{

foreach (var item in \_liste.Select((value, i) => new { i, value }))

{

if (harfmiktar[item.i] == max)

{

bulunankelimeler.Add(new KelimeModel { KelimeID = item.value.KelimeID, Kelime = item.value.Kelime, KelimeAnlam = item.value.KelimeAnlam });

}

}

}

public override string ToString()

{

StringBuilder sb = new StringBuilder();

foreach (var item in bulunankelimeler)

{

sb.AppendLine(item.Kelime + " => " + item.KelimeAnlam);

}

string ifade;

switch (max)

{

case 9:

puan = 15;

ifade = "Puan: 15" + Environment.NewLine + "( " + max + " Harf Eşleşti.)";

break;

case 8:

puan = 11;

ifade = "Puan: 11" + Environment.NewLine + "( " + max + " Harf Eşleşti.)";

break;

case 7:

puan = 9;

ifade = "Puan: 9" + Environment.NewLine + "( " + max + " Harf Eşleşti.)";

break;

case 6:

puan = 7;

ifade = "Puan: 7" + Environment.NewLine + "( " + max + " Harf Eşleşti.)";

break;

case 5:

puan = 5;

ifade = "Puan: 5" + Environment.NewLine + "( " + max + " Harf Eşleşti.)";

break;

case 4:

puan = 4;

ifade = "Puan: 4" + Environment.NewLine + "( " + max + " Harf Eşleşti.)";

break;

case 3:

puan = 3;

ifade = "Puan: 3" + Environment.NewLine + "( " + max + " Harf Eşleşti.)";

break;

case 2:

puan = 1;

ifade = "Puan: 1" + Environment.NewLine + "( " + max + " Harf Eşleşti.)";

break;

default:

puan = 0;

ifade = "Puan: 0" + Environment.NewLine + "( " + max + " Harf Eşleşti.)";

break;

}

sb.Append(ifade);

return sb.ToString();

}

}

}

Islem.cs

using System;

using System.Collections.Generic;

using System.Text;

using static BirKelimeBirIslem.BLL.Operator;

namespace BirKelimeBirIslem.BLL

{

public class Islem:Oyun

{

private List<int> SiradakiSayi { get; set; }

private List<Operator\_List> Operatorler { get; set; }

public int DegerHesapla

{

get

{

int deger = SiradakiSayi[0];

for (int i = 1; i < SiradakiSayi.Count; i++)

{

switch (Operatorler[i - 1])

{

case Operator\_List.Topla:

deger += SiradakiSayi[i];

break;

case Operator\_List.Cikar:

deger -= SiradakiSayi[i];

break;

case Operator\_List.Carp:

deger \*= SiradakiSayi[i];

break;

case Operator\_List.Bol:

deger /= SiradakiSayi[i];

break;

}

}

return deger;

}

}

public Islem(int \_siradaki\_sayi)

{

this.SiradakiSayi = new List<int>() { \_siradaki\_sayi };

this.Operatorler = new List<Operator\_List>();

}

public Islem(Islem \_denklembaslangici, Operator\_List \_islem, int \_siradakisayi)

{

this.SiradakiSayi = new List<int>((int[])\_denklembaslangici.SiradakiSayi.ToArray().Clone());

this.SiradakiSayi.Add(\_siradakisayi);

this.Operatorler = new List<Operator\_List>((Operator\_List[])\_denklembaslangici.Operatorler.ToArray().Clone());

this.Operatorler.Add(\_islem);

}

public override string ToString()

{

StringBuilder sb = new StringBuilder();

int siradaki = SiradakiSayi[0];

int \_siradakideger = -1;

for (int i = 1; i < SiradakiSayi.Count; i++)

{

string satir = YeniIslemSatiriOlustur(siradaki, i, ref \_siradakideger);

sb.Append(satir);

sb.Append(Environment.NewLine);

siradaki = \_siradakideger;

}

return sb.ToString();

}

private string YeniIslemSatiriOlustur(int \_simdikisayi, int \_kartsirasi, ref int \_siradakideger)

{

int siradaki\_sayi = SiradakiSayi[\_kartsirasi];

Operator\_List islem = (Operator\_List)Operatorler[\_kartsirasi - 1];

string islemsembol = IslemSembolEkle(islem);

\_siradakideger = YeniDegerHesapla(\_simdikisayi, siradaki\_sayi, islem);

return string.Format("{0} {1} {2} = {3}", \_simdikisayi, islemsembol, siradaki\_sayi, \_siradakideger);

}

private string IslemSembolEkle(Operator\_List Operator)

{

switch (Operator)

