P175B502 Objektinio programavimo pagrindai II

Egzamino temos

1-as klausimas (4 tšk.)

Algoritmai (rikiavimas, dviejų tvarkingų rinkinių suliejimas);
Struktūros (*struct*), *DateTime*, *TimeSpan*;
Išvardijimo tipas (*enum*);
Dinaminis masyvas (*List*): savybės ir metodai;
Dėklo klasė:

- Stack Class (System.Collections);
- Stack(T) Class (System.Collections.Generic);

Eilės klasė:

- Queue Class (System.Collections);
- Queue(T) Class (System.Collections.Generic);

Žodyno klasė:

- Dictionary(Tkey, Tvalue) Class (System.Collections.Generic);
- SortedDictionary(Tkey, Tvalue) Class (System.Collections.Generic);

Surikiuoto sąrašo klasė:

- SortedList Class (System.Collections);
- SortedList Class(Tkey, TValue) Class (System.Collections.Generic);

Aibės klasė: HashSet(T) Class (System.Collections.Generic);

2-as klausimas (2 tšk.)

Paveldėjimas:

- Klasių hierarchija (bazinė ir išvestinė klasės);
- Bazinės, išvestinės klasės savybės ir konstruktoriai, jų paveldimumas;
- Konstruktoriai su numatytomis parametrų reikšmėmis;
- Klasės elementų matomumo požymiai (*private*, *public*, *protected*);
- Objektų sukūrimas ir priskyrimo veiksmai paveldėjime;

Polimorfizmas:

- Statinis polimorfizmas (operatorių ir metodų užklojimas);
- Dinaminis polimorfizmas (virtualūs metodai ir jų užklojimas);
- Abstrakčios klasės, abstraktūs metodai ir savybės;
- Abstrakčios klasės objektų masyvas;
- Sealed klasė;
- Object klasė;

3-as klausimas (4 tšk.)

Susietasis vienkryptis (dvikryptis) sąrašas:

- duomenų klasė;
- mazgo klasė;
- sąrašo klasė;
- formavimas (atvirkštine ir tiesiogine tvarka);
- spausdinimas (peržiūra);
- elementų šalinimas;
- elementų įterpimas;
- paieška;
- tvarkymas;
- rikiavimas „minmax“ ir „burbuliuko“ metodais;
- sąrašas su fiktyviais elementais;

Susietasis sąrašas:

- LinkedListNode(T);
- LinkedList (T);

Egzamino bilieto pavyzdys

INSTRUKCIJA ATSISKAITANČIAJAM:

Atliekant užduotis, *papildykite* tas programos vietas, kurios yra nurodytos žodžiais:
`// ATLIKITE: ...`
Reikiamus 3 projektus *įsikelkite* į savo katalogą pagal egzamino bilieto variantą.

1-as klausimas (4 tšk.)

Tekstiniame faile *Asmenys.txt* yra duomenys apie atvykusius asmenis:

Pavardė vardas, amžius, atvykimo laikas (valanda: minutės: sekundės).

Papildykite programą:

- Sudarykite eilę `Eile`, į kurią surašomi visi dinaminio masyvo (`List`) asmenys, kurių atvykimo laikas yra duotame intervale [`atvykimoPradzia, atvykimoPradzia`] ir yra kartotinis duotam žingsniui `žingsnis`.
- Atspausdinkite eilės elementų kiekį; pirmą eilės elementą.
- Sudarykite rikiuotą žodyną `Zodynas`, į kurį surašomi visi dinaminio masyvo (`List`) asmenys, kurių amžius didesnis už pirmo eilės `Eile` asmens amžių. Žodyno raktai – sveiki skaičiai, paeiliui imami iš duoto sveikų skaičių masyvo `Raktai`, o reikšmės – `Asmuo` klasės objektai.
- Atspausdinkite žodyną;
- Raskite ir atspausdinkite žodyne surašytų asmenų amžiaus aritmetinį vidurkį.

2-as klausimas (2 tšk.)

Įmonė prieš įdarbinant naujus darbuotojus sudaro kandidatų sąrašą. Priėmus kandidatą į darbą, mokamas *800* eurų bazinis atlyginimas ir atsižvelgiama į ankstesnius jo darbo rodiklius. Pirmiausia nuspręsta ieškoti programavimo specialistų. Pabaikite rašyti programą, kurį padėtų įmonei apskaičiuoti programuotojams mokamus atlyginimus, jei būtų nuspręsta juos priimti į darbą.

