1. uzdevums

Izmantojot **ArrayList** izveidot programmu, kas nodrošina sekojošās darbības:

1. Pievienot sarakstam jaunu elementu (pievienoti tiek int tipa mainīgie);
2. Dzēst no saraksta kādu elementu (pārbaudīt vai tāds elements vispār eksistē);
3. Sakārtot sarakstu;
4. Dzēst no saraksta elementu pēc tā indeksa;
5. Apgriezt sarakstu otrādi;
6. Izdzēst konkrētu saraksta apgabalu;
7. Pievienot sarakstam jaunu elementu konkrētā vietā;
8. Pievienot sarakstam elementu kopu konkrētā vietā;
9. Pievienot sarakstam elementu kopu saraksta beigās;
10. Iztīrīt visu sarakstu;
11. Attēlot saraksta vērtības.

Pie visām darbībām veikt pārbaudes, lai programma lietotāju brīdinātu, ka šo darbību nevar veikt.

2. uzdevums

1. Izveidot klasi **Students** ar šādām īpašībām: vārds un uzvārds.
2. Nodefinēt divus **ArrayList** studentu sarakstu – Melno un Balto. Programmai ir jābūt sekojošiem darbības principiem:
    a. Melnā un baltā saraksta aizpildīšana ar klases vērtībām;
    b. Studentu pārvietošana no melnā uz balto sarakstu un otrādi.
3. Nodrošināt iespēju sarakstu saglabāt uz failu un nolasīt no faila!

3. uzdevums

Izveidot **Queue** sarakstu. Realizēt iespēju:

1. Pievienot sarakstam jaunus elementus.
2. Izņemt no saraksta elementus (elementu skaitu nosaka lietotājs).
3. Lietotājs ievada elementu, kuru grib atrast sarakstā. Ja elements tiek atrasts, programma izvada elementa numuru. Ja elements netika atrasts, paziņojumu, ka šāds elements nav.
4. Atgriež informāciju par sarakstu: elementu skaitu un izvada visus elementus.
5. Izdzēst visu sarakstu.

4. uzdevums

Izveidot **Stack** sarakstu. Realizēt iespēju:

1. Aizpildīt sarakstu, izvadīt uz ekrāna.
2. Izveidot masīvu no saraksta elementiem.
3. Izvadīt masīvu uz ekrāna.
4. Dzēst sarakstu.

Klase ArrayList

Klase `ArrayList` ir dinamisks masīvs, kas var saturēt nenoteiktu daudzumu dažādu tipu objektus.

Šīs klases darbības principi ir sekojoši:

1. Izveidojot jaunu `ArrayList` objektu automātiski tiek paredzēts, ka tas varēs glabāt desmit dažādu tipu objektus.
2. Ja programmas darbības laikā objekta elementu skaits pārsniedz rezervētos četru, klases `ArrayList` kapacitāte automātiski tiek palielināta vēl par četriem elementiem.

Klases `ArrayList` svarīgākās īpašības un metodes

Īpašības

Nosaukums	Tips	Apraksts	
`Capacity`	`Int`	Īpašība, kas norāda cik elementus var glabāt sarakstā	Rakstīt, Lasīt
`Count`	`Int`	Īpašība, kas norāda cik elementus satur saraksts	Lasīt
`IsFixedSisze`	`Bool`	Īpašība, kas norāda vai sarakstam ir ierobežots elementu skaits	Lasīt
`IsReadOnly`	`Bool`	Īpašība, kas norāda vai sarakstā ir iespējams ierakstīt	Lasīt

Metodes

Nosaukums	Apraksts
`Add`	Metode elementa pievienošanai sarakstam
`AddRange`	Metode vairāku elementa pievienošanai sarakstam saraksta beigās
`Clear`	Metode, kas izdzēš visus elementus no saraksta
`Contains`	Metode, kas pārbauda vai saņemtais elements atrodas sarakstā
`IndexOf`	Metode, kas atrod un atgriež saņemtā elementa indeksu sarakstā (pirmo sastapto)
`Insert`	Metode elementa pievienošanai sarakstam konkrētā vietā
`InsertRange`	Metode vairāku elementa pievienošanai sarakstam konkrētā vietā
`LastIndexOf`	Metode, kas atrod un atgriež saņemtā elementa indeksu sarakstā (pēdējo sastapto)
`Remove`	Metode, kas atrod un izdzēš saņemto elementu no saraksta (pirmo sastapto)
`RemoveAt`	Metode, kas izdzēš elementu ar saņemto indeksu
`RemoveRange`	Metode, kas izdzēš veselu apgabalu
`Reverse`	Metode, kas pārveido saraksta elementus otrādākā secībā
`Sort`	Metode, kas sakārto visus vai daļu saraksta elementus