{

case Operator\_List.Topla:

return "+";

case Operator\_List.Cikar:

return "-";

case Operator\_List.Carp:

return "\*";

case Operator\_List.Bol:

return "/";

default:

return null;

}

}

private int YeniDegerHesapla(int \_sayi1, int \_sayi2, Operator\_List \_islem)

{

switch (\_islem)

{

case Operator\_List.Topla:

return \_sayi1 + \_sayi2;

case Operator\_List.Cikar:

return \_sayi1 - \_sayi2;

case Operator\_List.Carp:

return \_sayi1 \* \_sayi2;

case Operator\_List.Bol:

return \_sayi1 / \_sayi2;

default:

return -1;

}

}

}

}

CozumBul.cs

using System;

using System.Collections.Generic;

using System.Text;

using static BirKelimeBirIslem.BLL.Operator;

namespace BirKelimeBirIslem.BLL

{

public class CozumBul

{

public List<int> TumSayilar { get; private set; }

public int Hdf2 { get; private set; }

private Islem Islem { get; set; }

private int enYakin = int.MaxValue;

public int puan;

public CozumBul(int \_hedef, List<int> \_tekbasamakli, int \_ikibasamakli)

{

TumSayilar = new List<int>();

this.Hdf2 = \_hedef;

this.TumSayilar.Add(\_ikibasamakli);

this.TumSayilar.AddRange(\_tekbasamakli);

Coz();

}

public void Coz()

{

for (int i = 0; i < TumSayilar.Count; i++)

{

Islem \_denklem = new Islem(TumSayilar[i]);

List<int> arttikliste = KisaListeOlustur(TumSayilar, i);

if (CozumAra(\_denklem, arttikliste))

break;

}

}

private List<int> KisaListeOlustur(List<int> \_eskiliste, int \_sirano)

{

List<int> yeniliste = new List<int>();

for (int i = 0; i < \_eskiliste.Count; i++)

if (i != \_sirano)

yeniliste.Add(\_eskiliste[i]);

return yeniliste;

}

private bool CozumAra(Islem \_denklembaslangici, List<int> \_artikliste)

{

for (int i = 0; i < \_artikliste.Count; i++)

{

foreach (Operator\_List islem in Enum.GetValues(typeof(Operator\_List)))

{

List<int> yeniartikliste = KisaListeOlustur(\_artikliste, i);

int siradakisayi = \_artikliste[i];

if (CozumeKartiEkle(siradakisayi, islem, \_denklembaslangici, yeniartikliste))

return true;

}

}

return false;

}

private bool CozumeKartiEkle(int \_siradakisayi, Operator\_List \_islem, Islem \_denklembaslangici, List<int> \_artikliste)

{

Islem denklem = new Islem(\_denklembaslangici, \_islem, \_siradakisayi);

if (Math.Abs(Hdf2 - denklem.DegerHesapla) <= enYakin)

{

enYakin = Math.Abs(Hdf2 - denklem.DegerHesapla);

this.Islem = denklem;

if (denklem.DegerHesapla == Hdf2) return true;

}

if (\_artikliste.Count == 0)

{

return false;

}

return CozumAra(denklem, \_artikliste);

}

public override string ToString()

{

StringBuilder sb = new StringBuilder();

sb.Append(this.Islem.ToString());

int fark = Math.Abs(this.Islem.DegerHesapla - Hdf2);

string ifade;

switch (fark)

{

case 0:

puan = 10;

ifade = "Puan: 10" + Environment.NewLine + "(Tam Sonuç)";

break;

case 1:

puan = 9;

ifade = "Puan: 9" + Environment.NewLine;

break;

case 2:

puan = 8;

ifade = "Puan: 8" + Environment.NewLine;

break;

case 3:

puan = 7;

ifade = "Puan: 7" + Environment.NewLine;

break;

case 4:

puan = 6;

ifade = "Puan: 6" + Environment.NewLine;

break;

case 5:

puan = 5;

ifade = "Puan: 5" + Environment.NewLine;

break;

case 6:

puan = 4;

ifade = "Puan: 4" + Environment.NewLine;

break;

case 7:

puan = 3;

ifade = "Puan: 3" + Environment.NewLine;

break;

case 8:

puan = 2;

ifade = "Puan: 2" + Environment.NewLine;

break;

case 9:

puan = 1;

ifade = "Puan: 1" + Environment.NewLine;

break;

default:

puan = 0;

ifade = "Puan Alamadınız!" + Environment.NewLine;

break;

}

sb.Append(Environment.NewLine + ifade);

return sb.ToString();

}

}

}