Napredno i objektno programiranje

Zoološki vrt – Implementiranje sustava za upravljanje zoološkim vrtom

Petar Banjak 3.G

Elektrotehnička i ekonomska škola

Nova Gradiška

Sadržaj

[**Uvod** 1](#_Toc162381833)

[**Klase** 2](#_Toc162381834)

[Klasa „Nastamba“ 2](#_Toc162381835)

[Klasa „Zivotinje“ 3](#_Toc162381836)

[Klasa „Zaposlenici“ 4](#_Toc162381837)

[Klasa „PodatkovniKontekst“ 6](#_Toc162381838)

[**Forme** 9](#_Toc162381839)

[Forma „Nastamba“ 9](#_Toc162381840)

[Forma „Zivotinje“ 11](#_Toc162381841)

[Forma „Zaposlenici“ 13](#_Toc162381842)

[Forma „Zaposlenici i zivotinje“ 17](#_Toc162381843)

[Forma „Edit“ 18](#_Toc162381844)

# 

# **Uvod**

Potrebno je pomoću programa Visual Studio i programskog jezika C# implementirati sustav za upravljanje zoološkim vrtom s nastambama, životinjama i zaposlenicima zaduženim za pojedinu nastambu. Klase, IComparable, ToString(). Zatim je potrebno to povezati preko formi te napraviti podatkovni kontekst za jednu od klasa i također prikaz podataka u listBox-u za istu tu klasu; spremanje svih podataka u predodređene ListBoxove učitavanje, spremanje, brisanje podataka, sortiranje i filtriranje po jednom kriteriju.

# **Klase**

## Klasa „Nastamba“

Slika na kojoj se prikazuje tekst, snimka zaslona, softver, Multimedijski softver

Opis je automatski generiran

***Deklaracija klase „Nastamba“***

**Internal** – klasa je dostupna samo unutar istog assembly-a.

**Nastamba** – ime klase.

**:** **IComparable** – pokazuje da klasa implementira sučelje IComparable. To znači da se instance ove klase mogu uspoređivati međusobno.

**Privatna polja** – pohranjuju informacije o klasi Nastamba. Predstavljaju ime (ime), životinje (zivotnje), broj životinja (broj\_zivotinja) i veličinu (velicina).

**Svojstva** – omogućuju čitanje (get) i postavljanje (set) vrijednosti privatnih polja. Na primjer, „Ime“ omogućuje pristup i postavljanje vrijednosti polja „ime“.

**public int CompareTo(object obj) –** metoda dolazi iz sučelja IComparable koje je implementirano u klasi Nastamba. Ona omogućuje usporedbu dvaju objekata kako bi se odredio njihov redoslijed. Metoda uspoređuje trenutnu instancu (this) s objektom koji se predaje kao argument (obj).

Uspoređuje se „Zivotinje“ (tip životinja) trenutne instance s istim svojstvom objekta koji je proslijeđen.

Ako je rezultat te usporedbe jednak nuli, tada se uspoređuje „Ime“ (ime nastambe) kako bi se dodatno razlikovali objekti.

Na kraju, rezultat usporedbe se vraća.

**public override string ToString() –** nadjačava (override) metodu ToString. Metoda vraća string koji predstavlja trenutnu instancu. U ovom slučaju, vraća se niz koji sadrži ime i vrstu životinje odvojene zarezom.

## Klasa „Zivotinje“

Slika na kojoj se prikazuje tekst, snimka zaslona, softver, Multimedijski softver

Opis je automatski generiran

***Klasa „Zivotinje“***

**Internal** – klasa je dostupna samo unutar istog assembly-a.

**Zivotinje** – ime klase.

**:** **IComparable** – pokazuje da klasa implementira sučelje IComparable. To znači da se instance ove klase mogu uspoređivati međusobno.

**Privatna polja** – pohranjuju informacije o klasi Zivotinje. Predstavljaju zivotinjske\_vrste (životinjske vrste), vrstu životinja (vrsta), pasminu životinje (pasmina) i prehranu životinje (prehrana).

