**Navodila za izdelavo seminarske naloge PRO2 2020/21**

**Naslov naloge: *SeminarskaPRO2PriimekIme***

**Nalogo lahko izdelate v konzolnem načinu, v okenskem okolju (Windows Forms ali WPF) ali pa v obliki spletne aplikacije (ASP.NET WEB Application)**

**Oddaja naloge:**

* **V spletno učilnico: arhiviran projekt + baza + dokument v Word-u (4 točke) ( vse arhivirano: zip, rar, 7z, …)**
* **Po oddaji naloge je obvezen tudi zagovor naloge: osebno v šoli, video posnetek po mailu, Skype (po dogovoru) (2 točki).**

***Osnova za seminarsko nalogo sta ena ali dve datoteki tipa XML ali JSON, v katerih je vsaj po 20 vrstic podatkov.***

* ***Vsebini datotek: datoteke je sestavljena iz podatkov iz katerih boste ustvarjali objekte. Zato morate napisati razred (*2 TOČKI*) in v njem*** 
  + ***zasebna polja ali lastnosti/property-ji, ki se ujemajo s podatki iz datoteke.***
  + ***konstruktor,***
  + ***javne metode ali property-ji za dostop do zasebnih polj,***
  + ***prekrivna metoda ToString.***

**Primer vsebine take datoteke:**

Podatki iz PRVE datoteke (če je datoteka tipa XML)

*<?xml version="1.0"?>*

*<ArrayOfKronometer xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">*

*<Kronometer>*

*<Naziv>Toni Baloni</Naziv>*

*<start>2018-12-03T01:41:20.417</start>*

*<cilj>2018-12-03T02:05:13.504</cilj>*

*</Kronometer>*

*<Kronometer>*

*<Naziv>Polde Bibic</Naziv>*

*<start>2018-12-03T01:40:23.434</start>*

*<cilj>2018-12-03T02:22:15.999</cilj>*

*</Kronometer>*

*<Kronometer>*

*<Naziv>Polde Kodri</Naziv>*

*<start>2018-12-03T01:40:13.434</start>*

*<cilj>2018-12-03T02:22:15.999</cilj>*

*</Kronometer>*

*<Kronometer>*

*<Naziv>Toni Ban</Naziv>*

*<start>2018-12-03T01:41:20.417</start>*

*<cilj>2018-12-03T02:05:13.504</cilj>*

*</Kronometer>*

*<Kronometer>*

*<Naziv>Minka Pinka</Naziv>*

*<start>2018-12-03T01:48:20.111</start>*

*<cilj>2018-12-03T02:13:11.754</cilj>*

*</Kronometer>*

*<Kronometer>*

*<Naziv>Tonka Marn</Naziv>*

*<start>2018-12-03T01:41:29.417</start>*

*<cilj>2018-12-03T02:05:13.504</cilj>*

*</Kronometer>*

*</ArrayOfKronometer>...*

**Sestavite program, ki uporablja**

1. bazo podatkov,
2. poljubno lastno generično zbirko
3. lastni enosmerni verižni seznam.

**Program naj prikaže meni s tremi možnostmi** (1. *BAZA, 2. GENERIČNA ZBIRKA, 3. VERIŽNI SEZNAM*), ki operirajo s podatki v vseh treh programskih strukturah. Vsaka od opcij glavnega menija naj ima glede na zahteve seminarske naloge še ustrezno število podopcij (podmenijev). (**SKUPAJ 40 točk**):

1. **BAZA podatkov (tip mdf ali sdf, lahko pa tudi katerikoli drug tip: access, firebird, MySQL, … (13 TOČK) (POZOR: če boste uporabili bazo, ki ne bo tipa mdf oz. sdf, morate v seminarski nalogi razložiti/opisati potrebne objekte za delo s tako bazo in dodati link do spletne strani, kjer lahko dobimo urejevalnik za tako bazo).**
   1. Ustvarite svojo bazo z vsaj dvema tabelama: vsaj eno polje v tabelah naj bo ključno polje tipa *autoincrement*. V ti dve tabeli prenesite podatke iz datotek, ki ste jih ustvarili na začetku.(*2 točki*);
   2. Tabele naj bo možno urejati: dodajanje, spreminjanje, brisanje (*5 točk*).
   3. Uporabite *using* stavke (*1 točka*) in varovalne bloke(*1 točka*);
   4. Dodano naj bodo vsaj 3 poizvedbe tipa *ExecuteReader* in vsaj 3 poizvedbe tipa *ExecuteScalar*(*4 točke*);
2. **LASTNA GENERIČNA ZBIRKA( ali struktura tipa *Dictionary*)(13 TOČK):**