Duomenys:

- Abstrakti bazinė klasė `Kandidatas`, laukai: *Pavardė ir vardas, amžius (sveikas sk.), darbo stažas (realus sk., metais).*
- Išvestinė klasė `Programuotojas`, laukai: *komandinė darbo patirtis (realus sk., metais), nusiskundimų skaičius (sveikas sk.).*

Programuotojams mokamas atlyginimas (MA) skaičiuojamas pagal formule:

$MA = BD + 0.2BD \cdot 1.1p + 0.1BD \cdot (-n)$
BD – bazinis atlyginimo dydis, p – komandinė darbo patirtis, n – nusiskundimų skaičius.

3-as klausimas (4 tšk.)

Tekstiniame faile `"Knygos.txt"` yra duomenys apie knygas:

pavadinimas, išleidimo metai, puslapių skaičius. (Laukai skiriami ; (kabliataškiais))
Duotos klasės `Knyga`, `Mazgas`, sąrašo klasė `Sąrašas` ir klasė `Program`.
Skaitykite duomenis į klasės sąrašas konteinerį, *spausdinkite* konteinerio duomenis faile, *pašalinkite* iš sąrašo knygas, kurių žodžių skaičius knygos pavadinime yra didesnis už klaviatūra nurodytą skaičių.
Paruoškite duomenų failo pavyzdį.

Paruošti naudoti programų tekstai

1-o klausimo programos tekstas:

```
public class Asmuo
{
    public string pavVard { get; set; }      // pavardė, vardas
    public int amžius { get; set; }          // amžius
    public TimeSpan laikas { get; set; }     // atvykimo laikas

    public Asmuo(string pavVard, int amžius, TimeSpan laikas)
    {
        this.pavVard = pavVard;
        this.amžius = amžius;
        this.laikas = laikas;
    }

    public override string ToString()
    {
        string eilute;
        eilute = string.Format("{0, -17}  {1}        {2}", pavVard, amžius, laikas);
        return eilute;
    }

    // Užklotas metodas GetHashCode()
    public override int GetHashCode()
    {
        return base.GetHashCode();
    }
}

class Program
{
    const string CFd1 = "..\\..\\Asmenys.txt";             // asmenų duomenų failo vardas

    static void Main(string[] args)
    {
        // Asmenų sąrašo sudarymas ir spausdinimas
        List<Asmuo> AsmenuList = SkaitytiAsmuoList(CFd1);
        SpausdintiAsmenuList(AsmenuList, "Pradiniai duomenys");
        Queue<Asmuo> Eile = new Queue<Asmuo>();
        TimeSpan atvykimoPradzia = new TimeSpan( 7, 0, 0); // tikrinamo laiko intervalo pradžia
        TimeSpan atvykimoPabaiga = new TimeSpan(10, 0, 0); // tikrinamo laiko intervalo pabaiga
        TimeSpan žingsnis = new TimeSpan(0, 10, 0);        // laiko intervalo peržiūros žingsnis
        int [] Raktai = {7, 9, 4, 1, 6, 2, 3};             // žodyno raktai

        // ATLIKITE: visus nurodytus skaičiavimus
    }

    // spausdina asmenų duomenų lentelę
    static void SpausdintiAsmenuList( List<Asmuo> AsmuoList, string antraste)
    {
        const string virsus =
                    "-------------------------------------------------  \r\n"
                  + " Nr.  Pavardė, vardas   Amžius    Atvykimo laikas  \r\n"
                  + "-------------------------------------------------";
        Console.WriteLine("\n " + antraste);
        Console.WriteLine(virsus);
        for (int i = 0; i < AsmuoList.Count; i++)
        {
            Asmuo zmog = AsmuoList[i];
            Console.WriteLine("{0, 3}   {1}", i + 1, zmog);
        }
        Console.WriteLine("-------------------------------------------------\n");
    }

    // skaito asmenų duomenų failą
    static List<Asmuo> SkaitytiAsmuoList(string fv)
    {
        // asmenų dinaminis masyvas
```

```
            List<Asmuo> AsmuoList = new List<Asmuo>();
            using (StreamReader srautas = new StreamReader(fv, Encoding.GetEncoding(1257)))
            {
                string eilute;
                while ((eilute = srautas.ReadLine()) != null)
                {
                    string[] eilDalis = eilute.Split(';');
                    string pav = eilDalis[0];
                    int amžius = int.Parse(eilDalis[1]);
                    TimeSpan laikas = TimeSpan.Parse(eilDalis[2]);
                    Asmuo naujas = new Asmuo(pav, amžius, laikas);
                    AsmuoList.Add(naujas);
                }
            }
            return AsmuoList;
        }

        // spausdina žodyno reikšmes
        // naudoja ENumerator
        public static void Spausdinti(SortedDictionary<int, Asmuo> zodynas)
        {
            var enumerator = zodynas.GetEnumerator();
             while (enumerator.MoveNext())
            {
                 object item = enumerator.Current;
                 Console.WriteLine("  {0}  ", item);
            }
            Console.WriteLine();
        }

        // Formuoja eilę
        static void Atrinkti(List<Asmuo> AsmenuList, Queue<Asmuo> Eile,
                             TimeSpan atvykimoPradzia, TimeSpan atvykimoPabaiga,
                             TimeSpan žingsnis)
        {
            // ATLIKITE: Dinaminio masyvo asmenys, kurių atvykimo laikas yra duotame
            // intervale [atvykimoPradzia, atvykimoPradzia] ir yra kartotinis duotam
            // žingsniui žingsnis, įrašomi į eilės konteinerį.
        }

        // Formuoja žodyną
        static void Formuoti(List<Asmuo> AsmenuList, SortedDictionary<int, Asmuo> Zodynas,
                                int metai, int [] Raktai)
        {
            // ATLIKITE: Dinaminio masyvo asmenys, kurių amžius didesnis už duotą sveiką skaičių
            // metai, surašomi į rikiuotą žodyną. Žodyno raktai – sveiki skaičiai, paeiliui imami
            // iš duoto sveikų skaičių masyvo Raktai, o reikšmės – Asmuo klasės objektai.
        }
    }
```