**Svojstva** – omogućuju čitanje (get) i postavljanje (set) vrijednosti privatnih polja. Na primjer, „Vrsta“ omogućuje pristup i postavljanje vrijednosti polja „vrsta“.

**public int CompareTo(object obj) –** metoda dolazi iz sučelja IComparable koje je implementirano u klasi Nastamba. Ona omogućuje usporedbu dvaju objekata kako bi se odredio njihov redoslijed. Metoda uspoređuje trenutnu instancu (this) s objektom koji se predaje kao argument (obj).

Uspoređuje se „Vrsta“ (vrsta životinje) trenutne instance s istim svojstvom objekta koji je proslijeđen.

Ako je rezultat te usporedbe jednak nuli, tada se uspoređuje „Pasmina“ (pasmina životinje) kako bi se dodatno razlikovali objekti.

Na kraju, rezultat usporedbe se vraća.

**public override string ToString() –** nadjačava (override) metodu ToString. Metoda vraća string koji predstavlja trenutnu instancu. U ovom slučaju, vraća se niz koji sadrži vrstu životinje i pasminu odvojene zarezom.

## Klasa „Zaposlenici“

Slika na kojoj se prikazuje tekst, snimka zaslona, softver, Multimedijski softver

Opis je automatski generiran

***Klasa „Zaposlenici“***

**Internal** – klasa je dostupna samo unutar istog assembly-a.

**Zaposlenici** – ime klase.

**:** **IComparable** – pokazuje da klasa implementira sučelje IComparable. To znači da se instance ove klase mogu uspoređivati međusobno.

**Privatna polja** – pohranjuju informacije o klasi Zaposlenici. Predstavljaju ime (ime zaposlenika), prezime (prezime zaposlenika), godine (godine zaposlenika) i radni\_polozaj zaposlenika (radni položaj zaposlenika).

**Svojstva** – omogućuju čitanje (get) i postavljanje (set) vrijednosti privatnih polja. Na primjer, „Ime“ omogućuje pristup i postavljanje vrijednosti polja „ime“.

**public int CompareTo(object obj) –** metoda dolazi iz sučelja IComparable koje je implementirano u klasi Zaposlenici. Ona omogućuje usporedbu dvaju objekata kako bi se odredio njihov redoslijed. Metoda uspoređuje trenutnu instancu (this) s objektom koji se predaje kao argument (obj).

Uspoređuje se „Prezime“ (prezime zaposlenika) trenutne instance s istim svojstvom objekta koji je proslijeđen.

Ako je rezultat te usporedbe jednak nuli, tada se uspoređuje „Ime“ (ime zaposlenika) kako bi se dodatno razlikovali objekti.

Na kraju, rezultat usporedbe se vraća.

**public override string ToString() –** nadjačava (override) metodu ToString. Metoda vraća string koji predstavlja trenutnu instancu. U ovom slučaju, vraća se niz koji sadrži prezime i ime odvojeno zarezom

\***NAPOMENA –** u klasi Nastamba smo dodali varijablu „zaposlenik“ tipa Zaposlenici i varijablu „zivotinja“ tipa Zivotinje kako bismo mogli povezati s ostalim klasama. Također u uspoređivanju kod metode CompareTo() smo napravili usporedbu pomoću serijskog broja nastambe, umjesto životinja i imena nastambe, tako što smo uključili varijablu „serijski\_broj“ tipa int

**\*NAPOMENA –** u klasi Zivotinje smo kod metode CompareTo() umjesto usporedbe preko životinje i vrste životinje smo napravili usporedbu preko serijskog broja životinje tako što smo dodali varijablu „serijski\_broj“ tip int

## Klasa „PodatkovniKontekst“

Slika na kojoj se prikazuje tekst, snimka zaslona, softver, Multimedijski softver

Opis je automatski generiran

**private List<Zaposlenici> \_zaposlenici -** Ova linija deklarira privatno polje \_zaposlenici tipa List<Zaposlenici>. Ova lista će sadržavati objekte klase Zaposlenici.

**public List<Zaposlenici> zaposlenici { get { return this.\_zaposlenici; } } -** Ova linija definira javno svojstvo zaposlenici koje omogućuje vanjskom kodu pristup privatnoj listi \_zaposlenici. Koristi se jednostavan getter.