Vienkārša programma darbam ar saraksta elementiem

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

class Program
    {
        static void Main(string[] args)
        {
            //izveido jaunu ArrayList tipa objektu
            ArrayList aL = new ArrayList();
            //ieliek int tipa mainīgo bez nosaukuma
            aL.Add(10);
            //izveido int tipa mainīgo c un divreiz ieliek saraksta
            int c = 5;
            aL.Add(c);
            aL.Add(c);

            //elementu izvadīšana
            for (int i = 0; i < aL.Count; i++)
```

```
            {
                Console.WriteLine("Elements[{0}]={1}", i, aL[i]);
            }
            Console.WriteLine();
            //dzēš saraksta 0 elementu
            aL.RemoveAt(0);

            //vēlreiz izvada visu sarasktu
            for (int i = 0; i < aL.Count; i++)
            {
                Console.WriteLine("Elements[{0}]={1}", i, aL[i]);
            }
            Console.WriteLine();
            //dzēš pirmo elementu kas vienāds ar  c
            aL.Remove(c);

            //vēlreiz izvada visu sarasktu
            for (int i = 0; i < aL.Count; i++)
            {
                Console.WriteLine("Elements[{0}]={1}", i, aL[i]);
            }
        }
    }
```

Programmas skaidrojums

Šī programma demonstrē pamat darbības ar klasi `ArrayList`. Sākumā tiek izveidots jauns `ArrayList` objekts `aL`, kuram netiek norādīts izmērs, pēc tam ar metodes `Add`() palīdzību tiek ievietots skaitlis desmit (tips netiek norādīts, to programma nosaka automātiski) pēc tam tiek izveidots int tipa mainīgais `c` ar vērtību `5` un līdzīgi kā iepriekš ievietots sarakstā. Saraksta elementu izvadīšanai izveido `for` ciklu ar skaitītāju i, kas mainās no 0 līdz elementu skaitam sarakstā un izmantojot `Console.WriteLine` izvada elementu indeksus un vērtības uz ekrāna.

Programma darbam ar dažādu tipu saraksta elementiem

```
    class Program
    {
        static void Main(string[] args)
        {
            //izveido jaunu ArrayList tipa objektu
            ArrayList aL = new ArrayList();
            //ieliek int tipa mainīgo bez nosaukuma
            aL.Add(10);
            //ieliek doble tipa mainīgo bez nosaukuma
            aL.Add(10.50068);
            //izveido int tipa mainīgo c un ieliek saraksta
            int c = 5;
            aL.Add(c);
            //izveido float tipa mainīgo b un ieliek saraksta
            float b = 5.56F;
            aL.Add(b);
            //elementu izvadīšana
            for (int i = 0; i < aL.Count; i++)
            {
                Console.WriteLine("Elements[{0}]={1}", i, aL[i]);
            }
        }
    }
```

Programmas skaidrojums

Arī šī piemēra darbības principi ir līdzīgi, tikai sarakstā tiek likti dažādu tipu elementi un pēc tam izvadīti uz ekrāna. Par cik klases Console metode WriteLine ir polimorfa un pēc būtības var atēlot visus iebūvētos tipus pie izvadīšanas programmētājam nav jādomā par elementu tipiem, bet gadījumā

ja sarakstu ir jāpiešķir kādam mainīgajam mums jāzin ir saraksta elementa tips un jēviec atbilstošās pārveidošanas piemēram:

Lai kādam mainīgajam piešķirtu saraksta nulto elementu būs jāveic šāda tipu pārveidošana:

```
int z = (int)aL[0];
```

vai

```
int h = (double)aL[1];
```

Saraksta kapacitātes pārbaudes programma

```
class Program
{
    static void Main(string[] args)
    {
        //izveido jaunu ArrayList tipa objektu
        ArrayList aL = new ArrayList();
        //ieliek vienu elementu
        aL.Add(5);
        //izvada saraksta kapacitāti uz ekrāna
        Console.WriteLine(aL.Capacity);
        //ieliek sarakstā vēl četrus elementus
        for (int i=0; i < 4; i++)
        {
            aL.Add(i);
        }
        //izvada saraksta kapacitāti uz ekrāna
        Console.WriteLine(aL.Capacity);
    }
}
```