

ДИНАМИЧКИ НИЗОВИ

Статички низови су низови фиксне величине, елементи морају бити истог типа и елементи се не могу додати или уклонити.

```
int[] niz = new int[]{1,2,3,4};
```

ArrayList је динамички низ који садржи листу објеката. Дакле, **ArrayList** је не генеричка колекција објеката чија се величина може динамички повећавати – смањивати.

```
using System;
using System.Collections;

class Osoba {
    string ime;
    int godina;
    public Osoba(string ime, int godina){
        this.ime = ime;
        this.godina = godina;
    }
}

class Program
{
    static void Main(string[] args)
    {
        ArrayList lista1 = new ArrayList() { 1,2,3,4};
        lista1.Add(5);

        Osoba a = new Osoba("Pera Peric", 17);
        Osoba b = new Osoba("Sima Simic", 18);
        int[] niz = new int[] { 1, 2, 3, 4 };
        string s = "skola";

        lista1.Add(a);
        lista1.Add(b);
        lista1.Add(niz);
        lista1.Add(s);
        Console.ReadKey();
    }
}
```

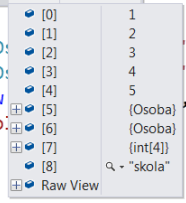
У наведеном примеру, дефинисали смо низ са 4 елемента преко **ArrayList**. Додали смо и 5 елемент.

Међутим, **ArrayList** омогућава додавање објеката који нису истог типа.

```
static void Main(string[] args)
{
    ArrayList lista1 = new ArrayList() { 1,2,3,4};
    lista1.Add(5);

    Osoba a = new Osoba("Pera Peric", 17);
    Osoba b = new Osoba("Sima Simic", 18);
    int[] niz = new int[] { 1, 2, 3, 4 };
    string s = "skola";

    lista1.Add(a);
    lista1.Add(b);
    lista1.Add(niz);
    lista1.Add(s);
    Console.ReadKey();
}
```



List <T> је динамички низ који садржи објекте истог типа. Дакле, **List<T>** је генеричка колекција објеката чија се величина може динамички повећавати – смањивати.

Ако модификујемо претходни пример компајлер ће пријавити грешку јер покушавамо да додамо елементе који нису истог типа.

```
class Program
{
    static void Main(string[] args)
    {
        List<Osoba> lista2 = new List<Osoba>();
        Osoba a = new Osoba("Pera Peric", 17);
        Osoba b = new Osoba("Sima Simic", 18);
        lista2.Add(a);
        lista2.Add(b);

        int[] niz = new int[] { 1, 2, 3, 4 };
        string s = "skola";

        lista2.Add(5);
        lista2.Add(niz);
        lista2.Add(s);
        Console.ReadKey();
    }
}
```

199. Датим кодом у програмском језику C# креира се пет објеката класе *Osoba* која имплементира интерфејс *IComparable*. Допунити код програма наредбом која, помоћу колекције података, формира генеричку листу особа - променљива *lista* и наредбу која врши сортирање те листе. Анализирати дати код и одредити који од понуђених одговора је потребно дописати у 7. и 10. линију кода како би се правилно декларисала и сортирала променљива *lista*.

```

1.  static void Main(string[] args) {
2.  Osoba a = new Osoba("Marko Ilic", 34);
3.  Osoba b = new Osoba("Mirko Prljic", 30);
4.  Osoba c = new Osoba("Danilo Sekara", 24);
5.  Osoba d = new Osoba("Sara Males", 15);
6.  Osoba e = new Osoba("Borko Ilic", 34);
7.  _____;
8.  lista.Add(a); lista.Add(b); lista.Add(c);
9.  lista.Add(d); lista.Add(e);
10. _____;
11. Console.WriteLine("Prikaz osoba po godinama starosti:");
12. foreach (Osoba x in lista) { Console.WriteLine(x); }
13. }
```

2

Заокружити број испред тачног одговора:

1. `List lista = newList(); lista.Sort(null);`
2. `List<Osoba> lista; lista.Sort();`
3. `ArrayList<Osoba> lista = new ArrayList<Osoba>(); lista.Sort();`
4. `List<Osoba> lista = newList<Osoba>(); lista.Sort();`

Понуђени одговори под 2. и 3. имају синтаксну грешку. Дилема је одговор под 1 или 4. Имајући у виду да је потребно формирати **генеричку листу особа**, дакле сви елементи морају да буду истог типа, тачан одговор је под 4.

СОРТИРАЊЕ ДИНАМИЧКИХ НИЗОВА

Сортирање листе простих типова, илустровано је на следећем примеру.

```

using System;
using System.Collections.Generic;

class Program
{
    static void Main(string[] args)
    {
        List<int> brojevi = new List<int>() { 5, 8, 7, 4, 2, 0, 1 };

        Console.WriteLine("Brojevi pre sortiranja 1. varijanta");
        brojevi.ForEach(Console.WriteLine);

        Console.WriteLine("Brojevi pre sortiranja 2. varijanta");
        foreach (int x in brojevi)
            Console.WriteLine(x);

        brojevi.Sort();
        Console.WriteLine("Brojevi posle sortiranja");
        brojevi.ForEach(Console.WriteLine);
    }
}
```

Ако покушамо да применимо исти начин за сортирање листе са сложеним типовима. Појавиће се грешка.

