**Поени:**

# Предиспитни пројекат из Објектно оријентисаног програмирања

Студент:Дамир Трелак, ИТ 13/18

Датум: 30.04.2020.

# Питања и одговори:

Подестник:

Тема програма је произвољна.

**За 10 поена:** Конзолна апликација са класама као засебним фајловима. Класе садрже својства, методе статичке-нестатичке, композицију класа или нестинг, наслеђивање .

**За 15 поена*:*** *Windows forms* апликација са класама као засебним фајловима. Класе садрже својства, методе статичке-нестатичке, композицију класа или нестинг, наслеђивање.

Или

Конзолна апликација са класама као засебним фајловима. Класе садрже својства, методе статичке-нестатичке, композицију класа или нестинг, наслеђивање, апстрактне класе или интерфејси, полиморфизам

**За 20 поена:** *Windows forms* апликација са класама као засебним фајловима. Класе садрже својства, методе статичке-нестатичке, композицију класа или нестинг, наслеђивање, апстрактне класе или интерфејси, полиморфизам

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System;

using System.IO;

namespace Projekat\_OOP\_1

{

public class Pacijent

{

private string ime;

private string prezime;

private int godine;

public Pacijent(string ime, string prezime, int godine) : base()

{

this.ime = ime;

this.prezime = prezime;

this.godine = godine;

}

public virtual string Ime

{

get

{

return ime;

}

}

public virtual string Prezime

{

get

{

return prezime;

}

}

public virtual int Godine

{

get

{

return godine;

}

}

public override string ToString()

{

return ime + " " + prezime + " (" + godine + ")";

}

public abstract class Lekar

{

private string ime;

private string prezime;

private Pacijent[] pacijenti;

private int brojPacijenata;

public Lekar(string ime, string prezime, int maksimalanBrojPacijenata)

{

this.ime = ime;

this.prezime = prezime;

pacijenti = new Pacijent[maksimalanBrojPacijenata];

brojPacijenata = 0;

}

public virtual string Ime

{

get

{

return ime;

}

}

public virtual string Prezime

{

get

{

return prezime;

}

}

public virtual Pacijent[] Pacijenti

{

get

{

return pacijenti;

}

}

public virtual int BrojPacijenata

{

get

{

return brojPacijenata;

}

}

public virtual bool prihvati(Pacijent pacijent)

{

if (brojPacijenata + 1 < pacijenti.Length)

{

pacijenti[brojPacijenata++] = pacijent;

return true;

}

else

{

return false;

}

}

public abstract int plata();

public override string ToString()

{

return this.GetType().Name + " " + ime + " " + prezime;

}

}

public class Pedijatar : Lekar

{

public Pedijatar(string ime, string prezime) : base(ime, prezime, 200)

{

}

public override int plata()

{

int dodatakZaMaluDecu = 0;

for (int i = 0; i < BrojPacijenata; i++)

{

if (Pacijenti[i].Godine < 5)

{

dodatakZaMaluDecu += 100;

}

}

return BrojPacijenata \* 300 + dodatakZaMaluDecu;

}

public override bool prihvati(Pacijent pacijent)

{

if (pacijent.Godine > 18)

{

return false;

}

return base.prihvati(pacijent);

}

}

public class Pulmolog : Lekar

{

public Pulmolog(string ime, string prezime) : base(ime, prezime, 100)

{

}

public override int plata()

{

return BrojPacijenata \* 500;

}

}

public class Poliklinika

{

private Lekar[] lekari;

public Poliklinika()

{

ucitajLekare();

ucitajPacijente();

}

private void ucitajLekare()

{

StreamReader file = new StreamReader("Lekari.txt");

lekari = new Lekar[int.Parse(file.ReadLine().Trim())];

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

{

string ime = file.ReadLine().Trim();

string prezime = file.ReadLine().Trim();

string vrstaLekara = file.ReadLine().Trim();

switch (vrstaLekara)

{

case "pedijatar":

lekari[i] = new Pedijatar(ime, prezime);

break;

case "pulmolog":

lekari[i] = new Pulmolog(ime, prezime);

break;

}

}

file.Close();

}

private void ucitajPacijente()

{

StreamReader file = new StreamReader("Pacijenti.txt");

int brojPacijenata = int.Parse(file.ReadLine().Trim());

Random randomGenerator = new Random();

for (int i = 0; i < brojPacijenata; i++)

{

Pacijent pacijent = new Pacijent(file.ReadLine().Trim(), file.ReadLine().Trim(), int.Parse(file.ReadLine().Trim()));

Lekar lekar = lekari[randomGenerator.Next(lekari.Length)];

lekar.prihvati(pacijent);

}

file.Close();

}

public virtual Pulmolog saNajmanjomPlatom()

{

Pulmolog pulmolog = null;

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

{

if (lekari[i] is Pulmolog && (pulmolog == null || lekari[i].plata() < pulmolog.plata()))

{

pulmolog = (Pulmolog)lekari[i];

}

}

return pulmolog;

}

public virtual Pedijatar saNajstarijimPacijentima()

{

Pedijatar pedijatar = null;

double najveciProsek = 0;

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

{

if (lekari[i] is Pedijatar && (pedijatar == null || prosekGodinaPacijenata(lekari[i]) > najveciProsek))

{

pedijatar = (Pedijatar)lekari[i];

najveciProsek = prosekGodinaPacijenata(lekari[i]);

}

}

return pedijatar;

}

private double prosekGodinaPacijenata(Lekar lekar)

{

double suma = 0;

for (int i = 0; i < lekar.BrojPacijenata; i++)

{

suma += lekar.Pacijenti[i].Godine;

}

if (suma == 0)

{

return 0;

}

return lekar.BrojPacijenata;

}

}

public class Glavna

{

static void Main(string[] args)

{

Poliklinika poliklinika = new Poliklinika();

Pulmolog saNajmanjomPlatom = poliklinika.saNajmanjomPlatom();

Console.WriteLine("Pulmolog sa najmanjom platom " + (saNajmanjomPlatom == null ? "ne postoji" : "je " + saNajmanjomPlatom));

Pedijatar saNajstarijimPacijentima = poliklinika.saNajstarijimPacijentima();

Console.WriteLine("Pedijatar sa najstarijim pacijentima " + (saNajstarijimPacijentima == null ? "ne postoji" : "je " + saNajstarijimPacijentima));

}

}

}

}