**public PodatkovniKontekst() { this.\_zaposlenici = UcitajZaposlenike(); } -** Ova linija definira konstruktor PodatkovniKontekst(). Prilikom instanciranja objekta, konstruktor poziva metodu UcitajZaposlenike() i inicijalizira privatnu listu \_zaposlenici s rezultatom te metode.

**private string datZaposlenici = "zaposlenici.dat" -** Ova linija deklarira privatnu varijablu datZaposlenici tipa string i dodjeljuje joj vrijednost "zaposlenici.dat". Ova varijabla sadrži naziv datoteke koja se koristi prilikom učitavanja zaposlenika.

Slika na kojoj se prikazuje tekst, snimka zaslona, softver

Opis je automatski generiran

**private List<Zaposlenici> UcitajZaposlenike() {** - Ova linija definira privatnu metodu UcitajZaposlenike() koja vraća listu objekata tipa Zaposlenici.

**List<Zaposlenici> zaposlenici = new List<Zaposlenici>() -** Ovdje se stvara nova prazna lista zaposlenici koja će sadržavati objekte klase Zaposlenici.

**if (File.Exists(datZaposlenici)) -** Ova linija provjerava postojanje datoteke s imenom datZaposlenici prije nego što pokuša čitati iz nje. Ako datoteka postoji, ulazi u blok naredbi.

**using (StreamReader sr = new StreamReader(datZaposlenici)) -** Ova linija otvara datoteku za čitanje pomoću StreamReader objekta.

**while (!sr.EndOfStream) -** Petlja čita linije iz datoteke dok se ne dođe do kraja datoteke.

**string line = sr.ReadLine() -** Ova linija čita trenutnu liniju iz datoteke

**string[] polja = line.Split('|') -** Linija se razdvaja na polja koristeći znak '|' kao separator.

**Zaposlenici z = new Zaposlenici();**

**z.Ime = polja[0];**

**z.Prezime = polja[1];**

**z.Godine = int.Parse(polja[2]);**

**z.Radni\_Polozaj = polja[3] -** Stvara se objekt Zaposlenici i popunjavaju se podaci iz razdvojenih polja.

**zaposlenici.Add(z) -** Zaposlenik se dodaje u listu zaposlenici.

**return zaposlenici -** Vraća se lista zaposlenika koja je popunjena podacima iz datoteke.

Slika na kojoj se prikazuje tekst, Multimedijski softver, softver, grafički softver

Opis je automatski generiran

**public void SpremiZaposlenike() -** Ova linija označava početak javne metode SpremiZaposlenike()

**using (StreamWriter sw = new StreamWriter(datZaposlenici)) -** Otvara se StreamWriter objekt koji omogućuje pisanje u datoteku s imenom datZaposlenici. Ovaj blok „using“ osigurava automatsko zatvaranje resursa nakon izvršenja bloka.

**foreach (Zaposlenici z in this.zaposlenici) -** Ova linija označava početak foreach petlje koja prolazi kroz sve objekte tipa Zaposlenici u listi zaposlenici.

**sw.WriteLine($"{z.Ime}, {z.Prezime}, {z.Godine}, {z.Radni\_Polozaj}") -** Za svakog zaposlenika u listi, ova linija zapisuje redak u datoteku. Koristi se interpolacija niza ($"{...}") kako bi se formatirao redak s podacima o zaposleniku.