**Uporabite generični razred GeneričnaZbirka, ki smo ga spoznali na predavanjih in vajah**

public class GeneričnaZbirka<T>

{

private T[] elementi; //tabelarično polje

private int velikost; //polje hrani trenutno število podatkov v tabeli

public GeneričnaZbirka(int n = 0) //konstruktor

{ elementi = new T[n]; velikost = n; }//začetna dimenzija tabele/zbirke

public T this[int indeks] //indeksiranje

{

get { return elementi[indeks]; } //dostop do posameznih polj

set { elementi[indeks] = value; } //prirejanje vrednostim poljem

}

//napišimo property, s katerim pridobimo atribut velikost

public int Velikost

{

get { return velikost; }

}

//še get metoda za prodobivanje polja velikost

public int VrneVelikost()

{

return velikost;

}

public void OdstraniVse()

{

elementi = new T[0];

velikost = 0;

}

public void Add(T podatek) //metoda za dodajanje novega elementa

{

Array.Resize(ref elementi, elementi.Length + 1);

elementi[velikost] = podatek; //podatek zapišemo v prvo prosto celico

velikost = velikost + 1; //število zasedenih celic

}

//generična metoda za brisanje celice z določenim indeksom

public void Brisanje(int indeksCelice)

{

if (velikost == 0)

Console.WriteLine("Zbirka je prazna, brisanje NI možno!");

//celico brišemo le, če je njen indeks manjši od dimenzije zbirke

// if (indeksCelice < elementi.Length && indeksCelice >= 0)

else if (indeksCelice < elementi.Length)

{

T[] zacasna = new T[elementi.Length - 1];

int j = 0;

for (int i = 0; i < elementi.Length; i++)

{

if (i != indeksCelice)

{

zacasna[j] = elementi[i];

j++;

}

}

elementi = zacasna;

velikost = velikost - 1;//zmanjšamo velikost zbirke

}

else Console.WriteLine("Brisanje NI možno, ker indeks št

"+indeksCelice+" NE obstaja!");

}

//generična metoda za izpis poljubne zbirke

public void IzpisZbirke()

{

if (velikost == 0)

Console.WriteLine("Zbirka je prazna!");

else

{

Console.WriteLine("Izpis ZBIRKE: ");

for (int i = 0; i < elementi.Length; i++)

Console.WriteLine(elementi[i].ToString() + " ");

Console.WriteLine();

}

}

}

* 1. Elementi zbirke naj bodo **objekti.** Ustvarite jih s pomočjo razreda, ki naj vsebuje vse potrebne atribute, ki ustrezajo zapisu ene od datotek (XML ali JSON), ki ste ju ustvarili na začetku (DESERIALIZACIJA) (1 točka). Razred naj vsebuje še:
     1. Minimalni konstruktor, ki tvori objekt s privzetimi (smiselnimi) vrednostmi; (*1 točka*)
     2. Metodo *public override string ToString()*, ki vse podatke v objektu zapiše v obliki ustreznega niza; (*1 točka*)
     3. Konstruktor s toliko parametri, kot je polj v razredu. Pravilno (smiselno) prirejanje polj preverjajte s pomočjo ustreznih *set* metod (če so podatki nesmiselni, naj bo prirejena vrednost privzeta). V nobenem primeru naj ne bo možno ustvariti objekta z nesmiselnimi podatki; (2 *točki*)
     4. Objektno metodo, ki s pomočjo prekrivne metode *ToString* podatke iz zbirke zapiše nazaj (SERIALIZACIJA) v ustrezno datoteko tipa XML ali JSON. Ime datoteke naj bo enako imenu tabele iz baze; (1 *točka*)
  2. Napišite klasično metodo s katero bi nad generično zbirko realizirali eno od poizvedb, ki ste jih naredili pri tabelah v bazi podatkov v nalogi št. 1(*2 točki*);
  3. Napišite po 5 poizvedb tipa LINQ (5 x Query, 5xLambda) (2 točki)
  4. Napišite generični metodi, ki dobita za vhodni podatek poljubno zbirko tipa *GeneričnaZbirka*. Prva metoda je poljubna, druga pa naj vrne generično zbirko tipa *Stack* ali generično zbirko tipa *Queue*. Druga metoda torej vrne sklad ali pa vrsto. (*2 točki*)
  5. Zbirko uredite s pomočjo ene od metod za urejanje podatkov: kriterij urejanja je odvisen od podatkov, ki jih hranijo objekti, ki so v zbirki (*1 točka*);

1. **LASTNI ENOSMERNI VERIŽNI SEZNAM (6 TOČK):**

**Seznam naj bo ustvarjen iz objektov tvorjenih iz razreda *Vozel*, ki smo ga spoznali na predavanjih in vajah.**

public class Vozel<T>

{

private T podatek; // hranimo podatke tipa T

private Vozel<T> naslednji; // referenca na naslednjega

//začetek seznama je shranjen v objektu Zacetek

private Vozel<T> Zacetek;

public Vozel<T> ZacetekSeznama //lastnost, ki vrne začetek seznama

{

get { return Zacetek; }

}

public Vozel() // privzeti konstruktor nastavi privzete vrednosti

{

this.Vsebina = default(T);

// kazalec na naslednji element

this.Nasl = null;

Zacetek = this;