2-o klausimo programos tekstas:

```
abstract class Kandidatas : Object
{
   protected const double BazinisDydis = 800.00;       // Bazinis atlyginimo dydis
   protected string PavVrd { get; set; }               // Pavardė ir vardas
   protected int Amzius    { get; set; }               // Amžius
   protected double Stažas { get; set; }               // Darbo stažas (metais)
   // Klasės konstruktorius
   public Kandidatas(string PavVrd = "", int Amzius = 0, double Stažas = 0.0)
   {
       this.PavVrd = PavVrd;
       this.Amzius = Amzius;
       this.Stažas = Stažas;
   }
   // Abstraktus metodas
   public abstract double Atlyginimas();

   public override string ToString()
   {
     // ATLIKITE: Užklokite kandidato spausdinimo eilute metodą
   }
}

class Programuotojas : Kandidatas
{
    // ATLIKITE: Aprašykite klasės savybes ir konstruktorių
    // ATLIKITE: Užklokite programuotojo atlyginimo skaičiavimo metodą
    // ATLIKITE: Užklokite programuotojo spausdinimo eilute metodą
}

class Program
{
    static void Main(string[] args)
    {
        // Programuotojų objektų masyvas P(n)
        int n = 3;  Programuotojas[] P = new Programuotojas[n];
        // P(n) objektų užpildymas reikšmėmis
        P[0] = new Programuotojas("Programuotojas1", 29,  1.1, 1.5, 0);
        P[1] = new Programuotojas("Programuotojas2", 39, 11.5, 2.2, 3);
        P[2] = new Programuotojas("Programuotojas3", 30,  3.0, 3.6, 0);
        // ATLIKITE: Papildykite Main metodą reikiamais veiksmais
    }

    public static void Spausdinti(Kandidatas[] K, int kn)
    {
       // ATLIKITE: Spausdinkite kiekvieno kandidato atlyginimą ekrane
    }
}
```