# **Forme**

## Forma „Nastamba“

Dizajn

Slika na kojoj se prikazuje tekst, snimka zaslona, softver, Ikona na računalu

Opis je automatski generiran

***Dizajn forme „Nastamba“***

Dizajn ove forme se sastoji od 5 button-a i jednog listBox-a

**„Zaposlenici“** – otvara formu „Zaposlenici“

**„Životinje“** – otvara formu „Životinje“

**„Životinje i zaposlenici“** – otvara formu „Zivotinje i zaposlenici“

**„Uredi“** – otvara formu „Zivotinje i zaposlenici“

**„Izlaz“** – izlaz iz formi

Programski kod

Slika na kojoj se prikazuje tekst, snimka zaslona, softver, operacijski sustav

Opis je automatski generiran

***Programski kod forme „Nastamba“***

**new Zaposlenici().Show(); -** naredba za otvaranje forme „Zaposlenici“

**new Zivotinje().Show(); -** naredba za otvaranje forme „Zivotinje“

**Environment.Exit(0); -** naredba za izlaz iz formi

**new Zaposlenici\_i\_zivotinje().Show(); -** naredba za otvaranje forme „Zaposlenici i zivotinje“

## Forma „Zivotinje“

Dizajn

Slika na kojoj se prikazuje tekst, softver, Ikona na računalu, računalo

Opis je automatski generiran

***Dizajn forme „Zivotinje“***

Dizajn ove forme se sastoji od listBox-a i 3 button-a

**„Dodaj“** – otvara formu „Edit“

**„Uredi“ -** otvara formu „Edit“

**„Obriši“ –** briše označene elemente u listBox-u

Programski kod

Slika na kojoj se prikazuje tekst, snimka zaslona, softver, Multimedijski softver

Opis je automatski generiran

***Programski kod forme „Zivotinje“***

**new Edit().Show();** - otvara formu „Edit“

## Forma „Zaposlenici“

Dizajn

***Slika na kojoj se prikazuje snimka zaslona, tekst, softver, Ikona na računalu

Opis je automatski generiran***

***Dizajn forme „Zaposlenici“***

Dizajn ove forme se sastoji od listBox-a, 3 button-a, textBoxa i comboBoxa

**„Dodaj“** – otvara formu „Zaposlenici\_adder“

**„Uredi“ -** otvara formu „Zaposlenici\_adder“

**„Obriši“ –** briše označene elemente u listBox-u

**„Search“ –** textBox pomoću kojeg pretražujemo elemente u listBoxu

**comboBox** – preko njega filtriramo elemente u listBoxu

Programski kod

Slika na kojoj se prikazuje tekst, snimka zaslona, softver

Opis je automatski generiran

***Programski kod forme „Zaposlenici“***

**podatkovniKontekst podatkovniKontekst = new podatkovniKontekst();** - Ovdje se stvara nova instanca klase podatkovniKontekst.

**listBox2.Items.Clear();** - Ovdje se brišu sve stavke (items) iz listBox2.

**List<Model.Zaposlenici> zaposleni = podatkovniKontekst.UcitajZaposlenike();** - Ovdje se poziva metoda UcitajZaposlenike() iz objekta podatkovniKontekst koja vraća listu objekata tipa Zaposlenici

**if (zaposleniciPrikaz.Count > 0)** - Ovdje se provjerava da li zaposleniciPrikaz, pretpostavljena druga lista zaposlenika, ima više od 0 elemenata.

**foreach (Model.Zaposlenici zap in zaposleniciPrikaz)** - Ovo je petlja koja prolazi kroz sve elemente liste zaposleniciPrikaz. Svaki element liste smješta se u varijablu zap, a zatim se taj element dodaje u listBox2.

**else** - Ovaj blok se izvršava ako zaposleniciPrikaz.Count nije veći od 0, što znači da nema elemenata u zaposleniciPrikaz.

**foreach (Model.Zaposlenici zap in zaposleni)** - Ovo je druga petlja koja prolazi kroz sve elemente liste zaposleni koja sadrži sve zaposlenike dohvaćene iz podatkovniKontekst. Svaki zaposlenik se dodaje u listBox2.

Slika na kojoj se prikazuje tekst, snimka zaslona, softver, Multimedijski softver

Opis je automatski generiran***Programski kod searchBox-a***

**zaposleniciPrikaz.Clear();** - Ovdje se brišu svi postojeći podaci iz liste zaposleniciPrikaz. To se radi kako bi se osiguralo da lista bude prazna prije nego što se dodaju novi podaci.

**podatkovnikontekst = new podatkovniKontekst**(); - Stvara se nova instanca podatkovniKontekst. Ova linija može biti suvišna ako je podatkovniKontekst već stvoren negdje drugdje u klasi.

