ZADATAK

U programskom jeziku C# kreirati sledeće tipove:

Javni generički interfejs **IUporediv** koji nalaže implementaciju metode **Uporedi**. Metoda prihvata jednu generičku promenljivu, a povratni tip je **int.**

Javni generički tip **Lista,** gde se garantuje da su elementi **Liste** međusobno uporedivi (**IUporediv**). Tip Lista predstavlja implementaciju jednostruko povezane liste.

Tip **Lista** treba da sadrži:

- Konstruktor koji treba da prihvati vrednost generičkog tipa i da je postavi kao glavu liste.
- Metodu **Add** koja prihvata generičku vrednost i dodaje je u Listu.
- Svojstvo **BrojElemenata,** kojim se može dohvatiti broj elemenata u listi
- Indexer koji vraća element **Liste** koji se nalazi na poziciji koja odgovara zadatom indeksu (redni broj elementa).
- Generičku metodu **Sortiraj,** koji od kreirane liste pravi sortiranu uz pomoć metoda iz Interfejsa **IUporedi**.
- Generičku metodu koja vraća **IEnumerable** odgovarajućeg tipa kojim se prolazi kroz celu listu.

Kreirati dva proizvoljna tipa koja se mogu smestiti u listu, i napisati test funkciju.

```
/// <summary>
/// Genericki interfejs za uporedjivanje dva ganericka tipa.
/// </summary>
/// <typeparam name="T"> Genericki tip. </typeparam>
public interface IUporediv<T>
{
    /// <summary>
    /// Metod za uporedjivanje dva objekta istog generickog tipa.
    /// </summary>
    /// <param name="vrednost"> Referenca na objekat istog tipa </param>
    /// <returns> 1 - prvi objekat ima vecu vrednost,
                  0 - imaju istu vrednost,
                  -1 drugi element ima vecu vrednost
         </returns>
    int Uporedi(T vrednost);
}

/// <summary>
/// Jedan element generickе liste.
/// </summary>
/// <typeparam name="T"> Genericki tip. </typeparam>
public class Element<T>
{
    public T vrednost;         // Vrednost
    public Element<T> sledeci; // Pokazivac na sledeci

    public Element(T v)
    {
        vrednost = v;
    }
}



    /// <summary>
```

```
        /// Indekser koji moze da postavi ili dohvati vrednost odgovarajuceg elementa.
        /// </summary>
        /// <param name="indeks"> Indeks elementa u listi. </param>
        /// <returns> Vrednost genericke promenljive koje se nalazi na "indeks" poziciji.
</returns>
        public T this[int indeks]
        {
            get
            {
                Element<T> tmp = glava;

                for (int i = 0; i < indeks; i++)
                            {
                                tmp = tmp.sledeci;
                            }

                return tmp.vrednost;
            }

            set
            {
                Element<T> tmp = glava;

                for (int i = 0; i < indeks; i++)
                            {
                                tmp = tmp.sledeci;
                            }

                tmp.vrednost = value;
            }
        }

        /// <summary>
        /// Svojstvo koje vraca broj elemenata koji se nalaze u listi.
        /// </summary>
        public long BrojElemenata
        {
            get
            {
                int brojElemenata = 1;

                Element<T> tmp = glava;
                while (tmp.sledeci != null)
                {
                    tmp = tmp.sledeci;
                    brojElemenata++;
                }

                return brojElemenata;
            }
        }

        /// <summary>
        /// Metod koji sortira elemente u listi
        /// Koristi se selection sort kao primer koriscenja indeksera.
        /// </summary>
        public void Sortiraj()
        {
            for (int i = 0; i < BrojElemenata; i++)
            {
                for (int j = i + 1 ; j < BrojElemenata; j++)
                {
                    T v1 = this[i];
                    T v2 = this[j];

                    if (v1.Uporedi(v2) == -1)
                    {
                        this[i] = v2;
                        this[j] = v1;
```

```
                }
            }

        }
    }

    public void Add(T novaVrednost)
    {
        Element<T> tmp = glava;

        while (tmp.sledeci != null)
        {
            tmp = tmp.sledeci;
        }

        tmp.sledeci = new Element<T>(novaVrednost);
    }

    public IEnumerator<T> GetEnumerator()
    {
        return new EnumeratorKrozListu<T>(this);
    }

    System.Collections.IEnumerator System.Collections.IEnumerable.GetEnumerator()
    {
        throw new NotImplementedException();
    }
}



public class EnumeratorKrozListu<T> : IEnumerator<T> where T : class, IUporediv<T>
{
    Lista<T> lista;
    Element<T> trenutni = null;

    public EnumeratorKrozListu(Lista<T> lista)
    {
        this.lista = lista;
    }

    public T  Current
    {
            get { return trenutni.vrednost; }
    }

    public void  Dispose()
    {

    }

    object  System.Collections.IEnumerator.Current
    {
            get { throw new NotImplementedException(); }
    }

    public bool  MoveNext()
    {
        if (trenutni == null)
        {
            trenutni = lista.Glava;
            return true;
        }
        else
        {
            if (trenutni.sledeci != null)
            {
                trenutni = trenutni.sledeci;
                return true;
```

```
            }
            else
                return false;

        }
    }

    public void  Reset()
    {
        trenutni = null;
    }
}

/// <summary>
/// Primer klase koja moze da se smesti u Lista<T>.
/// </summary>
public class Student : IUporediv<Student>
{
    public string ime;
    public string prezime;
    public double prosek;

    public Student(string ime, string prezime, double prosek)
    {
        this.ime = ime;
        this.prezime = prezime;
        this.prosek = prosek;
    }

    public int Uporedi(Student student)
    {
        if (prosek > student.prosek)
            return 1;
        else if (prosek == student.prosek)
            return 0;
        else
            return -1;
    }

    public override string  ToString()
    {
            return ime + " " + prezime + ". Prosek : " + prosek;
    }
}

/// <summary>
/// Primer klase koja moze da se smesti u Lista<T>.
/// </summary>
public class Radnik : IUporediv<Radnik>
{
    public string ime;
    public double plata;

    public Radnik(string ime, double plata)
    {
        this.ime = ime;
        this.plata = plata;
    }

    public int Uporedi(Radnik radnik)
    {
        if (plata > radnik.plata)
            return 1;
        else if (plata == radnik.plata)
            return 0;
        else
            return -1;
    }
```

```
    public override string ToString()
    {
        return ime + ". Plata : " + plata;
    }
}
```