3-o klausimo programos tekstas:

```
public class Knyga
{
    public string Pavadinimas { get; set; }
    public int Metai { get; set; }
    public int PuslapiuSk { get; set; }
    public Knyga(string pavad = "", int metai = 0, int puslSk = 0)
    {
        this.Pavadinimas = pavad;
        this.Metai = metai;
        this.PuslapiuSk = puslSk;
    }
    public override string ToString()
    {
        string eilute;
        eilute = string.Format("{0, -20}  {1, 4:d}    {2, 4:d}",
                               Pavadinimas, Metai, PuslapiuSk);
        return eilute;
    }
}
// Klasė sąrašo vienam elementui saugoti
public sealed class Mazgas
{
    public Knyga Duom { get; set; }     // duomenys
    public Mazgas Kitas { get; set; }   // nuoroda į kitą elementą
    public Mazgas(Knyga duom, Mazgas adresas)
    {
        this.Duom = duom;
        this.Kitas = adresas;
    }
}
// Klasė knygų duomenims saugoti
public sealed class Sąrašas
{
    private Mazgas pr;  // sąrašo pradžia
    private Mazgas pb;  // sąrašo pabaiga
    private Mazgas ss;  // sąrašo sąsaja
    // Pradinės sąrašo nuorodų reikšmės
    public Sąrašas()
    {
        this.pr = null;
        this.pb = null;
        this.ss = null;
    }
    // Grąžina sąrašo sąsajos elemento reikšmę (duomenis)
    public Knyga ImtiDuomenis() { return ss.Duom; }
    // Sukuriamas sąrašo elementas ir prijungiamas prie sąrašo pabaigos
    // nauja – naujo elemento reikšmė (duomenys)
    public void DėtiDuomenisT(Knyga nauja)
    {
        // ATLIKITE: padėkite naują elementą sąrašo pabaigoje
    }
    // Sąsajai priskiriama sąrašo pradžia
    public void Pradžia() { ss = pr; }
    // Sąsajai priskiriamas sąrašo sekantis elementas
    public void Kitas() { ss = ss.Kitas; }
    // Grąžina true, jeigu sąsaja netuščia
    public bool Yra() { return ss != null; }
    // Šalina sąsajos rodomą elementą
    public void Salinti()
    {
        // ATLIKITE: pašalinkite sąsajos rodomą elementą
   }
}
class Program
{
    static void Main(string[] args)
    {
        const string CFd = @"..\..\Knygos.txt";
```

```
            const string CFr = @"..\..\Rezultatai.txt";

            if (File.Exists(CFr))
                File.Delete(CFr);

            // ATLIKITE: skaitykite duomenis iš failo į tiesioginį sąrašą Knygos,
            //           spausdinkite duomenis, pašalinkite knygas,
            //           kurių pavadinime yra daugiau nei nurodytas knygų skaičius,
            //           spausdinkite sąrašą.

            Console.WriteLine("Programa darbą baigė.");
        }

        // Skaitomos objektų reikšmės iš failo ir sudedamos į sąrašą tiesiogine tvarka
        // fv – duomenų failo vardas
        // Grąžina - sąrašo objekto nuorodą
        static Sąrašas ĮvestiTiesiog(string fv)
        {
            Sąrašas A = new Sąrašas();
            using (var failas = new StreamReader(fv, Encoding.GetEncoding(1257)))
            {
                string pavad;
                int metai;
                int pslSk;
                string eilute;
                while ((eilute = failas.ReadLine()) != null)
                {
                    var eilDalis = eilute.Split(';');
                    pavad = eilDalis[0];
                    metai = Convert.ToInt32(eilDalis[1]);
                    pslSk = Convert.ToInt32(eilDalis[2]);
                    var Knyga = new Knyga(pavad, metai, pslSk);
                    A.DėtiDuomenisT(Knyga);
                }
            }
            return A;
        }
        // Sąrašo A duomenys spausdinami lentele faile fv
        // fv – duomenų failo vardas
        // A  - sąrašo objekto adresas
        // antraste  - lentelės pavadinimas
        static void Spausdinti(string fv, Sąrašas A, string antraste)
        {
            using (var failas = new StreamWriter(fv, true))
            {
                failas.WriteLine(antraste);
                failas.WriteLine("-------------------------------------");
                failas.WriteLine("Pavadinimas          Metai Puslapiai ");
                failas.WriteLine("-------------------------------------");
                // Sąrašo peržiūra, panaudojant sąsajos metodus
                for (A.Pradžia(); A.Yra(); A.Kitas())
                {
                    failas.WriteLine(A.ImtiDuomenis());
                }
                failas.WriteLine("-------------------------------------\n");
            }
        }
        // Iš sąrašo išmeta knygas, kurių pavadinime yra didesnis nei nurodytas žodžių skaičius
        // A  - sąrašo objekto adresas
        // zodSkaicius - žodžių skaičius knygos pavadinime
        static void Išmesti(Sąrašas A, int zodSkaicius)
        {
            // ATLIKITE: pašalinkite iš sąrašo knygas, kurių pavadinime yra didesnis nei
            //                                        nurodytas žodžių skaičius
        }
    }
```