**List<Model.Zaposlenici> zaposleniciKomplet = podatkovnikontekst.UcitajZaposlenike();** - Poziva se metoda UcitajZaposlenike() iz objekta podatkovniKontekst kako bi se dohvatila kompletna lista zaposlenika.

**foreach (Model.Zaposlenici zap in zaposleniciKomplet)** - Ovo je petlja koja prolazi kroz svakog zaposlenika u listi zaposleniciKomplet.

**if (zap.Prezime.Contains(searchBox.Text.ToLower())) zaposleniciPrikaz.Add(zap);** - Ovdje se provjerava da li prezime (Prezime) zaposlenika sadrži tekst koji je unesen u searchBox. Ako sadrži, taj zaposlenik se dodaje u listu zaposleniciPrikaz.

**addZap();** - Poziva se funkcija addZap() kako bi se ažurirao prikaz liste zaposlenika na osnovu promjena koje su napravljene pretraživanjem. Ova funkcija dodaje zaposlenike iz liste zaposleniciPrikaz u listBox2, kako bi se prikazali rezultati pretraživanja.

Slika na kojoj se prikazuje tekst, snimka zaslona, softver

Opis je automatski generiran

***Programski kod za ComboBox***

**zaposleniciPrikaz.Clear();** - Ovdje se brišu svi postojeći podaci iz liste zaposleniciPrikaz. To se radi kako bi se osiguralo da lista bude prazna prije nego što se dodaju novi podaci.

**podatkovnikontekst = new podatkovniKontekst**(); - Stvara se nova instanca podatkovniKontekst. Ova linija može biti suvišna ako je podatkovnikontekst već stvoren negdje drugdje u klasi.

**List<Model.Zaposlenici> zaposleniciFilter = this.podatkovnikontekst.UcitajZaposlenike();** - Poziva se metoda UcitajZaposlenike() iz objekta podatkovnikontekst kako bi se dohvatila kompletna lista zaposlenika.

**if (filterBox.SelectedItem.ToString() == "Godine")** - Ovdje se provjerava je li odabrana stavka u filterBox (vjerojatno ComboBox) jednaka "Godine".

**zaposleniciFilter.Sort((z1, z2) => z2.Godine.CompareTo(z1.Godine));** - Ako je odabrana stavka "Godine", lista zaposleniciFilter se sortira prema atributu Godine u silaznom redoslijedu.

**zaposleniciFilter.Reverse();** - Nakon sortiranja, lista se invertira kako bi bila u rastućem redoslijedu godina.

**foreach (Model.Zaposlenici zap in zaposleniciFilter)** - Ovdje se prolazi kroz sve zaposlenike u sortiranoj listi zaposleniciFilter.

**zaposleniciPrikaz.Add(zap);** - Svaki zaposlenik se dodaje u listu zaposleniciPrikaz. Sljedeće dva bloka if rade istu logiku kao i prvi, ali s drugim atributom po kojem se sortira lista. Ako je odabrana stavka "Ime", zaposlenici će biti sortirani po imenu.

**addZap();** - Poziva se funkcija addZap() kako bi se ažurirao prikaz liste zaposlenika na temelju promjena koje su napravljene filtriranjem i sortiranjem. Ova funkcija dodaje zaposlenike iz liste zaposleniciPrikaz u listBox2, kako bi se prikazali rezultati filtriranja i sortiranja.

## Forma „Zaposlenici i zivotinje“

Dizajn

Slika na kojoj se prikazuje tekst, snimka zaslona, softver, Trokut

Opis je automatski generiran

***Dizajn forme „Zaposlenici i zivotinje“***

Dizajn ove forme se sastoji od 2 listBox-a , button-a, comboBox-a i DateTimePicker-a

**2 listBox-a –** listBox-evi od klasa „zivotinje“ i „zaposlenici“

**DateTimePicker –** odabir datuma i vremena

**comboBox –** odabir padajućim izbornikom

**„Obriši“ –** brišemo odabirom ono iz listBox-eva

## Forma „Edit“

Dizajn

Slika na kojoj se prikazuje snimka zaslona, tekst, softver, Ikona na računalu

Opis je automatski generiran

***Dizajn forme „Edit“***

Dizajn ove forme se sastoji od 3 button-a i 4 textBox-a