У следећем примеру имамо сложени тип са подацима које је потребно сортирати по годинама, компајлер пријављује грешку.

```
using System;
using System.Collections.Generic;

class Osoba
{
    public string ime;
    public int godina;
    public Osoba(string ime, int godina)
    {
        this.ime = ime;
        this.godina = godina;
    }
}

class Program
{
    static void Main(string[] args)
    {
        Osoba a = new Osoba("Marko Ilic", 34);
        Osoba b = new Osoba("Mirko Prljic", 30);
        Osoba c = new Osoba("Danilo Sekara", 24);
        Osoba d = new Osoba("Sala Males", 15);
        Osoba e = new Osoba("Borko Ilic", 34);

        List<Osoba> lista = new List<Osoba>();

        lista.Add(a); lista.Add(b); lista.Add(c); lista.Add(d); lista.Add(e);

        lista.Sort();
        Console.WriteLine("Prikaz osoba po godinama starosti: ");
        foreach (Osoba x in lista) Console.WriteLine(x.godina);
    }
}
```

Проблем се решава преко интерфејса **IComparable<T>**:

```
class Osoba: IComparable<Osoba>
{
    public string ime;
    public int godina;
    public Osoba(string ime, int godina)
    {
        this.ime = ime;
        this.godina = godina;
    }

    public int CompareTo(Osoba other)
    {
        return this.godina.CompareTo(other.godina);
    }
}
```

АНАЛИЗИРАТИ СЛЕДЕЋИ ПРИМЕР И ДАТИ ТАЧАН ОДГОВОР:

198. Дат је код програма у програмском језику C# који дефинише класу **Osoba** са њеним методама и атрибутима. Анализирати дати код и одредити који од понуђених одговора је потребно дописати у 1. ред кода како би метода била тачно дефинисана.

```

1.  public class Osoba _____ {
2.      private string ime;
3.      private string prezime;
4.      int godina;
5.      public Osoba(string ime, string prezime, int godina) {
6.          this.ime = ime;
7.          this.prezime = prezime;
8.          this.godina = godina;
9.      }
10.     public int GetGodina() { return this.godina; }
11.     public int CompareTo(Osoba obj) {
12.         if (this.godina > obj.GetGodina()) return 1;
13.         else if (this.godina < obj.GetGodina()) return -1;
14.         else return 0;
15.     }
16.     public override string ToString() {
17.         return "Ime: " + this.ime + "\tPrezime: " +
18.         this.prezime + "\tGodina: " + this.godina;
19.     }

```

1

Заокружити број испред тачног одговора:

1. : IComparable<Osoba>
2. : IComparable
3. : IEquatable<Osoba>
4. : Comparer

200. Дат је код програма у програмском језику C# који дефинише класу Osoba са њеним методама и атрибутима. Анализирати дати код и на основу декларације метода CompareTo(...) и Clone() одредити код које недостаје у првој линији.

```

1.  class Osoba : _____{
2.      private string ime;
3.      private double dohodak;
4.      public Osoba(string ime, double dohodak) {
5.          this.ime = ime;
6.          this.dohodak = dohodak;
7.      }
8.      public int CompareTo(Osoba osb) {
9.          if (this.dohodak < osb.dohodak) return -1;
10.         else if (this.dohodak > osb.dohodak) return 1;
11.         else return 0;
12.     }
13.     public Object Clone() {
14.         return this.MemberwiseClone();
15.     }
16.     public override string ToString(){
17.         return "Ime: "+this.ime+"\nDohodak: "+this.dohodak;
18.     }
19. }

```

2

Заокружити број испред тачног одговора:

1. IComparable<Osoba>, ICloneable
2. Comparable<Osoba>, Cloneable
3. IComparable<Osoba>, ICloneable<Osoba>
4. IComparable, ICloneable

За клонирање објекта користи се кључна реч **ICloneable**.

Задатак: Анализирати програмски код и проценити да ли ће се појавити грешка.

```
using System;

class Program
{
    public static void m(Object x) {
        Console.WriteLine(x.ToString());
    }

    static void Main()
    {
        Object o = new Object();
        m(o);

        Program.m(new Student());
        Program.m(new Osoba());
        Program.m(new Object());
    }
}

class Osoba : Object {
    public string toString() { return "Osoba"; }
}

class Student : Osoba {
    public string toString() { return "Student"; }
}

class MasterSudent : Student { }
```

Модификовати претходни задатак, променити тип параметра у методи **m**, тј. уместо **Object** поставити **Student**. Проценити тачност задатка и објаснити које методе ће пријавити грешку и зашто.

АНАЛИЗИРАТИ СЛЕДЕЋИ ПРИМЕР И ДАТИ ТАЧНЕ ОДГОВОРЕ:

226. Дат је код програма у програмском језику C# којим су дефинисане четири класе: `class Program` која садржи `Main(String[] args)` методу, `class MasterStudent`, `class Student` и `class Osoba`. Приликом превођења овог кода компајлер јавља грешку. Заокружити бројеве испред понуђених исказа који описују разлоге настајања грешке приликом компајлирања.

```
class Program {
    static void Main(string[] args){
        m(new MasterStudent());
        m(new Student());
        m(new Osoba());
        m(new Object());
    }
    public static void m(Student x){
        Console.WriteLine(x.ToString());
    }
}

class MasterStudent : Student{ }
class Student: Osoba{
    public string ToString(){ return "Student"; }
}

class Osoba: Object{
    public string ToString(){ return "Osoba"; }
}
```

1. Грешка је у позиву `m(new MasterStudent())`.
2. Грешка је у позиву `m(new Student())`.
3. Грешка је у позиву `m(new Osoba())`.
4. Грешка је у позиву `m(new Object())`.