}

public Vozel(T podatek) //dodatni konstruktor

: this()//dedujemo bazični konstruktor

{

this.Vsebina = podatek;

}

public T Vsebina //Lastnost/property

{

get { return this.podatek; }

set { this.podatek = value; }

}

public Vozel<T> Nasl //Lastnost/property

{

get { return this.naslednji; }

set { this.naslednji = value; }

}

public void Dodaj(T podatek)//Metoda za dodajanje elementa na začetek seznama

{

Vozel<T> nov = new Vozel<T>(podatek);

nov.Nasl = Zacetek;

Zacetek = nov;

}

public void Izpis<T>(Vozel<T> Zacetek) // Metoda za izpis seznama

{

//shranim začetek seznama

Console.Write("ZAČETEK->");

//premikamo se toliko časa, da je naslov naslednjega podatka null

while (Zacetek != null)

{

Console.Write(Zacetek.Vsebina + " ");

Zacetek = Zacetek.naslednji;

}

Console.Write("->KONEC\n");

}

public void Izpis() //običajna metoda za izpis poljubnega seznama

{

//shranim začetek seznama

if (this != null)

//lahko tudi takole: if (Zacetek!=null)

{

//zapomnim si (shranim) začetek seznama

Vozel<T> zacasni = ZacetekSeznama;

Console.Write("ZAČETEK->");

// premikamo se toliko časa, da je naslov naslednjega podatka null

while (zacasni != null)

{

Console.Write(zacasni.Vsebina + " ");

zacasni = zacasni.naslednji;

}

Console.Write("->KONEC\n");

}

else Console.WriteLine("Seznam je prazen!");

}

/\*metoda za dodajanje novega podatka na konec seznama: drugi parameter je klican po referenci, saj se bo lahjo spremenil začetek seznama\*/

public void VstaviNaKonec(T podatek)

{

Vozel<T> kjeSmo = Zacetek;//pomožni objekt za premik po seznamu

while (kjeSmo.Nasl != null)/\*dokler ne pridemo do konca seznama, oz. dokler ne najdemo podatka za brisanje\*/

{

kjeSmo = kjeSmo.Nasl;//prestavimo se na naslednji objekt

}

//ko pridemo iz zanke kazalec tekoči zagotovo kaže na zadnji objekt

Vozel<T> nov = new Vozel<T>();

nov.Vsebina = podatek;

kjeSmo.Nasl = nov;

}

public int PrestejElemente() //Metoda, ki vrne število elementov seznama

{

int stevec = 0;

if (Zacetek != null)

{

Vozel<T> kjeSmo = Zacetek;//začetek seznama

stevec = 1;

while (kjeSmo.Nasl != null)

{

stevec++;

kjeSmo = kjeSmo.Nasl;

}

}

return stevec;

}

/\*metoda ki v seznamu poišče podatek "podatek" in ga odstrani: prvi parameter je klican po referenci, saj se bo lahjo spremenil začetek seznama\*/

public void Izbriši(T podatek)

{

if (Zacetek.Vsebina != null)//če seznam ni prazen

{

//preverimo, če je iskani podatek že na začetku seznama

if (Zacetek.Vsebina.Equals(podatek))

{

if (Zacetek.naslednji != null)

Zacetek = Zacetek.Nasl;

else Zacetek = null;

}

else

{

Vozel<T> tekoči = Zacetek;//dodatni objekt za premik po seznamu

while (tekoči.Nasl != null) /\*dokler ne pridemo do konca seznama, oz. dokler ne najdemo podatka za brisanje\*/

{

//ali se vsebina NASLEDNJEGA objekta ujema z našim podatkom

if (tekoči.Nasl.Vsebina.Equals(podatek))

{

//preverimo, če ne gre slučajno za zadnji podatek v seznamu

if (tekoči.Nasl.Nasl != null)

{

//ker to ni zadnji objekt,prestavimo kazalec za ena naprej

tekoči.Nasl = tekoči.Nasl.Nasl;

}

else tekoči.Nasl = null;/\*ker gre za ZADNJI objekt, nastavimo kazalec na null\*/

break;//izhod iz zanke while

}

else tekoči = tekoči.Nasl;//

}

}

}

}

1. Elementi enosmernega verižnega seznama naj bodo objekti preneseni iz zapisov ene od datotek, ki ste ju ustvarili na začetku, ali iz prej ustvarjene lastne generične zbirke *GeneričnaZbirka* (*1 točka*);
2. No konkretnih podatkih preverite vse objektne metode razreda Vozel (2 točki)
3. Napišite vsaj tri **lastne (NE iz spletne učilnice)** metode (dve generični in eno klasično), s katerima bi nad verižnim seznamom ustvarili vsaj dve poizvedbi (3 *točke*). Metode naj imajo za parameter naslov začetka verižnega seznama;

**Kriteriji za oceno: 0 – 20 = 5, 21 – 24 = 6, 25 – 28 = 7,29 – 32 = 8, 33 – 36 = 9, 37 – 40 = 10**